OpenShift Container Platform 4.8

インストール

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

Last Updated: 2023-06-02

OpenShift Container Platform 4.8 インストール

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

法律上の通知

概要

本書では、OpenShift Container Platform のインストール方法と、一部の設定プロセスの詳細について説明します。

目次

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

1.1. OPENSHIFT CONTAINER PLATFORM インストールの概要

OpenShift Container Platform インストールプログラムは柔軟性を提供します。インストールプログラムを使用して、インストールプログラムがプロビジョニングし、クラスターで維持するインフラストラクチャーでクラスターをデプロイしたり、ユーザーが独自に準備し、維持するインフラストラクチャーでクラスターをデプロイしたりすることができます。

OpenShift Container Platform クラスターの基本的な 2 つのタイプとして、インストーラーでプロビジョニングされるインフラストラクチャークラスターとユーザーによってプロビジョニングされるインフラストラクチャークラスターがあります。

これらのクラスターのタイプにはどちらにも以下の特徴があります。

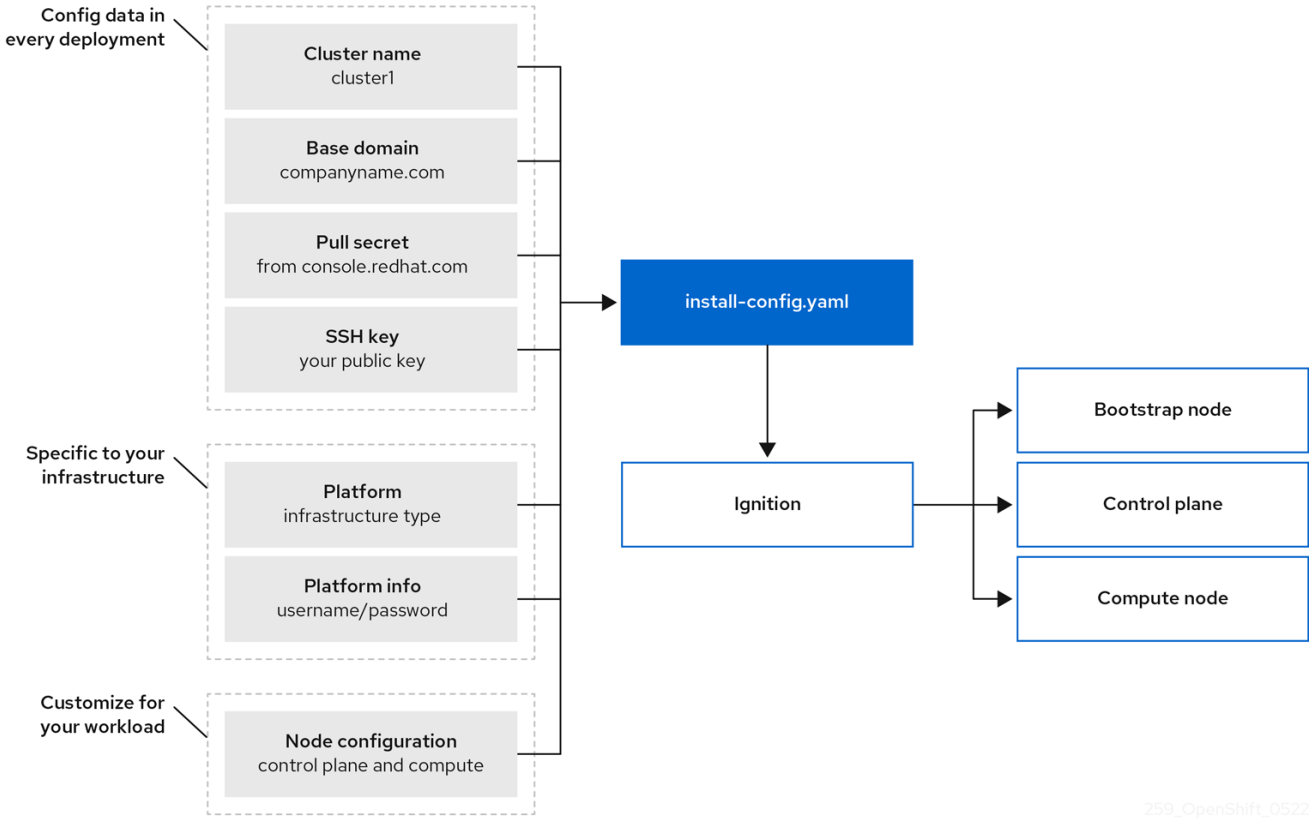
- 単一障害点のない可用性の高いインフラストラクチャーがデフォルトで利用可能である。
- 管理者は適用される更新内容および更新タイミングを制御できる。

同一のインストールプログラムを使用してこれらクラスターの両方のタイプをデプロイできます。インストールプログラムで生成される主なアセットは ブートストラップ、マスターおよびワーカーマシンの Ignition 設定です。これらの 3 つの設定および適切に設定されたインフラストラクチャーを使用して、OpenShift Container Platform クラスターを起動することができます。

OpenShift Container Platform インストールプログラムは、クラスターのインストールを管理するために一連のターゲットおよび依存関係を使用します。インストールプログラムには、達成する必要のある一連のターゲットが設定され、それぞれのターゲットには一連の依存関係が含まれます。各ターゲットはそれぞれの依存関係の条件が満たされ次第、別個に解決されるため、インストールプログラムは複数のターゲットを並行して達成できるように動作します。最終的なターゲットはクラスターを実行することです。コマンドを実行するのではなく依存関係の条件を満たすことにより、インストールプログラムは、コマンドを実行してコンポーネントを再度作成する代わりに、既存のコンポーネントを認識し、それらを使用することができます。

以下の図は、インストールのターゲットと依存関係のサブセットを示しています。

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



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

OpenShift Container Platform 4.8 クラスターのすべてのコントロールプレーンマシンは、Ignition と呼ばれる最初の起動時に使用される重要なプロビジョニングツールが含まれる RHCOS を使用する必要があります。このツールは、クラスターのマシンの設定を可能にします。オペレーティングシステムの更新は、Operator によってクラスター全体に展開されるコンテナーイメージに組み込まれる Atomic OSTree リポジトリーとして提供されます。実際のオペレーティングシステムの変更については、rpm-ostree を使用する atomic 操作として各マシンでインプレースで実行されます。これらのテクノロジーを組み合わせて使用することにより、OpenShift Container Platform では、プラットフォーム全体を最新の状態に保つインプレースアップグレードで、その他のアプリケーションをクラスターで管理するかのようにオペレーティングシステムを管理することができます。これらのインプレースアップグレードにより、オペレーションチームの負担を軽減することができます。

RHCOS をすべてのクラスターマシンのオペレーティングシステムとして使用する場合、クラスターはオペレーティングシステムを含む、そのコンポーネントとマシンのすべての側面を管理します。このため、インストールプログラムと Machine Config Operator のみがマシンを変更することができます。インストールプログラムは Ignition 設定ファイルを使用して各マシンの状態を設定し、Machine Config Operator はインストール後に、新規証明書またはキーの適用などのマシンへの変更を実行します。

1.1.1. インストールプロセス

OpenShift Container Platform クラスターをインストールする場合、インストールプログラムを OpenShift Cluster Manager サイトの適切な インフラストラクチャープロバイダー ページからダウンロードします。このサイトでは以下を管理しています。

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

OpenShift Container Platform 4.8 では、インストールプログラムは、一連のアセットに対して一連のファイル変換を実行する Go バイナリーファイルです。インストールプログラムと対話する方法は、インストールタイプによって異なります。

- インストーラーでプロビジョニングされるインフラストラクチャーのクラスターの場合、インフラストラクチャーのブートストラップおよびプロビジョニングは、ユーザーが独自に行うのではなくインストールプログラムが代行します。インストールプログラムは、クラスターをサポートするために必要なネットワーク、マシン、およびオペレーティングシステムのすべてを作成します。
- クラスターのインフラストラクチャーを独自にプロビジョニングし、管理する場合には、ブートストラップマシン、ネットワーク、負荷分散、ストレージ、および個々のクラスターマシンを含む、すべてのクラスターインフラストラクチャーおよびリソースを指定する必要があります。

インストール時には、お使いのマシンタイプ用の **install-config.yaml** という名前のインストール設定ファイル、Kubernetes マニフェスト、および Ingition 設定ファイルの 3 つのファイルセットを使用します。

重要

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

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

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

重要

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

インストーラーでプロビジョニングされるインフラストラクチャーでのインストールプロセス
デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーです。デフォルトで、インストールプログラムはインストールウィザードとして機能し、独自に判別できない値の入力を求めるプロンプトを出し、残りのパラメーターに妥当なデフォルト値を提供し

ます。インストールプロセスは、高度なインフラストラクチャーシナリオに対応するようカスタマイズすることもできます。インストールプログラムは、クラスターの基盤となるインフラストラクチャーをプロビジョニングします。

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

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

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

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

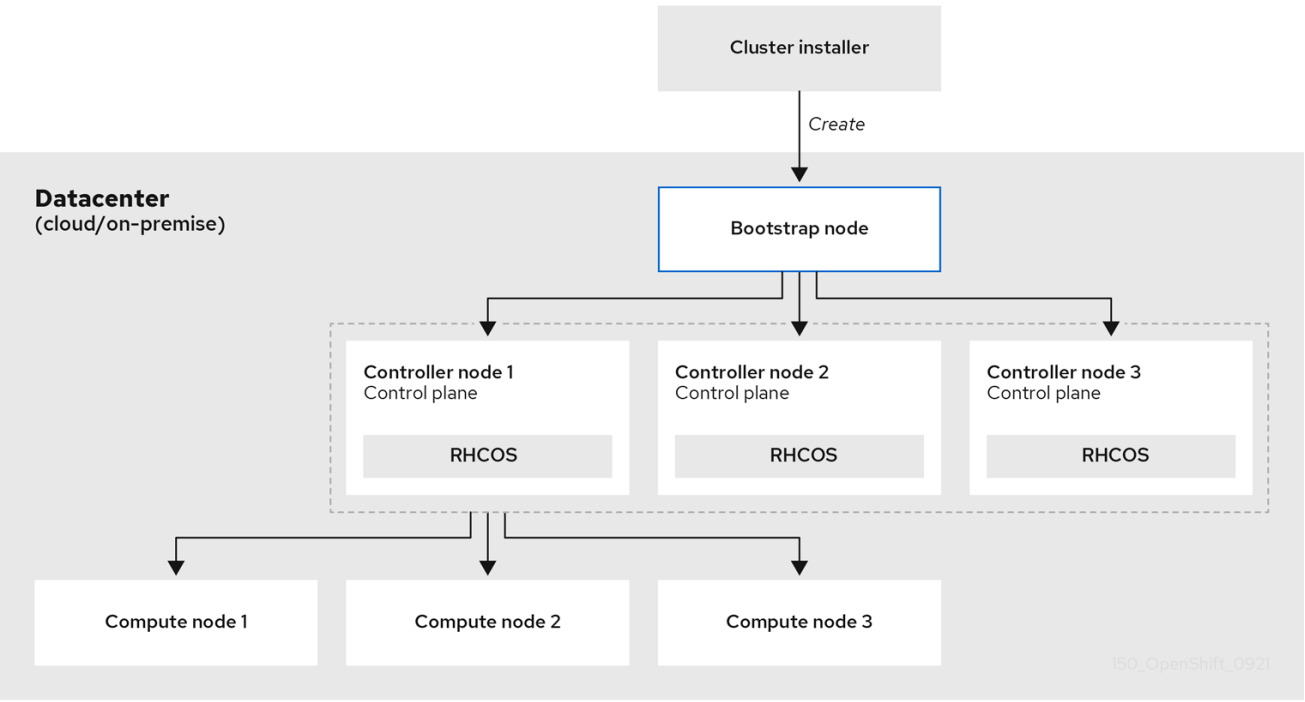
インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合、以下を含むクラスターリソースを管理し、維持する必要があります。

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

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

インストールプロセスの詳細
クラスターの各マシンにはプロビジョニング時にクラスターについての情報が必要になるため、OpenShift Container Platform は初期設定時に一時的な **bootstrap** マシンを使用し、必要な情報を永続的なコントロールマシンに提供します。これは、クラスターの作成方法を記述する Ignition 設定ファイルを使用して起動されます。ブートストラップマシンは、コントロールプレーンを設定するコントロールプレーンマシン (別名マスターマシン) を作成します。その後、コントロールプレーンマシンはコンピュートマシン (ワーカーマシンとしても知られる) を作成します。以下の図はこのプロセスを示しています。

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



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

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

クラスターのブートストラップには、以下のステップが関係します。

1. ブートストラップマシンが起動し、コントロールプレーンマシンの起動に必要なリモートリソースのホスティングを開始します。(ユーザーがインフラストラクチャーをプロビジョニングする場合には手動の介入が必要になります)。

2. ブートストラップマシンは、単一ノードの etcd クラスターと一時的な Kubernetes コントロールプレーンを起動します。

3. コントロールプレーンマシンは、ブートストラップマシンからリモートリソースをフェッチし、起動を終了します。(ユーザーがインフラストラクチャーをプロビジョニングする場合には手動の介入が必要になります)。

4. 一時的なコントロールプレーンは、実稼働コントロールプレーンマシンに対して実稼働コントロールプレーンをスケジュールします。

5. Cluster Version Operator (CVO) はオンラインになり、etcd Operator をインストールします。etcd Operator はすべてのコントロールプレーンノードで etcd をスケールアップします。

6. 一時的なコントロールプレーンはシャットダウンし、コントロールを実稼働コントロールプレーンに渡します。

7. ブートストラップマシンは OpenShift Container Platform コンポーネントを実稼働コントロールプレーンに挿入します。

8. インストールプログラムはブートストラップマシンをシャットダウンします。(ユーザーがインフラストラクチャーをプロビジョニングする場合には手動の介入が必要になります)。

9. コントロールプレーンはコンピュートノードを設定します。

10. コントロールプレーンは一連の Operator の形式で追加のサービスをインストールします。

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

1.1.2. インストール後のノード状態の確認

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

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

注記

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

インストールが完了したら、以下の手順を使用してクラスター内のノードの状態を監視し続けることができます。

前提条件

- インストールプログラムはターミナルで正常に解決されます。

手順

1. すべてのワーカーノードのステータスを表示します。

```
$ oc get nodes
```

出力例

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

2. すべてのワーカーマシンノードのフェーズを表示します。

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

出力例

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

関連情報

- インストール後
- インストールの検証

インストールのスコープ

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

関連情報

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

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

OpenShift Container Platform バージョン 4.8 では、インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの場合、以下のプラットフォームにインストールできます。

- Amazon Web Services (AWS)
- Google Cloud Platform (GCP)
- Microsoft Azure
- Red Hat OpenStack Platform (RHOSP) バージョン 13 および 16
    - OpenShift Container Platform の最新リリースは、最新の RHOSP のロングライフリリースおよび中間リリースの両方をサポートします。RHOSP リリースの互換性についての詳細は、OpenShift Container Platform on RHOSP support matrix を参照してください。
- Red Hat Virtualization (RHV)
- VMware vSphere
- VMware Cloud (VMC) on AWS
- ベアメタル

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

重要

インストール後は、以下の変更はサポートされません。

- クラウドプロバイダープラットフォームの組み合わせ
- クラスターがインストールされているプラットフォームとは異なるプラットフォームの永続ストレージフレームワークを使用するなどの、クラウドプロバイダーのコンポーネントの組み合わせ

OpenShift Container Platform バージョン 4.8 では、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターの場合、以下のプラットフォームにインストールできます。

- AWS
- Azure
- GCP
- RHOSP
- RHV
- VMware vSphere
- VMware Cloud on AWS
- ベアメタル
- IBM Z または LinuxONE
- IBM Power Systems

プラットフォームでサポートされているケースに応じて、ユーザーがプロビジョニングしたインフラストラクチャーにインストールすると、完全なインターネットアクセスでマシンを実行したり、クラスターをプロキシーの背後に配置したり、**制限付きネットワークインストール** を実行したりできます。ネットワークが制限された環境でのインストールでは、クラスターのインストールに必要なイメージをダウンロードして、ミラーレジストリーに配置し、そのデータを使用してクラスターをインストールできます。vSphere またはベアメタルインフラストラクチャーのネットワークが制限されたインストールでは、プラットフォームコンテナーのイメージをプルするためにインターネットにアクセスする必要がありますが、クラスターマシンはインターネットへの直接のアクセスを必要としません。

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

関連情報

- それぞれのサポートされているプラットフォームで利用できるインストールのタイプについての詳細は、各種プラットフォームのサポートされているインストール方法 を参照してください。
- インストール方法を選択し、必要なリソースを準備する方法については、クラスターインストール方法の選択およびそのユーザー向けの準備 を参照してください。

第2章 クラスターインストール方法の選択およびそのユーザー向けの準備

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

2.1. クラスターのインストールタイプの選択

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

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

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

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)
- Red Hat OpenStack Platform (RHOSP)
- Red Hat Virtualization (RHV)
- IBM Z および LinuxONE
- Red Hat Enterprise Linux (RHEL) KVM 用の IBM Z および LinuxONE
- IBM Power
- VMware vSphere
- VMware Cloud (VMC) on AWS
- ベアメタルまたはその他のプラットフォームに依存しないインフラストラクチャー

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

OpenShift Container Platform を使用する必要があるが、クラスターを独自に管理することを望まない場合は、複数のマネージドサービスオプションを使用できます。Red Hat によって完全に管理されるクラスターが必要な場合は、OpenShift Dedicated または OpenShift Online を使用することができます。OpenShift を Azure、AWS、IBM Cloud、または Google Cloud でマネージドサービスとして使用することもできます。マネージドサービスの詳細は、OpenShift の製品 ページを参照してください。クラウド仮想マシンを仮想ベアメタルとして OpenShift Container Platform クラスターをインストールする場合、対応するクラウドベースのストレージはサポートされません。

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

OpenShift Container Platform 3 を使用したことがあり、OpenShift Container Platform 4 を使用してみ

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

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

2.1.3. クラスターで既存のコンポーネントを使用する必要がありますか？

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

インストーラーでプロビジョニングされるインフラストラクチャークラスターは、クラスターまたはその基礎となるマシンを、AWS、Azure、または GCP、または VMC on AWS に対してカスタマイズせずにデプロイできます。これらのインストール方法は、実稼働対応の OpenShift Container Platform クラスターをデプロイする最も高速な方法です。

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

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

インストーラーでプロビジョニングされるインフラストラクチャー方法を使用して、RHOSP、RHOSP with Kuryr、RHV、vSphere、および bare metal のハードウェアに適切なマシンインスタンスを作成できます。

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

ユーザーによってプロビジョニングされるインフラストラクチャーは、既存のハードウェアで実行することもできます。RHOSP、RHOSP on SR-IOV、RHV、IBM Z or LinuxONE、IBM Power、または vSphere を使用する場合は、特定のインストール手順を使用してクラスターをデプロイします。サポートされる他のハードウェアを使用する場合は、ベアメタルのインストール の手順に従います。

2.1.4. クラスターに追加のセキュリティーが必要ですか？

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

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

非接続のクラスターまたはネットワークが制限されたクラスターなど、インターネットへのアクセスが限定されたクラスターをインストールする必要がある場合、インストールパッケージをミラーリング し、そこからクラスターをインストールできます。AWS、GCP、IBM Z or LinuxONE、IBM Z or LinuxONE with RHEL KVM、IBM Power、vSphere、VMC on AWS、または bare metal のネットワークが制限された環境へのユーザーによってプロビジョニングされるインフラストラクチャーのインストールの詳細な手順を実行します。AWS、GCP、VMC on AWS、RHOSP、RHV、および vSphere の詳細な手順に従って、インストーラーでプロビジョニングされるインフラストラクチャーを使用してクラスターをネットワークが制限された環境にインストールすることもできます。

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

また、インストール時に FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリー を使用するようにクラスターマシンを設定することもできます。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

2.2. インストール後のユーザー向けのクラスターの準備

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

実稼働クラスターの場合、以下の統合を設定する必要があります。

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

2.3. ワークロードについてのクラスターの準備

ワークロードのニーズによっては、アプリケーションのデプロイを開始する前に、追加の手順が必要になる場合があります。たとえば、アプリケーションの ビルドストラテジー をサポートするインフラス

トラクチャーを準備した後に、低レイテンシー のワークロードをプロビジョニングしたり、機密のワークロードを保護 したりする必要がある場合があります。アプリケーションワークロードの モニターリング を設定することもできます。Windows ワークロード を実行する予定の場合は、インストールプロセス時に OVN-Kubernetes を使用したハイブリッドネットワーク を有効にする必要があります。ハイブリッドネットワークは、クラスターのインストール後に有効にすることはできません。

2.4. 各種プラットフォームのサポートされているインストール方法

各種のプラットフォームで各種のインストールを実行できます。

注記

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

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

	AWS	Azure	GCP	RHOSP	RHV	ベアメタル	vSphere	VMC	IBM Z	IBM Power
デフォルト	✓	✓	✓		✓	✓	✓	✓		
カスタム	✓	✓	✓	✓	✓		✓	✓		
ネットワークのカスタマイズ	✓	✓	✓	✓			✓	✓		
ネットワークが制限されたインストール	✓		✓	✓	✓	✓	✓	✓		
プライベートクラスター	✓	✓	✓							
既存の仮想プライベートネットワーク	✓	✓	✓							

	AWS	Azure	GCP	RHOSP	RHV	ベアメタル	vSphere	VMC	IBM Z	IBM Power
government リージョン	✓	✓								

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

	AWS	Azure	GCP	RHOSP	SR-IOV上のRHOSP	RHV	ベアメタル	vSphere	VMC	IBM Z	RHEL KVMを使用したIBM Z	IBM Power	プラットフォームの指定なし
カスタム	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ネットワークのカスタマイズ				✓			✓	✓	✓				
ネットワークが制限されたインストール	✓		✓				✓	✓	✓	✓	✓	✓	

	AWS	Azure	GCP	RHOSP	SR-IOV上のRHOSP	RHV	ベアメタル	vSphere	VMC	IBM Z	RHEL KVMを使用したIBM Z	IBM Power	プラットフォームの指定なし
クラスタープロジェクト外でホストされる共有VPC			✓										

第3章 非接続インストールのイメージのミラーリング

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

重要

必要なコンテナーイメージを取得するには、インターネットへのアクセスが必要です。この手順では、ご使用のネットワークとインターネットのどちらにもアクセスできるミラーホストにミラーレジストリーを配置します。ミラーホストへのアクセスがない場合は、Operator カタログのミラーリング 手順に従って、イメージをネットワークの境界をまたがって移動できるデバイスにコピーします。

3.1. 前提条件

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

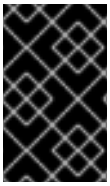
  Red Hat Quay のエンタイトルメントをお持ちの場合は、Red Hat Quay のデプロイに関するドキュメント 概念実証 (実稼働以外) 向けの Red Hat Quay のデプロイ または Quay Operator の使用による OpenShift への Red Hat Quay のデプロイ を参照してください。レジストリーの選択およびインストールがにおいてさらにサポートが必要な場合は、営業担当者または Red Hat サポートにお問い合わせください。

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

3.2. ミラーレジストリーについて

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

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



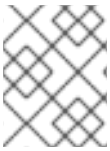
重要

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

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

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

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



注記

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

関連情報

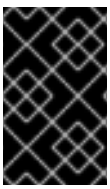
CRI-O ログを表示してイメージソースを表示する方法は、イメージのプルソースの表示 を参照してください。

3.3. ミラーホストの準備

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

3.3.1. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

3.4. イメージのミラーリングを可能にする認証情報の設定

Red Hat からミラーへのイメージのミラーリングを可能にするコンテナーイメージレジストリーの認証情報ファイルを作成します。

警告

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

警告

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

前提条件

- 切断された環境で使用するミラーレジストリーを設定しました。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。

- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. **registry.redhat.io** プルシークレットを Red Hat OpenShift Cluster Manager からダウンロードし、**.json** ファイルに保存します。

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

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

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

3. JSON 形式でプルシークレットのコピーを作成します。

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

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

4. ファイルを ~/**.docker/config.json** または **$XDG_RUNTIME_DIR/containers/auth.json** として保存します。
ファイルの内容は以下の例のようになります。

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

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

```
  "auths": {
```

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

1. **<mirror_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:8443**

2. **<credentials>** については、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

ファイルは以下の例のようになります。

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

3.5. RED HAT OPENSHIFT のミラーレジストリー

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

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

前提条件

- OpenShift Container Platform サブスクリプション

- Podman 3.3 および OpenSSL がインストールされた Red Hat Enterprise Linux (RHEL) 8。
- Red Hat Quay サービスの完全修飾ドメイン名。DNS サーバーを介して解決する必要があります。
- ターゲットホストでパスワードなしに使用できる **sudo** アクセス権。
- ターゲットホストでのキーベースの SSH 接続。SSH キーは、ローカルインストール用に自動的に生成されます。リモートホストの場合は、独自の SSH キーを生成する必要があります。
- vCPU 2 つ以上。
- RAM 8 GB。
- OpenShift Container Platform 4.8 リリースイメージの場合は約 8.7 GB、OpenShift Container Platform 4.8 リリースイメージおよび OpenShift Container Platform 4.8 Red Hat Operator イメージの場合は約 668 GB。ストリームあたり最大 1TB 以上が推奨されます。

重要

これらの要件は、リリースイメージと Operator イメージだけでテストしたローカルテスト結果に基づいています。ストレージ要件は、組織のニーズによって異なります。ユーザーによって、複数の z ストリームをミラーリングするなどの場合に、より多くのスペースを必要とする場合があります。標準の Red Hat Quay 機能を使用して、不要なイメージを削除し、スペースを解放できます。

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

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

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

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

Red Hat Openshift 導入用のミラーレジストリーは、リリースイメージや Red Hat Operator イメージなど、切断された OpenShift Container Platform クラスターのインストールに必要なイメージのホスティングに限定されます。Red Hat Enterprise Linux (RHEL) マシンのローカルストレージを使用して、RHEL でサポートされるストレージは、**Red Hat Openshift 導入用のミラーレジストリー**でサポートされます。お客様が作成したコンテンツは、**Red Hat Openshift 導入用のミラーレジストリー**でホストしないでください。

Red Hat Quay とは異なり、**Red Hat Openshift 導入用のミラーレジストリー**は高可用性レジストリーではなく、ローカルファイルシステムストレージのみがサポートされています。クラスターのグループの更新時にクラスターが複数あると単一障害点を生み出す可能性があるため、複数のクラスターで **Red Hat Openshift 導入用のミラーレジストリー**を使用することはお勧めしません。**Red Hat Openshift 導入用のミラーレジストリー**を活用して、OpenShift Container Platform コンテンツを他のクラスターに提供できる Red Hat Quay などの実稼働環境レベルの高可用性レジストリーをホストできるクラスターをインストールすることをお勧めします。

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

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

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

注記

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

手順

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

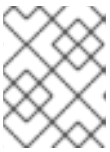
2. **mirror-registry** ツールを使用して、現在のユーザーアカウントでローカルホストに**Red Hat Openshift 導入用のミラーレジストリー** をインストールします。使用可能なフラグの完全なリストは、Red Hat OpenShift フラグのミラーレジストリーを参照してください。

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

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

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

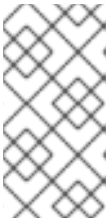
1 生成された root CA 証明書を信頼するようにシステムを設定して、**--tls-verify=false** の実行を回避できます。詳細は、SSL を使用した Red Hat Quay への接続の保護および認証局を信頼するようにシステムを設定するを参照してください。



注記

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

4. ログイン後、OpenShift Container Platform イメージをミラーリングできます。必要性に応じて、本書の OpenShift Container Platform イメージリポジトリーのミラーリングまたは Operator カタログのミラーリングのセクションを参照してください。

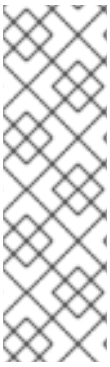


注記

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

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

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



注記

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

手順

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

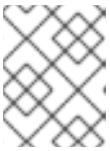
2. **mirror-registry** ツールを使用して、現在のユーザーアカウントでローカルホストに**Red Hat Openshift 導入用のミラーレジストリー** をインストールします。使用可能なフラグの完全なリストは、Red Hat OpenShift フラグのミラーレジストリーを参照してください。

```
$ sudo ./mirror-registry install -v \
  --targetHostname <host_example_com> \
  --targetUsername <example_user> \
  -k ~/.ssh/my_ssh_key \
  --quayHostname <host_example_com> \
  --quayRoot <example_directory_name>
```

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

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

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



注記

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

4. ログイン後、OpenShift Container Platform イメージをミラーリングできます。必要性に応じて、本書の OpenShift Container Platform イメージリポジトリーのミラーリングまたは Operator カタログのミラーリングのセクションを参照してください。

注記

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

3.6. RED HAT OPENSHIFT のミラーレジストリーのアップグレード

- 次のコマンドを実行して、ローカルホストから **Red Hat OpenShift のミラーレジストリー** をアップグレードできます。

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

注記

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

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

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

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

注記

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

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

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

Flags	説明
--autoApprove	対話型プロンプトを無効にするブール値。**true** に設定すると、ミラーレジストリーをアンインストールするときに **quayRoot** ディレクトリーが自動的に削除されます。指定しない場合には、デフォルトは **false** に設定されます。
--initPassword	Quay のインストール中に作成された init ユーザーのパスワード。空白を含まず、8 文字以上にする必要があります。
--initUser string	初期ユーザーのユーザー名を表示します。指定しない場合、デフォルトで **init** になります。
--quayHostname	クライアントがレジストリーへの接続に使用するミラーレジストリーの完全修飾ドメイン名。**Quayconfig.yaml** の **SERVER_HOSTNAME** に相当します。DNS で解決する必要があります。指定しない場合は、デフォルトは **<target Hostname>:8443** です。[1]
--quayRoot, **-r**	**root CA.key**、**root CA.pem**、**root CA.srl** 証明書など、コンテナーイメージレイヤーと設定データが保存されるディレクトリー。OpenShift Container Platform 4.8 リリースイメージの場合は約 8.7 GB、OpenShift Container Platform 4.8 リリースイメージおよび OpenShift Container Platform 4.8 Red Hat Operator イメージの場合は約 668 GB が必要です。指定しない場合、デフォルトは / **etc/quay-install** になります。
--ssh-key, **-k**	SSH ID キーのパス。指定しない場合、デフォルトは ~/**.ssh/quay_installer** です。
--sslCert	SSL/TLS 公開鍵/証明書へのパス。デフォルトは **{quay Root}/quady-config** で、指定しない場合は自動生成されます。

Flags	説明
--sslCheckSkip	**config.yaml** ファイルの **SERVER_HOSTNAME** に対する証明書のホスト名のチェックをスキップします。[2]
--sslKey	HTTPS 通信に使用される SSL/TLS 秘密鍵へのパス。デフォルトは **{quay Root}/quady-config** で、指定しない場合は自動生成されます。
--targetHostname, -H	Quay のインストール先のホスト名。デフォルトは **$HOST** になります。たとえば、指定していない場合にはローカルホストになります。
--targetUsername, -u	SSH に使用するターゲットホストのユーザー。デフォルトは **$USER** です。たとえば、指定しない場合は現在のユーザーになります。
--verbose, -v	デバッグログと Ansible Playbook の出力を表示します。
--version	**Red Hat Openshift 導入用のミラーレジストリー** のバージョンを表示します。

1. システムのパブリック DNS 名がローカルホスト名と異なる場合は、**--quayHostname** を変更する必要があります。
2. **--ssl Check Skip** は、ミラーレジストリーがプロキシーの背後に設定されており、公開されているホスト名が内部の Quay ホスト名と異なる場合に使用されます。また、インストール中に、指定した Quay ホスト名に対して証明書の検証を行わない場合にも使用できます。

関連情報

- Red Hat Quay への接続を保護するための SSL の使用
- 認証局を信頼するようにシステムを設定する
- OpenShift Container Platform イメージリポジトリーのミラーリング
- Operator カタログのミラーリング

3.7. OPENSHIFT CONTAINER PLATFORM イメージリポジトリーのミラーリング

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

前提条件

- ミラーホストがインターネットにアクセスできる。
- ネットワークが制限された環境で使用するミラーレジストリーを設定し、設定した証明書および認証情報にアクセスできる。
- Red Hat OpenShift Cluster Manager からプルシークレット をダウンロードし、ミラーリポジトリーへの認証を含めるようにこれを変更している。

- Subject Alternative Name が設定されていない自己署名証明書を使用する場合は、この手順の **oc** コマンドの前に **GODEBUG=x509ignoreCN=0** を追加する必要があります。この変数を設定しない場合、**oc** コマンドは以下のエラーを出して失敗します。

```
x509: certificate relies on legacy Common Name field, use SANs or temporarily enable
Common Name matching with GODEBUG=x509ignoreCN=0
```

手順

ミラーホストで以下の手順を実行します。

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

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

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

   b. ローカルレジストリー名とホストポートをエクスポートします。

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

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

   c. ローカルリポジトリー名をエクスポートします。

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

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

   d. ミラーリングするリポジトリーの名前をエクスポートします。

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

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

   e. パスをレジストリープルシークレットにエクスポートします。

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

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

   f. リリースミラーをエクスポートします。

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

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

g. サーバーのアーキテクチャーのタイプをエクスポートします (例: **x86_64**)。

```
$ ARCHITECTURE=<server_architecture>
```

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

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

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

3. バージョンイメージをミラーレジストリーにミラーリングします。

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

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

ii. ミラーリングするイメージおよび設定マニフェストを確認します。

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

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

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

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

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

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

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

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

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

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

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

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

  注記

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

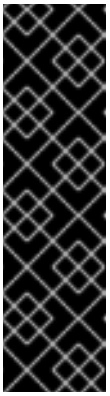
4. ミラーリングしたコンテンツをベースとしているインストールプログラムを作成するには、これを展開し、リリースに固定します。

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

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

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

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



重要

選択した OpenShift Container Platform バージョンに適したイメージを使用するには、ミラーリングされたコンテンツからインストールプログラムを展開する必要があります。

インターネット接続のあるマシンで、このステップを実行する必要があります。

非接続環境を使用している場合には、must-gather の一部として **--image** フラグを使用し、ペイロードイメージを参照します。

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

```
$ openshift-install
```

3.8. 非接続環境の CLUSTER SAMPLES OPERATOR

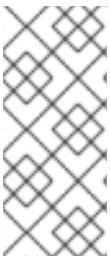
非接続環境で Cluster Samples Operator を設定するには、クラスターのインストール後に追加の手順を実行する必要があります。以下の情報を確認し、準備してください。

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

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

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

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



注記

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

この設定マップを、イメージストリームがインポートできるようにミラーリングする必要のあるイメージについての参照情報として使用できます。

- Cluster Samples Operator が **Removed** に設定される場合、ミラーリングされたレジストリーを作成するか、または使用する必要のある既存のミラーリングされたレジストリーを判別できます。
- 新しい設定マップをガイドとして使用し、ミラーリングされたレジストリーに必要なサンプルをミラーリングします。

- Cluster Samples Operator 設定オブジェクトの **skippedImagestreams** 一覧に、ミラーリングされていないイメージストリームを追加します。
- Cluster Samples Operator 設定オブジェクトの **samplesRegistry** をミラーリングされたレジストリーに設定します。
- 次に、Cluster Samples Operator を **Managed** に設定し、ミラーリングしたイメージストリームをインストールします。

3.9. 次のステップ

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

3.10. 関連情報

- must-gather の使用についての詳細は、特定の機能についてのデータの収集 を参照してください。

第4章 AWS へのインストール

4.1. AWS へのインストールの準備

4.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

4.1.2. OpenShift Container Platform OpenStack の AWS へのインストールについての要件

OpenShift Container Platform を Amazon Web Services (AWS) にインストールする前に、AWS アカウントを作成する必要があります。アカウント、アカウントの制限、アカウントのパーミッション、IAM ユーザーセットアップ、およびサポートされている AWS リージョンの設定についての詳細は、AWS アカウントの設定 を参照してください。

ご使用の環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、Amazon Web Services Security Token Service (AWS STS) を使用するための Cloud Credential Operator (CCO) の設定 を含む、他のオプションについて AWS の IAM の手動作成 を参照してください。

4.1.3. AWS に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

4.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる AWS インフラストラクチャーに、クラスターをインストールできます。

- **クラスターの AWS へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。デフォルトの設定オプションを使用して、クラスターを迅速にインストールできます。
- **カスタマイズされたクラスターの AWS へのインストール**: インストールプログラムがプロビジョニングする AWS インフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の数多くのカスタマイズオプションは、インストール後 に利用できます。

- **ネットワークのカスタマイズを使用したクラスターの AWS へのインストール**: インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **ネットワークが制限された環境での AWS へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、インストーラーでプロビジョニングされる AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。

- **クラスターの既存の Virtual Private Cloud へのインストール**: OpenShift Container Platform を既存の AWS Virtual Private Cloud (VPC) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **プライベートクラスターの既存の VPC へのインストール**: プライベートクラスターを既存の AWS VPC にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

- **クラスターの AWS の government またはシークレットリージョンへのインストール**: OpenShift Container Platform は、機密ワークロードをクラウドで実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されている AWS リージョンにデプロイできます。

4.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングする AWS インフラストラクチャーにクラスターをインストールできます。

- **独自に提供する AWS インフラストラクチャーへのクラスターのインストール**: 独自に提供する AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。提供される CloudFormation テンプレートを使用して、OpenShift Container Platform インストールに必要な各コンポーネントを表す AWS リソースのスタックを作成できます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したネットワークが制限された環境での AWS へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、独自に提供する AWS インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。ミラーリングされたコンテンツを使用して OpenShift Container Platform をインストールすることは可能ですが、クラスターが AWS API を使用するにはインターネットへのアクセスが必要です。

4.1.4. 次のステップ

- AWS アカウントの設定

4.2. AWS アカウントの設定

OpenShift Container Platform をインストールする前に、Amazon Web Services (AWS) アカウントを設定する必要があります。

4.2.1. Route 53 の設定

OpenShift Container Platform をインストールするには、使用する Amazon Web Services (AWS) アカウントに、Route 53 サービスの専用のパブリックホストゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。Route 53 サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、または AWS または他のソースから新規のものを取得できます。

注記

AWS で新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかります。AWS 経由でドメインを購入する方法についての詳細は、AWS ドキュメントの Registering Domain Names Using Amazon Route 53 を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を AWS に移行します。AWS ドキュメントの Making Amazon Route 53 the DNS Service for an Existing Domain を参照してください。

3. ドメインまたはサブドメインのパブリックホストゾーンを作成します。AWS ドキュメントの Creating a Public Hosted Zone を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。AWS ドキュメントの Getting the Name Servers for a Public Hosted Zone を参照してください。

5. ドメインが使用する AWS Route 53 ネームサーバーのレジストラーレコードを更新します。たとえば、別のアカウントを使ってドメインを Route 53 サービスに登録している場合は、AWS ドキュメントの Adding or Changing Name Servers or Glue Records のトピックを参照してください。

6. サブドメインを使用している場合は、その委任レコードを親ドメインに追加します。これにより、サブドメインの Amazon Route 53 の責任が付与されます。親ドメインの DNS プロバイダーによって要約された委任手順に従います。ハイレベルの手順の例については、AWS ドキュメントの Creating a subdomain that uses Amazon Route 53 as the DNS service without migrating the parent domain を参照してください。

4.2.1.1. AWS Route 53 の Ingress Operator エンドポイント設定

Amazon Web Services (AWS) GovCloud (US) US-West または US-East リージョンのいずれかにインストールする場合、Ingress Operator は Route53 およびタグ付けする API クライアントに **us-gov-west-1** リージョンを使用します。

Ingress Operator は、タグ付けするエンドポイントが 文字列 'us-gov-east-1' を含むように設定される場合、タグ付けする API エンドポイントとして **https://tagging.us-gov-west-1.amazonaws.com** を使用します。

AWS GovCloud (US) エンドポイントについての詳細は、GovCloud (US) についての AWS ドキュメントの Service Endpoints を参照してください。

重要

us-gov-east-1 リージョンにインストールする場合、プライベート、非接続インストールは AWS GovCloud ではサポートされません。

Route 53 設定の例

```
platform:
  aws:
    region: us-gov-west-1
    serviceEndpoints:
    - name: ec2
      url: https://ec2.us-gov-west-1.amazonaws.com
    - name: elasticloadbalancing
      url: https://elasticloadbalancing.us-gov-west-1.amazonaws.com
    - name: route53
      url: https://route53.us-gov.amazonaws.com (1)
    - name: tagging
      url: https://tagging.us-gov-west-1.amazonaws.com (2)
```

(1) Route 53 は、AWS GovCloud (US) リージョンの両方で **https://route53.us-gov.amazonaws.com** にデフォルト設定されます。

(2) US-West リージョンのみにタグ付けするためのエンドポイントがあります。クラスターが別のリージョンにある場合は、このパラメーターを省略します。

4.2.2. AWS アカウントの制限

OpenShift Container Platform クラスターは数多くの Amazon Web Services (AWS) コンポーネントを使用し、デフォルトの サービス制限 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。特定のクラスター設定を使用し、クラスターを特定の AWS リージョンにデプロイするか、またはアカウントを使って複数のクラスターを実行する場合、AWS アカウントの追加リソースを要求することが必要になる場合があります。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある AWS コンポーネントの制限を要約しています。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
インスタンスの制限	変動あり。	変動あり。	デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンノード (別名マスターノード) ● 3 つのワーカーノード これらのインスタンスタイプの数は、新規アカウントのデフォルト制限内の値です。追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、または異なるインスタンスタイプを使用するには、アカウントの制限を見直し、クラスターが必要なマシンをデプロイできることを確認します。 ほとんどのリージョンでは、ブートストラップおよびワーカーマシンは **m4.large** マシンを使用し、コントロールプレーンマシンは **m4.xlarge** インスタンスを使用します。これらのインスタンスタイプをサポートしないすべてのリージョンを含む一部のリージョンでは、**m5.large** および **m5.xlarge** インスタンスが代わりに使用されます。
Elastic IP (EIP)	0 - 1	アカウントごとに 5 つの EIP	クラスターを高可用性設定でプロビジョニングするために、インストールプログラムはそれぞれの リージョン内のアベイラビリティーゾーン にパブリックおよびプライベートのサブネットを作成します。各プライベートサブネットには NAT ゲートウェイ が必要であり、各 NAT ゲートウェイには別個の Elastic IP が必要です。AWS リージョンマップ を確認して、各リージョンにあるアベイラビリティーゾーンの数を判別します。デフォルトの高可用性を利用するには、少なくとも 3 つのアベイラビリティーゾーンがあるリージョンにクラスターをインストールします。アベイラビリティーゾーンが 6 つ以上あるリージョンにクラスターをインストールするには、EIP 制限を引き上げる必要があります。  **重要** **us-east-1** リージョンを使用するには、アカウントの EIP 制限を引き上げる必要があります。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
Virtual Private Cloud (VPC)	5	リージョンごとに 5 つの VPC	各クラスターは独自の VPC を作成します。
Elastic Load Balancing (ELB/NLB)	3	リージョンごとに 20	デフォルトで、各クラスターは、マスター API サーバーの内部および外部のネットワークロードバランサーおよびルーターの単一の Classic Elastic Load Balancer を作成します。追加の Kubernetes **Service** オブジェクトをタイプ **LoadBalancer** を指定してデプロイすると、追加のロードバランサーが作成されます。
NAT ゲートウェイ	5	アベイラビリティゾーンごとに 5 つ	クラスターは各アベイラビリティーゾーンに 1 つの NAT ゲートウェイをデプロイします。
Elastic Network Interface (ENI)	12 以上	リージョンごとに 350	デフォルトのインストールは 21 の ENI を作成し、リージョンの各アベイラビリティーゾーンに 1 つの ENI を作成します。たとえば、**us-east-1** リージョンには 6 つのアベイラビリティーゾーンが含まれるため、そのゾーンにデプロイされるクラスターは 27 の ENI を使用します。AWS リージョンマップ を確認して、各リージョンにあるアベイラビリティーゾーンの数を判別します。 追加の ENI が、クラスターの使用およびデプロイされたワークロード別に作成される追加のマシンおよび Elastic Load Balancer について作成されます。
VPC ゲートウェイ	20	アカウントごとに 20	各クラスターは、S3 アクセス用の単一の VPC ゲートウェイを作成します。
S3 バケット	99	アカウントごとに 100 バケット	インストールプロセスでは 1 つの一時的なバケットを作成し、各クラスターのレジストリーコンポーネントがバケットを作成するため、AWS アカウントごとに 99 の OpenShift Container Platform クラスターのみを作成できます。
セキュリティーグループ	250	アカウントごとに 2,500	各クラスターは、10 の個別のセキュリティーグループを作成します。

4.2.3. IAM ユーザーに必要な AWS パーミッション

注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例4.1 インストールに必要な EC2 パーミッション

- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**

- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例4.2 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**

- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**

注記

既存の VPC を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例4.3 インストールに必要な Elastic Load Balancing (ELB) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

例4.4 インストールに必要な Elastic Load Balancing (ELBv2) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**

例4.5 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**

- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**

注記

AWS アカウントに Elastic Load Balancer (ELB) を作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例4.6 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例4.7 インストールに必要な S3 パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**

- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例4.8 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

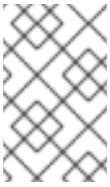
例4.9 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**

- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例4.10 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。代わりに、アカウントではネットワークリソースの削除に **tag:UntagResources** パーミッションのみが必要になります。

例4.11 共有インスタン出力ルが割り当てられたクラスターを削除するために必要なパーミッション

- **iam:UntagRole**

例4.12 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

注記

クラウドプロバイダーのクレデンシャルをミントモードで管理している場合に、IAM ユーザーには **iam:CreateAccessKey** と **iam:CreateUser** 権限も必要です。

例4.13 インスタンスのオプションのパーミッションおよびインストールのクォータチェック

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

4.2.4. IAM ユーザーの作成

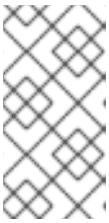
各 Amazon Web Services (AWS) アカウントには、アカウントの作成に使用するメールアドレスに基づく root ユーザーアカウントが含まれます。これは高度な権限が付与されたアカウントであり、初期ア

カウントにのみ使用し、請求設定また初期のユーザーセットの作成およびアカウントのセキュリティー保護のために使用することが推奨されています。

OpenShift Container Platform をインストールする前に、セカンダリー IAM 管理ユーザーを作成します。AWS ドキュメントの Creating an IAM User in Your AWS Account 手順を実行する際に、以下のオプションを設定します。

手順

1. IAM ユーザー名を指定し、**Programmatic access** を選択します。
2. **AdministratorAccess** ポリシーを割り当て、アカウントにクラスターを作成するために十分なパーミッションがあることを確認します。このポリシーはクラスターに対し、各 OpenShift Container Platform コンポーネントに認証情報を付与する機能を提供します。クラスターはコンポーネントに対し、それらが必要とする認証情報のみを付与します。



注記

必要なすべての AWS パーミッションを付与し、これをユーザーに割り当てるポリシーを作成することは可能ですが、これは優先されるオプションではありません。クラスターには追加の認証情報を個別コンポーネントに付与する機能がないため、同じ認証情報がすべてのコンポーネントによって使用されます。

3. オプション: タグを割り当て、メタデータをユーザーに追加します。
4. 指定したユーザー名に **AdministratorAccess** ポリシーが付与されていることを確認します。
5. アクセスキー ID およびシークレットアクセスキーの値を記録します。ローカルマシンをインストールプログラムを実行するように設定する際にこれらの値を使用する必要があります。

重要

クラスターのデプロイ時に、マルチファクター認証デバイスの使用中に生成した一時的なセッショントークンを使用して AWS に対する認証を行うことはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。

関連情報

- インストール前に Cloud Credential Operator (CCO) を手動モードに設定する手順については、AWS の IAM の手動作成 を参照してください。このモードは、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境で使用するか、または管理者レベルの認証情報シークレットをクラスターの **kube-system** プロジェクトに保存する選択をしない場合に使用します。

4.2.5. IAM ポリシーと AWS 認証

デフォルトでは、インストールプログラムは、ブートストラップ、コントロールプレーン、およびコンピュートインスタンスのインスタンスプロファイルを作成し、クラスターの動作に必要な権限を付与します。

ただし、独自の IAM ロールを作成して、インストールプロセスの一部として指定できます。クラスターをデプロイするため、またはインストール後にクラスターを管理するために、独自のロールを指定する必要がある場合があります。以下に例を示します。

- 組織のセキュリティーポリシーでは、より制限的なアクセス許可セットを使用してクラスターをインストールする必要があります。
- インストール後、クラスターは、追加サービスへのアクセスを必要とする Operator で設定されます。

独自の IAM ロールを指定する場合は、次の手順を実行できます。

- デフォルトのポリシーから始めて、必要に応じて調整します。詳細については、「IAM インスタンスプロファイルのデフォルトのアクセス許可」を参照してください。
- AWS IAM Access Analyzer (Identity and Access Management Access Analyzer) を使用して、クラスターのアクティビティーに基づくポリシーテンプレートを作成します。詳細は、「AWS IAM Analyzer を使用してポリシーテンプレートの作成」を参照してください。

4.2.5.1. IAM インスタンスプロファイルのデフォルトのアクセス許可

デフォルトでは、インストールプログラムは、ブートストラップ、コントロールプレーン、およびワーカーインスタンスの IAM インスタンスプロファイルを作成し、クラスターの動作に必要な権限を付与します。

次のリストでは、コントロールプレーンとコンピュートマシンのデフォルトのアクセス許可を指定します。

例4.14 コントロールプレーンインスタンスのプロファイル向けデフォルト IAM ロールのパーミッション

- **ec2:AttachVolume**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteVolume**
- **ec2:Describe***
- **ec2:DetachVolume**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyVolume**
- **ec2:RevokeSecurityGroupIngress**
- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**

- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerPolicy**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:DeleteListener**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeleteLoadBalancerListeners**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:Describe***
- **elasticloadbalancing:DetachLoadBalancerFromSubnets**
- **elasticloadbalancing:ModifyListener**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **kms:DescribeKey**

例4.15 コンピュートインスタンスプロファイル向けデフォルト IAM ロールのパーミッション

- **ec2:DescribeInstances**
- **ec2:DescribeRegions**

4.2.5.2. 既存の IAM ロールの指定

インストールプログラムがデフォルトのアクセス許可で IAM インスタンスプロファイルを作成できるようにする代わりに、**install-config.yaml** ファイルを使用して、コントロールプレーンとコンピュートインスタンスの既存の IAM ロールを指定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

1. コントロールプレーンマシンの既存のロールで **compute.platform.aws.iamRole** を更新します。

   コンピュートインスタンスの IAM ロールを含む install-config.yaml ファイルのサンプル

   ```
   compute:
   - hyperthreading: Enabled
     name: worker
     platform:
       aws:
         iamRole: ExampleRole
   ```

2. コンピュートマシンの既存のロールで **controlPlane.platform.aws.iamRole** を更新します。

   コントロールプレーンインスタンスの IAM ロールを含む install-config.yaml ファイルのサンプル

   ```
   controlPlane:
     hyperthreading: Enabled
     name: master
     platform:
       aws:
         iamRole: ExampleRole
   ```

3. ファイルを保存し、OpenShift Container Platform クラスターのインストール時に参照します。

関連情報

- クラスターのデプロイ を参照してください。

4.2.5.3. AWS IAM Analyzer を使用してポリシーテンプレートの作成

コントロールプレーンとコンピュートインスタンスプロファイルに必要な最小限のアクセス許可セットは、クラスターが日常の運用のためにどのように設定されているかによって異なります。

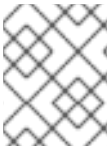
クラスターインスタンスに必要なアクセス許可を決定する 1 つの方法は、IAM Access Analyzer (AWS Identity and Access Management Access Analyzer) を使用してポリシーテンプレートを作成することです。

- ポリシーテンプレートには、クラスターが指定された期間に使用したアクセス許可が含まれています。
- その後、テンプレートを使用して、きめ細かい権限を持つポリシーを作成できます。

手順

全体的なプロセスは次のようになります。

1. CloudTrail が有効になっていることを確認します。CloudTrail は、ポリシーテンプレートの作成に必要な API 呼び出しを含め、AWS アカウントのすべてのアクションとイベントを記録します。詳細は、CloudTrail の操作 に関する AWS ドキュメントを参照してください。
2. コントロールプレーンインスタンスのインスタンスプロファイルとコンピューティングインスタンスのインスタンスプロファイルを作成します。PowerUserAccess などの寛容なポリシーを各ロールに割り当ててください。詳細は、インスタンスプロファイルロールの作成 に関する AWS ドキュメントを参照してください。
3. クラスターを開発環境にインストールし、必要に応じて設定します。クラスターが本番環境でホストするすべてのアプリケーションを必ずデプロイしてください。
4. クラスターを徹底的にテストします。クラスターをテストすると、必要なすべての API 呼び出しがログに記録されることが保証されます。
5. IAM Access Analyzer を使用して、各インスタンスプロファイルのポリシーテンプレートを作成します。詳細は、CloudTrail ログに基づいてポリシーを生成する ための AWS ドキュメントを参照してください。
6. きめ細かいポリシーを作成し、各インスタンスプロファイルに追加します。
7. 各インスタンスプロファイルから許容ポリシーを削除します。
8. 新しいポリシーで既存のインスタンスプロファイルを使用して実稼働クラスターをデプロイします。



注記

ポリシーに IAM 条件 を追加して、ポリシーをより制限し、組織のセキュリティー要件に準拠させることができます。

4.2.6. サポートされている AWS Marketplace リージョン

北米でオファーを購入したお客様は、AWS Marketplace イメージを使用して、OpenShift Container Platform クラスターをインストールすることができます。

このオファーはアメリカで購入する必要がありますが、サポートされるパーティションのいずれかにクラスターをデプロイできます。

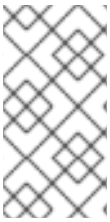
- 公開
- GovCloud

注記

AWS Marketplace イメージを使用した OpenShift Container Platform クラスターのデプロイは、AWS シークレットリージョンではサポートされていません。

4.2.7. サポートされている AWS リージョン

OpenShift Container Platform クラスターを以下のパブリックリージョンにデプロイできます。



注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

- **af-south-1** (Cape Town)
- **ap-east-1** (Hong Kong)
- **ap-northeast-1** (Tokyo)
- **ap-northeast-2** (Seoul)
- **ap-northeast-3** (Osaka)
- **ap-south-1** (Mumbai)
- **ap-southeast-1** (Singapore)
- **ap-southeast-2** (Sydney)
- **ca-central-1** (Central)
- **eu-central-1** (Frankfurt)
- **eu-north-1** (Stockholm)
- **eu-south-1** (Milan)
- **eu-west-1** (Ireland)
- **eu-west-2** (London)
- **eu-west-3** (Paris)
- **me-south-1** (Bahrain)
- **sa-east-1** (São Paulo)
- **us-east-1** (N. Virginia)
- **us-east-2** (Ohio)
- **us-west-1** (N. California)
- **us-west-2** (Oregon)

以下の AWS GovCloud リージョンがサポートされます。

- **us-gov-west-1**
- **us-gov-east-1**

AWS C2S シークレットリージョンがサポートされます。

- **us-iso-east-1**

4.2.8. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトオプションを使用した クラスターのクイックインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのクラウドのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール
  - CloudFormation テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

4.3. AWS の IAM の手動作成

クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境や、管理者がクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存する選択をしない場合に、クラスターのインストール前に Cloud Credential Operator (CCO) を手動モードにすることができます。

4.3.1. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。**credentialsMode** パラメーターの異なる値を **install-config.yaml** ファイルに設定し、組織のセキュリティー要件に応じて CCO を設定できます。

管理者レベルの認証情報シークレットをクラスターの **kube-system** プロジェクトに保存する選択をしない場合、OpenShift Container Platform をインストールする際に以下のいずれかのオプションを選択できます。

- **Amazon Web Services Security Token Service**
  CCO ユーティリティー (**ccoctl**) を使用して、Amazon Web Services Security Token Service (AWS STS) を設定できるようになりました。CCO ユーティリティーを使用して STS のクラスターを設定する場合、短期の権限に制限のあるセキュリティー認証情報を提供する IAM ロールをコンポーネントに割り当てます。

  注記

  このクレデンシャルストラテジーは、新しい OpenShift Container Platform クラスターでのみサポートされており、インストール中に設定する必要があります。この機能を使用するために、既存のクラスターが別のクレデンシャルストラテジーを使用するように再設定することはできません。

- **クラウド認証情報を手動で管理** します。
  CCO の **credentialsMode** パラメーターを **Manual** に設定し、クラウド認証情報を手動で管理できます。手動モードを使用すると、クラスターに管理者レベルの認証情報を保存する必要なく、各クラスターコンポーネントに必要なパーミッションのみを指定できます。お使いの環境でクラウドプロバイダーのパブリック IAM エンドポイントへの接続がない場合も、このモードを使用できます。ただし、各アップグレードについて、パーミッションを新規リリースイメージを使用して手動で調整する必要があります。また、それらを要求するすべてのコンポーネントについて認証情報を手動で指定する必要があります。

- **OpenShift Container Platform を mint モードでインストールした後に、管理者レベルの認証情報シークレットを削除** します。
credentialsMode パラメーターが **Mint** に設定された状態で CCO を使用している場合、OpenShift Container Platform のインストール後に管理者レベルの認証情報を削除したり、ローテーションしたりできます。Mint モードは、CCO のデフォルト設定です。このオプションには、インストール時に管理者レベルの認証情報が必要になります。管理者レベルの認証情報はインストール時に、付与された一部のパーミッションと共に他の認証情報を生成するために使用されます。元の認証情報シークレットはクラスターに永続的に保存されません。

注記

z-stream 以外のアップグレードの前に、認証情報のシークレットを管理者レベルの認証情報と共に元に戻す必要があります。認証情報が存在しない場合は、アップグレードがブロックされる可能性があります。

関連情報

- CCO ユーティリティー (**ccoctl**) を使用して AWS STS を使用するように CCO を設定する方法の詳細は、STS での手動モードの使用 を参照してください。
- OpenShift Container Platform のインストール後に管理者レベルの認証情報シークレットをローテーションするか、または削除する方法については、クラウドプロバイダーの認証情報のローテーションまたは削除 を参照してください。
- 利用可能なすべての CCO 認証情報モードとそれらのサポートされるプラットフォームの詳細については、Cloud Credential Operator について 参照してください。

4.3.2. IAM の手動作成

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、**install-config.yaml** ファイルを作成します。

```
$ openshift-install create install-config --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
```

```
- architecture: amd64
  hyperthreading: Enabled
...
```

1 この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

3. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ openshift-install create manifests --dir <installation_directory>
```

4. インストールプログラムが含まれるディレクトリーから、**openshift-install** バイナリーがビルドされる OpenShift Container Platform リリースイメージの詳細を取得します。

```
$ openshift-install version
```

出力例

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. このリリースイメージ内で、デプロイするクラウドをターゲットとする **CredentialsRequest** オブジェクトをすべて特定します。

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 --
credentials-requests --cloud=aws
```

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

サンプル CredentialsRequest オブジェクト

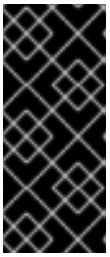
```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: cloud-credential-operator-iam-ro
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: cloud-credential-operator-iam-ro-creds
    namespace: openshift-cloud-credential-operator
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて

spec.secretRef に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

7. インストールプログラムが含まれるディレクトリーから、クラスターの作成に進みます。

```
$ openshift-install create cluster --dir <installation_directory>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。詳細は、クラウドプロバイダーのインストールコンテンツの手動でメンテナンスされる認証情報を使用したクラスターのアップグレードについてのセクションを参照してください。

4.3.3. 手動でメンテナンスされた認証情報を使用したクラスターのアップグレード

手動でメンテナンスされる認証情報をを含むクラスターの Cloud Credential Operator (CCO) の **upgradable** ステータスはデフォルトで **false** となります。

- 4.7 から 4.8 などのマイナーリリースの場合には、このステータスを使用することで、パーミッションを更新して **CloudCredential** リソースにアノテーションを付けてパーミッションが次のバージョンの要件に合わせて更新されていることを指定するまで、アップグレードができないようになります。このアノテーションは、**Upgradable** ステータスを **True** に変更します。

- 4.8.9 から 4.8.10 などの z-stream リリースの場合には、パーミッションは追加または変更されないため、アップグレードはブロックされません。

手動でメンテナンスされる認証情報でクラスターをアップグレードする前に、アップグレードするリリースイメージ用に認証情報を新規作成する必要があります。さらに、既存の認証情報に必要なパーミッションを確認し、これらのコンポーネントの新規リリースの新しいパーミッション要件に対応する必要があります。

手順

1. 新規リリースの **CredentialsRequest** カスタムリソースを抽出して検査します。クラウドプロバイダーのインストールコンテンツの IAM の手動作成についてのセクションでは、クラウドに必要な認証情報を取得し、使用する方法について説明します。

2. クラスターで手動でメンテナンスされる認証情報を更新します。

   - 新規リリースイメージによって追加される **CredentialsRequest** カスタムリソースの新規のシークレットを作成します。

   - シークレットに保存される既存の認証情報の **CredentialsRequest** カスタムリソースにパーミッション要件を変更した場合は、必要に応じてパーミッションを更新します。

3. 新規リリースですべてのシークレットが正しい場合は、クラスターをアップグレードする準備が整っていることを示します。

   a. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

   b. **CloudCredential** リソースを編集して、**metadata** フィールドに **upgradeable-to** アノテーションを追加します。

   ■

```
$ oc edit cloudcredential cluster
```

追加するテキスト

```
...
  metadata:
    annotations:
      cloudcredential.openshift.io/upgradeable-to: <version_number>
...
```

ここで、**<version_number>** は、アップグレードするバージョン (**x.y.z** 形式) に置き換えます。例: OpenShift Container Platform **4.8.2** (OpenShift Container Platform 4.8.2 の場合)

アノテーションを追加してから、upgradeable のステータスが変更されるまで、数分かかる場合があります。

4. CCO がアップグレードできることを確認します。

    a. Web コンソールの **Administrator** パースペクティブで、**Administration → Cluster Settings** に移動します。

    b. CCO ステータスの詳細を表示するには、**Cluster Operators** 一覧で **cloud-credential** をクリックします。

    c. **Conditions** セクションの **Upgradeable** ステータスが **False** の場合に、**upgradeable-to** アノテーションに間違いがないことを確認します。

Conditions セクションの **Upgradeable** ステータスが **True** の場合には、OpenShift Container Platform のアップグレードを開始できます。

4.3.4. mint モード

mint モードは、OpenShift Container Platform をサポートするプラットフォーム上の OpenShift Container Platform のデフォルトの Cloud Credential Operator (CCO) クレデンシャルモードです。このモードでは、CCO は提供される管理者レベルのクラウド認証情報を使用してクラスターを実行します。Mint モードは AWS と GCP でサポートされています。

mint モードでは、**admin** 認証情報は **kube-system** namespace に保存され、次に CCO によってクラスターの **CredentialsRequest** オブジェクトを処理し、特定のパーミッションでそれぞれのユーザーを作成するために使用されます。

mint モードには以下の利点があります。

- 各クラスターコンポーネントにはそれぞれが必要なパーミッションのみがあります。
- クラウド認証情報の自動の継続的な調整が行われます。これには、アップグレードに必要になる可能性のある追加の認証情報またはパーミッションが含まれます。

1 つの不利な点として、mint モードでは、**admin** 認証情報がクラスターの **kube-system** シークレットに保存される必要があります。

4.3.5. 管理者レベルの認証情報の削除またはローテーション機能を持つ mint モード

現時点で、このモードは AWS および GCP でのみサポートされます。

このモードでは、ユーザーは通常の mint モードと同様に管理者レベルの認証情報を使用して OpenShift Container Platform をインストールします。ただし、このプロセスはクラスターのインストール後の管理者レベルの認証情報シークレットを削除します。

管理者は、Cloud Credential Operator に読み取り専用の認証情報について独自の要求を行わせることができます。これにより、すべての **CredentialsRequest** オブジェクトに必要なパーミッションがあることの確認が可能になります。そのため、いずれかの変更が必要にならない限り、管理者レベルの認証情報は必要になりません。関連付けられた認証情報が削除された後に、必要な場合は、これは基礎となるクラウドで破棄するか、または非アクティブにできます。

注記

z-stream 以外のアップグレードの前に、認証情報のシークレットを管理者レベルの認証情報と共に元に戻す必要があります。認証情報が存在しない場合は、アップグレードがブロックされる可能性があります。

管理者レベルの認証情報はクラスターに永続的に保存されません。

これらの手順を実行するには、短い期間にクラスターでの管理者レベルの認証情報が必要になります。また、アップグレードごとに管理者レベルの認証情報を使用してシークレットを手動で再インストールする必要があります。

4.3.6. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトオプションを使用した クラスターの AWS へのクイックインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのクラウドのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール
  - CloudFormation テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

4.4. クラスターの AWS へのクイックインストール

OpenShift Container Platform バージョン 4.8 では、デフォルトの設定オプションを使用するクラスターを Amazon Web Services (AWS) にインストールできます。

4.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.4.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

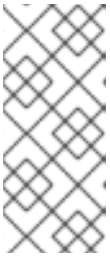
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

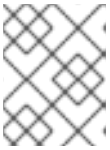
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

-

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.4.4. インストールプログラムの取得

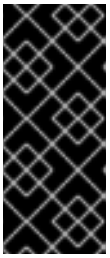
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.4.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

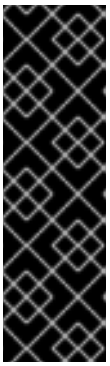
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

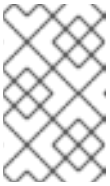


重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

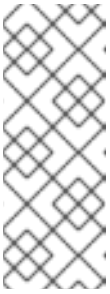


注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **aws** を選択します。

c. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。



注記

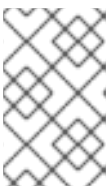
AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの ~/**.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

d. クラスターのデプロイ先とする AWS リージョンを選択します。

e. クラスターに設定した Route 53 サービスのベースドメインを選択します。

f. クラスターの記述名を入力します。

g. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。



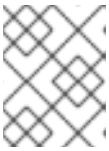
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

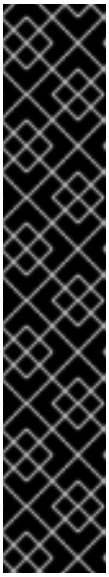
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

注記

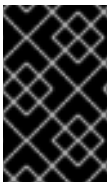
AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

関連情報

- AWS プロファイルおよび認証情報の設定についての詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

4.4.6. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.4.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.4.8. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

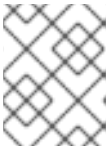
```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.4.9. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.4.10. 次のステップ

- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.5. カスタマイズによる AWS へのクラスターのインストール

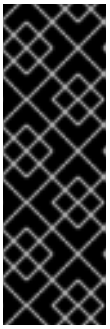
OpenShift Container Platform バージョン 4.8 では、インストールプログラムが Amazon Web Services (AWS) にプロビジョニングするインフラストラクチャーにカスタマイズされたクラスターをインストールすることができます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

注記

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

4.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.5.3. クラスターノードの SSH アクセス用のキーペアの生成

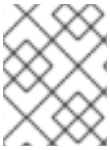
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

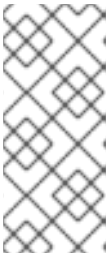
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

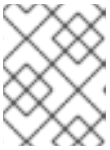
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

-

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

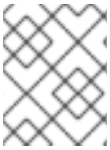
```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.5.4. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがワーカーノードのデプロイに使用する AMI ID が提供されます。



注記

AWS Marketplace イメージを使用した OpenShift Container Platform クラスターのデプロイは、シークレットリージョンではサポートされていません。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。
2. 使用する特定のリージョンの AMI ID を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、この値で **install-config.yaml** ファイルを更新する必要があります。

AWS Marketplace ワーカーノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) AWS Marketplace サブスクリプションの AMI ID。

(2) AMI ID は特定の AWS リージョンに関連付けられています。インストール設定ファイルを作成するときは、サブスクリプションの設定時に指定したものと同じ AWS リージョンを選択してください。

4.5.5. インストールプログラムの取得

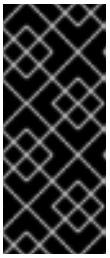
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.5.6. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

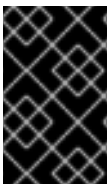
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

4.5.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.5.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.1 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.5.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.2 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.5.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.3 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

4.5.6.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表4.4 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.amiID	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。

パラメーター	説明	値
compute.platform.aws.iamRole	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードの OS ボリュームを特定の KMS キーで暗号化するために必要です。	有効な キー ID またはキー ARN。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。

パラメーター	説明	値
controlPlane.platform.aws.amiID	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
controlPlane.platform.aws.iamRole	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーを使用してコントロールプレーンノードの OS ボリュームを暗号化するために必要です。	有効な キー ID とキー ARN。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例:**m5.xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.amiID	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。

パラメーター	説明	値
platform.aws.hostedZone	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS サービスエンドポイント名。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名。
platform.aws.serviceEndpoints.url	AWS サービスエンドポイント URL。URL には **https** プロトコルを使用し、ホストは証明書を信頼する必要があります。	有効な AWS サービスエンドポイント URL。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。

パラメーター	説明	値
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

4.5.6.2. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

例4.16 マシンのインスタンスタイプ

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x

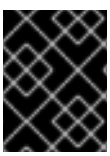
インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.5.6.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
```

```
  hyperthreading: Enabled 5
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 6
      type: m5.xlarge
  replicas: 3
compute: 7
- hyperthreading: Enabled 8
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 9
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
    amiID: ami-96c6f8f7 12
    serviceEndpoints: 13
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
fips: false 14
sshKey: ssh-ed25519 AAAA... 15
pullSecret: '{"auths": ...}' 16
```

1 10 11 16 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: このパラメーターを追加して、Cloud Credential Operator (CCO) に認証情報の機能を

3 7 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 9 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

12 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

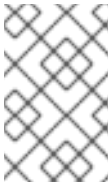
13 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

15 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

4.5.6.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に . を

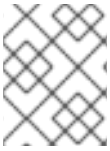
4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.5.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

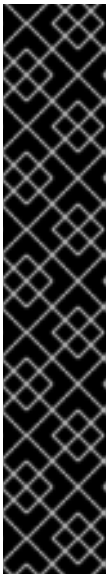
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



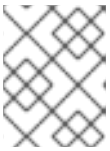
重要

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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。



注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

4.5.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.5.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.5.10. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

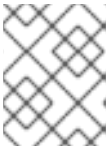
```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.5.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.5.12. 次のステップ

- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.6. ネットワークのカスタマイズによる AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、カスタマイズされたネットワーク設定オプションでクラスターを Amazon Web Services (AWS) にインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

4.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

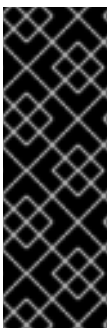
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.6.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

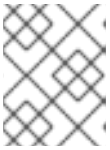
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.6.4. インストールプログラムの取得

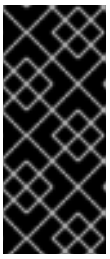
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

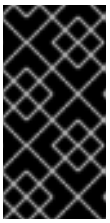
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

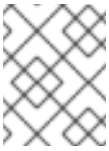
4.6.5. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

4.6.6. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

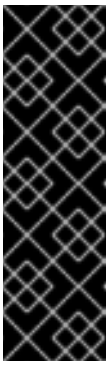
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

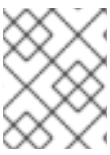
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

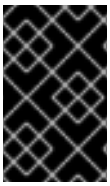
4.6.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.6.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.5 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.6.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.6 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

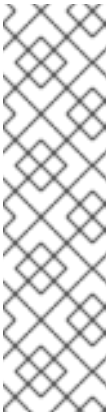
4.6.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.7 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

4.6.6.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表4.8 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.amiID	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
compute.platform.aws.iamRole	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。

パラメーター	説明	値
compute.platform.aws.rootVolume.type	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードの OS ボリュームを特定の KMS キーで暗号化するために必要です。	有効な キー ID またはキー ARN。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.amiID	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
controlPlane.platform.aws.iamRole	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。

パラメーター	説明	値
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーを使用してコントロールプレーンノードの OS ボリュームを暗号化するために必要です。	有効な キー ID とキー ARN。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例:**m5.xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.amiID	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
platform.aws.hostedZone	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)

パラメーター	説明	値
platform.aws.serviceEndpoints.name	AWS サービスエンドポイント名。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名。
platform.aws.serviceEndpoints.url	AWS サービスエンドポイント URL。URL には **https** プロトコルを使用し、ホストは証明書を信頼する必要があります。	有効な AWS サービスエンドポイント URL。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

4.6.6.2. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

> **例4.17 マシンのインスタンスタイプ**

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x

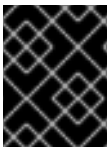
インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.6.6.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
```

```
  replicas: 3
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 12
    userTags:
      adminContact: jdoe
      costCenter: 7536
    amiID: ami-96c6f8f7 13
    serviceEndpoints: 14
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
pullSecret: '{"auths": ...}' 17
```

1 10 12 17 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: このパラメーターを追加して、Cloud Credential Operator (CCO) に認証情報の機能を動的に判別させようとするのではなく、CCO が指定されたモードを使用するように強制します。CCO モードの詳細は、**Platform Operatorのクラウド認証情報 Operator**を参照してください。

3 7 11 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 9 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

13 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

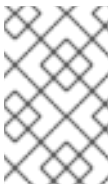
14 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

4.6.6.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

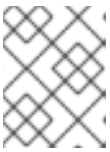
- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.6.7. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

4.6.7.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表4.9 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。
		
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		
		マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表4.10 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表4.11 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定
以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表4.12 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 暗号化を有効にするために空のオブジェクトを指定します。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表4.13 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表4.14 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

4.6.8. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

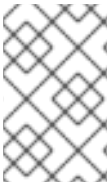
   OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

   ```
   apiVersion: operator.openshift.io/v1
   kind: Network
   metadata:
     name: cluster
   spec:
     defaultNetwork:
       openshiftSDNConfig:
         vxlanPort: 4800
   ```

   OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

   ```
   apiVersion: operator.openshift.io/v1
   kind: Network
   metadata:
     name: cluster
   spec:
     defaultNetwork:
       ovnKubernetesConfig:
         ipsecConfig: {}
   ```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。



注記

AWS で Network Load Balancer (NLB) を使用する方法についての詳細は、ネットワークロードバランサーを使用した AWS での ingress クラスタートラフィックの設定 を参照してください。

4.6.9. 新規 AWS クラスターでの Ingress コントローラーネットワークロードバランサーの設定

新規クラスターに AWS Network Load Balancer (NLB) がサポートする Ingress コントローラーを作成できます。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

新規クラスターの AWS NLB がサポートする Ingress コントローラーを作成します。

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

   ```
   $ ./openshift-install create manifests --dir <installation_directory> 1
   ```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-ingress-default-ingresscontroller.yaml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

3. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
      providerParameters:
        type: AWS
        aws:
          type: NLB
    type: LoadBalancerService
```

4. **cluster-ingress-default-ingresscontroller.yaml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

4.6.10. OVN-Kubernetes を使用したハイブリッドネットワークの設定

OVN-Kubernetes でハイブリッドネットワークを使用するようにクラスターを設定できます。これにより、異なるノードのネットワーク設定をサポートするハイブリッドクラスターが可能になります。たとえば、これはクラスター内の Linux ノードと Windows ノードの両方を実行するために必要です。

重要

クラスターのインストール時に、OVN-Kubernetes を使用してハイブリッドネットワークを設定する必要があります。インストールプロセス後に、ハイブリッドネットワークに切り替えることはできません。

前提条件

- **install-config.yaml** ファイルで **networking.networkType** パラメーターの **OVNKubernetes** を定義していること。詳細は、選択したクラウドプロバイダーでの OpenShift Container Platform ネットワークのカスタマイズの設定についてのインストールドキュメントを参照してください。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

ここでは、以下のようになります。

<installation_directory>

クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

ここでは、以下のようになります。

<installation_directory>

クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

3. **cluster-network-03-config.yml** ファイルをエディターで開き、以下の例のようにハイブリッドネットワークで OVN-Kubernetes を設定します。

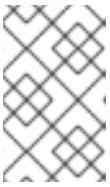
ハイブリッドネットワーク設定の指定

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
```

```
  hybridClusterNetwork: 1
  - cidr: 10.132.0.0/14
    hostPrefix: 23
  hybridOverlayVXLANPort: 9898 2
```

1 追加のオーバーレイネットワーク上のノードに使用される CIDR 設定を指定します。**hybridClusterNetwork** CIDR は **clusterNetwork** CIDR と重複できません。

2 追加のオーバーレイネットワークのカスタム VXLAN ポートを指定します。これは、vSphere にインストールされたクラスターで Windows ノードを実行するために必要であり、その他のクラウドプロバイダー用に設定することはできません。カスタムポートには、デフォルトの **4789** ポートを除くいずれかのオープンポートを使用できます。この要件についての詳細は、Microsoft ドキュメントの Pod-to-pod connectivity between hosts is broken を参照してください。



注記

Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019 は、カスタムの VXLAN ポートの選択をサポートしないため、カスタムの **hybridOverlayVXLANPort** 値を持つクラスターではサポートされません。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。
5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

注記

同じクラスターで Linux および Windows ノードを使用する方法についての詳細は、Understanding Windows container workloads を参照してください。

4.6.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

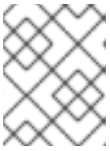
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

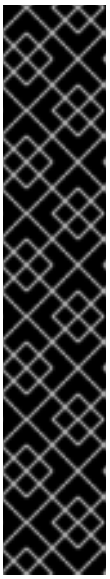
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

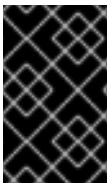
2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

4.6.12. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
   PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.6.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.6.14. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

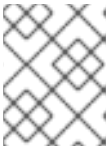
```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.6.15. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.6.16. 次のステップ

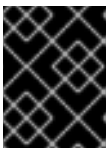
- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.7. ネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、既存の Amazon Virtual Private Cloud (VPC) でインストールリリースコンテンツの内部ミラーを作成して、ネットワークが制限された環境でクラスターを Amazon Web Services (AWS) にインストールできます。

4.7.1. 前提条件

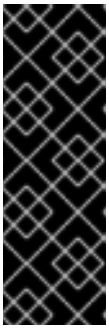
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得しています。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- AWS に既存の VPC が必要です。インストーラーでプロビジョニングされるインフラストラクチャーを使用してネットワークが制限された環境にインストールする場合は、インストーラーでプロビジョニングされる VPC を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - ミラーレジストリーが含まれる。
  - 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
- クラスターをホストするために AWS アカウントを設定 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.7.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネッ

トへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

4.7.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

4.7.3. カスタム VPC の使用について

OpenShift Container Platform 4.8 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

4.7.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。ホストゾーンは、**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドを使用して定義できます。

- パブリックアクセスでクラスターを使用する場合、クラスターが使用する各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを作成する必要があります。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

government リージョン

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

トップシークレットリージョン

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。
パブリックサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::SubnetNetworkAclAssociation**	VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。
インターネットゲートウェイ	• **AWS::EC2::InternetGateway** • **AWS::EC2::VPCGatewayAttachment** • **AWS::EC2::RouteTable** • **AWS::EC2::Route** • **AWS::EC2::SubnetRouteTableAssociation** • **AWS::EC2::NatGateway** • **AWS::EC2::EIP**	VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。
ネットワークアクセス制御	• **AWS::EC2::NetworkAcl** • **AWS::EC2::NetworkAclEntry**	VPC が以下のポートにアクセスできるようにする必要があります。 **ポート** / **理由** **80** / インバウンド HTTP トラフィック

コンポーネント	AWS タイプ	説明	
		443	インバウンド HTTPS トラフィック
		22	インバウンド SSH トラフィック
		1024 - 65535	インバウンド一時 (ephemeral) トラフィック
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

4.7.3.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

4.7.3.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

4.7.3.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

4.7.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.7.5. クラスターノードの SSH アクセス用のキーペアの生成

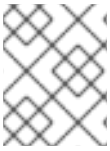
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

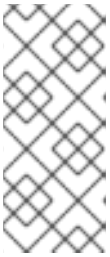
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.7.6. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

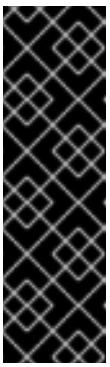
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。
      iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. クラスターをインストールする VPC のサブネットを定義します。

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

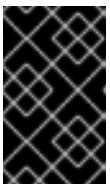
d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

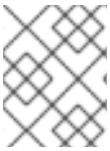


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

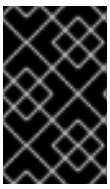
4.7.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.7.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.15 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.7.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.16 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

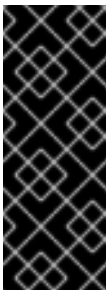
networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

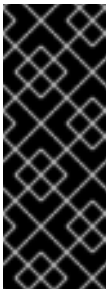
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.7.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.17 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

4.7.6.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表4.18 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.amiID	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
compute.platform.aws.iamRole	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。

パラメーター	説明	値
compute.platform.aws.rootVolume.type	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードの OS ボリュームを特定の KMS キーで暗号化するために必要です。	有効な キー ID またはキー ARN。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.amiID	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
controlPlane.platform.aws.iamRole	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。

パラメーター	説明	値
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーを使用してコントロールプレーンノードの OS ボリュームを暗号化するために必要です。	有効な キー ID とキー ARN。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例:**m5.xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.amiID	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
platform.aws.hostedZone	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)

パラメーター	説明	値
platform.aws.serviceEndpoints.name	AWS サービスエンドポイント名。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名。
platform.aws.serviceEndpoints.url	AWS サービスエンドポイント URL。URL には **https** プロトコルを使用し、ホストは証明書を信頼する必要があります。	有効な AWS サービスエンドポイント URL。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

4.7.6.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (11)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (12)
    - subnet-1
```

```
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 13
    serviceEndpoints: 14
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 15
fips: false 16
sshKey: ssh-ed25519 AAAA... 17
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 18
additionalTrustBundle: | 19
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 20
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 10 11 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: このパラメーターを追加して、Cloud Credential Operator (CCO) に認証情報の機能を動的に判別させようとするのではなく、CCO が指定されたモードを使用するように強制します。CCO モードの詳細は、**Platform Operatorのクラウド認証情報 Operator**を参照してください。

3 7 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 9 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

12 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

13 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

14 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

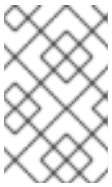
15 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

16 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

17 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

19 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

20 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

4.7.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対す

る呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーの

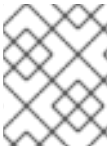
アイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

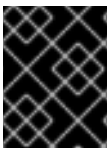


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.7.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

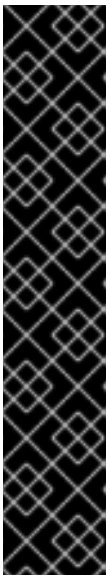
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

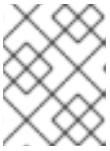
- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。



注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

4.7.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.7.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.7.10. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

4.7.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.7.12. 次のステップ

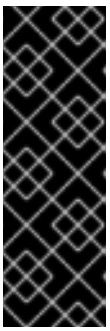
- インストールを検証 します。
- クラスターをカスタマイズします。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.8. クラスターの AWS の既存 VPC へのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) にインストールできます。インストールプログラムは、さらなるカスタマイズが可能な必要なインフラストラクチャーの残りをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

4.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.8.2. カスタム VPC の使用について

OpenShift Container Platform 4.8 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる

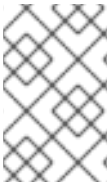
運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

4.8.2.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- クラスターが使用するアベイラビリティーゾーンごとにパブリックサブネットとプライベートサブネットを作成します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。このタイプの設定の例は、AWS ドキュメントの パブリックサブネットとプライベートサブネット (NAT) を使用した VPC を参照してください。

各サブネット ID を記録します。インストールを完了するには、 **install-config.yaml** ファイルの **プラットフォーム** セクションにこれらの値を入力する必要があります。AWS ドキュメントの サブネット ID の検索 を参照してください。

- VPC の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれている必要があります。サブネット CIDR ブロックは、指定したマシン CIDR に属している必要があります。
- VPC には、パブリックインターネットゲートウェイが接続されている必要があります。アベイラビリティーゾーンごとに以下が必要です。
  - パブリックサブネットには、インターネットゲートウェイへのルートが必要です。
  - パブリックサブネットには、EIP アドレスが割り当てられた NAT ゲートウェイが必要です。
  - プライベートサブネットには、パブリックサブネットの NAT ゲートウェイへのルートが必要です。
- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。
- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。ホストゾーンは、**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドを使用して定義できます。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

government リージョン

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

トップシークレットリージョン

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。</td></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明	
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

4.8.2.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

4.8.2.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、

S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

4.8.2.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

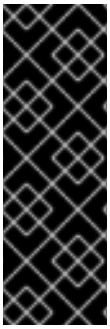
- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

4.8.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.8.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

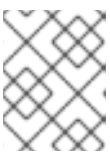
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.8.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

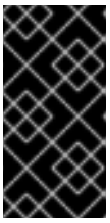
1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。

2. インフラストラクチャープロバイダーを選択します。

3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.8.6. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

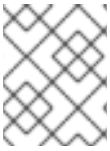
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

4.8.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.8.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.19 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.8.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.20 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.8.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.21 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

4.8.6.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表4.22 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.amiID	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。

パラメーター	説明	値
compute.platform.aws.iamRole	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードの OS ボリュームを特定の KMS キーで暗号化するために必要です。	有効な キー ID またはキー ARN。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。

パラメーター	説明	値
controlPlane.platform.aws.amiID	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
controlPlane.platform.aws.iamRole	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーを使用してコントロールプレーンノードの OS ボリュームを暗号化するために必要です。	有効な キー ID とキー ARN。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例:**m5.xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.amiID	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。

パラメーター	説明	値
platform.aws.hostedZone	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)
platform.aws.serviceEndpoints.name	AWS サービスエンドポイント名。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名。
platform.aws.serviceEndpoints.url	AWS サービスエンドポイント URL。URL には **https** プロトコルを使用し、ホストは証明書を信頼する必要があります。	有効な AWS サービスエンドポイント URL。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。

パラメーター	説明	値
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

4.8.6.2. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

例4.18 マシンのインスタンスタイプ

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x

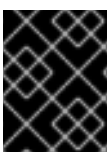
インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.8.6.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
```

```
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (11)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (12)
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 (13)
    serviceEndpoints: (14)
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV (15)
```

```
fips: false (16)
sshKey: ssh-ed25519 AAAA... (17)
pullSecret: '{"auths": ...}' (18)
```

(1) (10) (11) (18) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) オプション: このパラメーターを追加して、Cloud Credential Operator (CCO) に認証情報の機能を動的に判別させようとするのではなく、CCO が指定されたモードを使用するように強制します。CCO モードの詳細は、**Platform Operatorのクラウド認証情報 Operatorを参照してください。**

(3) (7) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(4) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(5) (8) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

(6) (9) 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

(12) 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

(13) クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

(14) AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

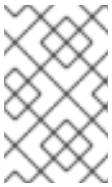
(15) 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

(16) FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

17 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

4.8.6.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

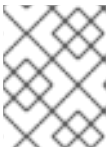
- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

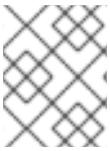


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.8.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

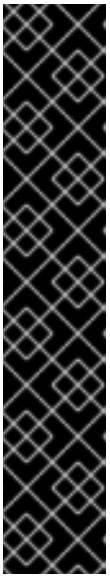
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



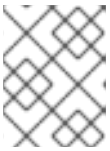
重要

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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。



注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

4.8.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.8.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.8.10. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.8.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.8.12. 次のステップ

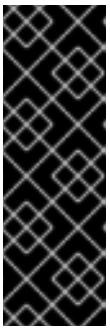
- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.9. プライベートクラスターの AWS へのインストール

OpenShift Container Platform バージョン 4.8 では、プライベートクラスターを Amazon Web Services (AWS) の既存の VPC にインストールできます。インストールプログラムは、さらなるカスタマイズが可能な必要なインフラストラクチャーの残りをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

4.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.9.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

<<<<<<< HEAD デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

4.9.2.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

4.9.2.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

4.9.3. カスタム VPC の使用について

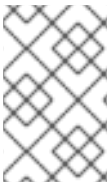
OpenShift Container Platform 4.8 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

4.9.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。
- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。

インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。

- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。ホストゾーンは、**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドを使用して定義できます。

- パブリックアクセスでクラスターを使用する場合、クラスターが使用する各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを作成する必要があります。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

government リージョン

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

トップシークレットリージョン

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

4.9.3.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

4.9.3.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

4.9.3.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

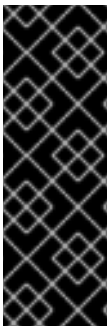
- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

4.9.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.9.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

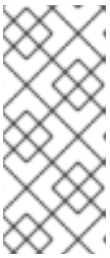
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

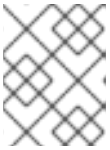
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

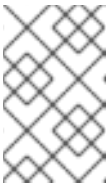
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.9.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.9.7. インストール設定ファイルの手動作成

内部ネットワークからのみアクセスでき、インターネット上に表示されないプライベート OpenShift Container Platform クラスターのインストールの場合、インストール設定ファイルを手動で生成する必要があります。

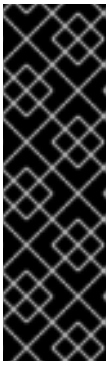
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

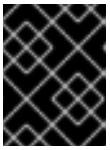
注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

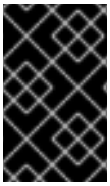
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

4.9.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.9.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.23 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

4.9.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.24 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.9.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.25 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

4.9.7.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表4.26 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.amiID	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
compute.platform.aws.iamRole	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。

パラメーター	説明	値
compute.platform.aws.rootVolume.type	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードの OS ボリュームを特定の KMS キーで暗号化するために必要です。	有効な キー ID またはキー ARN。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.amiID	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
controlPlane.platform.aws.iamRole	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。

パラメーター	説明	値
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーを使用してコントロールプレーンノードの OS ボリュームを暗号化するために必要です。	有効な キー ID とキー ARN。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例:**m5.xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.amiID	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
platform.aws.hostedZone	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)

パラメーター	説明	値
platform.aws.serviceEndpoints.name	AWS サービスエンドポイント名。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名。
platform.aws.serviceEndpoints.url	AWS サービスエンドポイント URL。URL には **https** プロトコルを使用し、ホストは証明書を信頼する必要があります。	有効な AWS サービスエンドポイント URL。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

4.9.7.2. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

> **例4.19 マシンのインスタンスタイプ**

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x

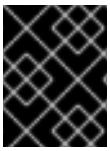
インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.9.7.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-west-2c
```

```
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 12
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 13
    serviceEndpoints: 14
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV 15
fips: false 16
sshKey: ssh-ed25519 AAAA... 17
publish: Internal 18
pullSecret: '{"auths": ...}' 19
```

1 10 11 19 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 オプション: このパラメーターを追加して、Cloud Credential Operator (CCO) に認証情報の機能を動的に判別させようとするのではなく、CCO が指定されたモードを使用するように強制します。CCO モードの詳細は、**Platform Operatorのクラウド認証情報 Operator**を参照してください。

3 7 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

4 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

5 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 9 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

12 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

13 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

14 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

15 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

16 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

17 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

4.9.7.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

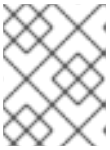
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存しま

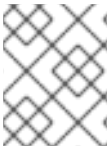


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

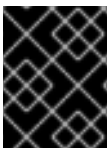


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.9.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、以下を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

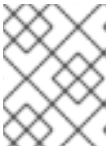
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

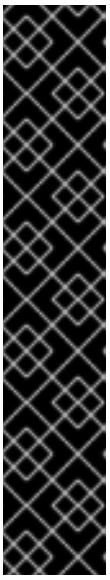
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

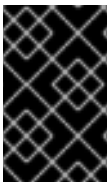
- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

4.9.9. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.9.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.9.11. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

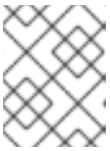
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

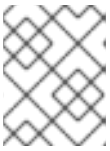


注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.9.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.9.13. 次のステップ

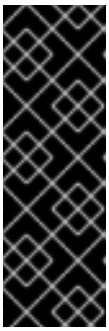
- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.10. AWS の GOVERNMENT またはシークレットリージョンへのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを Amazon Web Services (AWS) の government またはシークレットリージョンにインストールできます。リージョンを設定するには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

4.10.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.10.2. AWS government およびシークレットリージョン

OpenShift Container Platform はクラスターの AWS GovCloud (US) リージョンおよび AWS Commercial Cloud Services (C2S) トップシークレットリージョン へのデプロイをサポートします。これらのリージョンは、機密ワークロードをクラウドで実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されています。

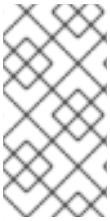
これらのリージョンには、選択する Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Images (AMI) が公開されていないため、そのリージョンに属するカスタム AMI をアップロードする必要があります。

以下の AWS GovCloud パーティションがサポートされます。

- **us-gov-west-1**
- **us-gov-east-1**

以下の AWS トップシークレットリージョンのパーティションがサポートされます。

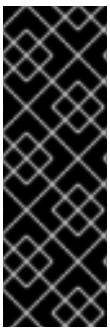
- **us-iso-east-1**



注記

AWS Top Secret Region でサポートされる最大 MTU は、AWS コマーシャルと同じではありません。インストール中の MTU の設定の詳細については、**ネットワークをカスタマイズした AWS へのクラスターのインストール** の **クラスターネットワークオペレーターの設定オブジェクト** セクションを参照してください。

AWS government またはシークレットリージョン、およびそれに伴うカスタム AMI は、RHCOS AMI がそれらのリージョンについて Red Hat によって提供されないため、**install-config.yaml** ファイルで手動で設定される必要があります。

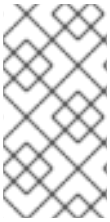


重要

C2S シークレットリージョンにデプロイする場合、AWS API にはカスタム CA 信頼バンドルが必要になるため、**install-config.yaml** ファイルの **additionalTrustBundle** フィールドでカスタム CA 証明書を定義する必要もあります。インストールプログラムが AWS API にアクセスできるようにするには、インストールプログラムを実行するマシンに CA 証明書を定義する必要もあります。CA バンドルをマシンの信頼ストアに追加し、**AWS_CA_BUNDLE** 環境変数を使用するか、または AWS 設定ファイルの **ca_bundle** フィールドに CA バンドルを定義する必要があります。

4.10.3. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。



注記

パブリックゾーンは、AWS GovCloud の Route 53 またはトップシークレットリージョンではサポートされません。そのため、クラスターは AWS government リージョンまたはシークレットリージョンにデプロイされる場合、プライベートである必要があります。

<<<<<<< HEAD デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

4.10.3.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

4.10.3.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

4.10.4. カスタム VPC の使用について

OpenShift Container Platform 4.8 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

4.10.4.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。AWS VPC の作成と管理の詳細は、AWS ドキュメントの Amazon VPC コンソールウィザードの設定 と VPC とサブネットの操作 を参照してください。

インストールプログラムには、以下の機能はありません。

- 使用するクラスターのネットワーク範囲を細分化する。

- サブネットのルートテーブルを設定する。
- DHCP などの VPC オプションを設定する。

クラスターをインストールする前に、以下のタスクを完了する必要があります。AWS VPC でのネットワーキングの設定の詳細は、 VPC ネットワーキングコンポーネント と VPC のルートテーブル を参照してください。

VPC は以下の特性を満たす必要があります。

- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。
  インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。
- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。
  独自の Route 53 ホストプライベートゾーンを使用する場合、クラスターのインストール前に既存のホストゾーンを VPC に関連付ける必要があります。ホストゾーンは、**install-config.yaml** ファイルの **platform.aws.hostedZone** フィールドを使用して定義できます。
- パブリックアクセスでクラスターを使用する場合、クラスターが使用する各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを作成する必要があります。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

government リージョン

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

トップシークレットリージョン

- **ec2.<region>.c2s.ic.gov**
- **elasticloadbalancing.<region>.c2s.ic.gov**
- **s3.<region>.c2s.ic.gov**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。</td></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明	
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

4.10.4.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

4.10.4.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、

S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

4.10.4.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

4.10.5. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.10.6. クラスターノードの SSH アクセス用のキーペアの生成

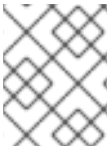
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

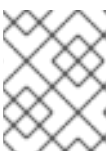
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

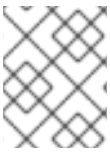
```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.10.7. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがワーカーノードのデプロイに使用する AMI ID が提供されます。



注記

AWS Marketplace イメージを使用した OpenShift Container Platform クラスターのデプロイは、シークレットリージョンではサポートされていません。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。

2. 使用する特定のリージョンの AMI ID を記録します。インストールプロセスの一環として、クラスターをデプロイする前に、この値で **install-config.yaml** ファイルを更新する必要があります。

AWS Marketplace ワーカーノードを含む install-config.yaml ファイルのサンプル

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-gov-west-1 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) AWS Marketplace サブスクリプションの AMI ID。

(2) AMI ID は特定の AWS リージョンに関連付けられています。インストール設定ファイルを作成するときは、サブスクリプションの設定時に指定したものと同じ AWS リージョンを選択してください。

4.10.8. インストールプログラムの取得

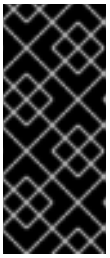
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.10.9. インストール設定ファイルの手動作成

OpenShift Container Platform を Amazon Web Services (AWS) でカスタム Red Hat Enterprise Linux CoreOS (RHCOS) AMI を必要とするリージョンにインストールする場合、インストール設定ファイルを手動で生成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

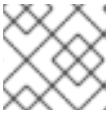
1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

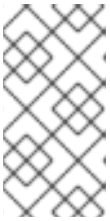
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

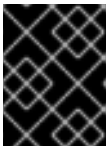
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

4.10.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.10.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.27 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

4.10.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.28 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.10.9.1.3. オプションの設定パラメーター

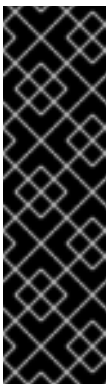
オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.29 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

4.10.9.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表4.30 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.amiID	クラスターのコンピュートマシンの起動に使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
compute.platform.aws.iamRole	コンピュートマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。

パラメーター	説明	値
compute.platform.aws.rootVolume.type	root ボリュームのタイプです。	有効な AWS EBS ボリュームタイプ (例: **io1**)。
compute.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、ワーカーノードの OS ボリュームを特定の KMS キーで暗号化するために必要です。	有効な キー ID またはキー ARN。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **m4.2xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.amiID	クラスターのコントロールプレーンマシンを起動するために使用される AWS AMI。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
controlPlane.platform.aws.iamRole	コントロールプレーンマシンプールインスタンスのプロファイルに適用される既存の AWS IAM ロール。これらのフィールドを使用して命名スキームに一致させ、IAM ロール用に事前に定義されたパーミッション境界を含めることができます。定義されていない場合は、インストールプログラムは新規の IAM ロールを作成します。	有効な AWS IAM ロール名。

パラメーター	説明	値
controlPlane.platform.aws.rootVolume.kmsKeyARN	KMS キーの Amazon リソース名 (キー ARN)。これは、特定の KMS キーを使用してコントロールプレーンノードの OS ボリュームを暗号化するために必要です。	有効な キー ID とキー ARN。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例:**m5.xlarge**)。**マシンのインスタンスタイプ** の表を参照してください。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.amiID	クラスターのすべてのマシンを起動するために使用される AWS AMI。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。これは、カスタム RHCOS AMI を必要とするリージョンに必要です。	設定した AWS リージョンに属するパブリッシュ済みまたはカスタムの RHCOS AMI。
platform.aws.hostedZone	クラスターの既存の Route 53 プライベートホストゾーン。独自の VPC を指定する場合も、既存のホストゾーンのみを使用できます。ホストゾーンは、インストール前にユーザーによって提供される VPC に関連付けられている必要があります。また、ホストゾーンのドメインはクラスタードメインまたはクラスタードメインの親である必要があります。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。	文字列 (例: **Z3URY6TWQ91KVV**)

パラメーター	説明	値
platform.aws.serviceEndpoints.name	AWS サービスエンドポイント名。カスタムエンドポイントは、FIPS などの AWS の代替エンドポイントを使用しなければならない場合にのみ必要です。カスタム API エンドポイントは、EC2、S3、IAM、Elastic Load Balancing、Tagging、Route 53、および STS AWS サービスに指定できます。	有効な AWS サービスエンドポイント 名。
platform.aws.serviceEndpoints.url	AWS サービスエンドポイント URL。URL には **https** プロトコルを使用し、ホストは証明書を信頼する必要があります。	有効な AWS サービスエンドポイント URL。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

4.10.9.2. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

> **例4.20 マシンのインスタンスタイプ**

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.10.9.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
credentialsMode: Mint (2)
controlPlane: (3) (4)
  hyperthreading: Enabled (5)
  name: master
  platform:
    aws:
      zones:
      - us-gov-west-1a
      - us-gov-west-1b
      rootVolume:
        iops: 4000
        size: 500
        type: io1 (6)
      type: m5.xlarge
  replicas: 3
compute: (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (9)
      type: c5.4xlarge
      zones:
      - us-gov-west-1c
```

```
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-gov-west-1
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (11)
    - subnet-1
    - subnet-2
    - subnet-3
    amiID: ami-96c6f8f7 (12)
    serviceEndpoints: (13)
      - name: ec2
        url: https://vpce-id.ec2.us-west-2.vpce.amazonaws.com
    hostedZone: Z3URY6TWQ91KVV (14)
fips: false (15)
sshKey: ssh-ed25519 AAAA... (16)
publish: Internal (17)
pullSecret: '{"auths": ...}' (18)
additionalTrustBundle: | (19)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
```

(1) (10) (18) 必須。

(2) オプション: このパラメーターを追加して、Cloud Credential Operator (CCO) に認証情報の機能を動的に判別させようとするのではなく、CCO が指定されたモードを使用するように強制します。CCO モードの詳細は、**Platform Operatorのクラウド認証情報 Operator**を参照してください。

(3) (7) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(4) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(5) (8) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

6 9 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

11 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

12 クラスターのマシンを起動するために使用される AMI の ID。これが設定されている場合、AMI はクラスターと同じリージョンに属する必要があります。

13 AWS サービスエンドポイント。未確認の AWS リージョンにインストールする場合は、カスタムエンドポイントが必要です。エンドポイントの URL は **https** プロトコルを使用しなければならず、ホストは証明書を信頼する必要があります。

14 既存の Route 53 プライベートホストゾーンの ID。既存のホストゾーンを指定するには、独自の VPC を指定する必要があり、ホストゾーンはすでにクラスターをインストールする前に VPC に関連付けられます。定義されていない場合は、インストールプログラムは新規のホストゾーンを作成します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

19 カスタム CA 証明書。AWS API にはカスタム CA 信頼バンドルが必要になるため、これは AWS C2S トップシークレットリージョンにデプロイする際に必要になります。

4.10.9.4. 公開済み RHCOS AMI のない AWS リージョン

Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) または AWS software development kit (SDK) のネイティブサポートなしに、OpenShift Container Platform クラスターを Amazon Web Services (AWS) リージョンにデプロイできます。パブリッシュ済みの AMI が AWS リージョンで利用できない場合は、クラスターをインストールする前にカスタム AMI をアップロードできます。これは、クラスターを AWS government またはシークレットリージョンにデプロイする場合に必要です。AWS government およびシークレットリージョンは AWS SDK によってサポートされます。

AWS SDK によってサポートされないリージョンにデプロイしている場合で、カスタム AMI を指定しない場合、インストールプログラムは **us-east-1** AMI をユーザーアカウントに自動的にコピーします。次にインストールプログラムは、デフォルトまたはユーザー指定の Key Management Service (KMS) キーを使用して、暗号化された EBS ボリュームでコントロールプレーンマシンを作成します。これにより、AMI は、パブリッシュ済みの RHCOS AMI と同じプロセスワークフローを実施することができます。

RHCOS AMI のネイティブサポートのないリージョンはパブリッシュされないため、クラスターの作成時にターミナルから選択することはできません。ただし、**install-config.yaml** ファイルでカスタム AMI を設定して、このリージョンにインストールすることができます。

4.10.9.5. AWS でのカスタム RHCOS AMI のアップロード

カスタム Amazon Web Services (AWS) リージョンにデプロイする場合、そのリージョンに属するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) をアップロードする必要があります。

前提条件

- AWS アカウントを設定している。
- 必要な IAM サービス出力 で、Amazon S3 バケットを作成している。
- RHCOS VMDK ファイルを Amazon S3 にアップロードしている。RHCOS VMDK ファイルは、インストールする OpenShift Container Platform のバージョンと同じか、またはそれ以下のバージョンである必要があります。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer を参照してください。

手順

1. AWS プロファイルを環境変数としてエクスポートします。

```
$ export AWS_PROFILE=<aws_profile> 1
```

1 **govcloud** などの AWS 認証情報を保持する AWS プロファイル名。

2. カスタム AMI に関連付けるリージョンを環境変数としてエクスポートします。

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

1 **us-gov-east-1** などの AWS リージョン。

3. 環境変数として Amazon S3 にアップロードした RHCOS のバージョンをエクスポートします。

```
$ export RHCOS_VERSION=<version> 1
```

1 **4.8.0** などの RHCOS VMDK バージョン。

4. Amazon S3 バケット名を環境変数としてエクスポートします。

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **containers.json** ファイルを作成し、RHCOS VMDK ファイルを定義します。

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. RHCOS ディスクを Amazon EBS スナップショットとしてインポートします。

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64** などの RHCOS ディスクがインポートされていることの説明。

2 RHCOS ディスクを説明する JSON ファイルへのファイルパス。JSON ファイルには、Amazon S3 バケット名とキーが含まれている必要があります。

7. イメージインポートのステータスを確認します。

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

出力例

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
```

```
                "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
            }
        }
    }
  ]
}
```

SnapshotId をコピーして、イメージを登録します。

8. RHCOS スナップショットからカスタム RHCOS AMI を作成します。

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
  --root-device-name '/dev/xvda' \
  --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. **x86_64**、**s390x**、または **ppc64le** などの RHCOS VMDK アーキテクチャータイプ。
2. インポートされたスナップショットの **Description**。
3. RHCOS AMI の名前。
4. インポートされたスナップショットからの **SnapshotID**。

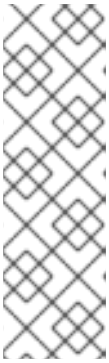
これらの API の詳細は、AWS ドキュメントの Importing a Disk as a Snapshot Using VM Import/Export および Creating a Linux AMI from a snapshot を参照してください。

4.10.9.6. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

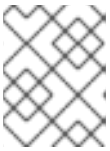
- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

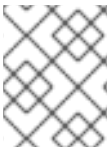


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

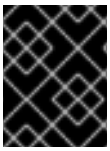


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.10.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

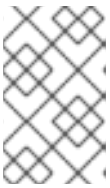
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

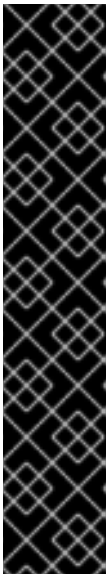
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

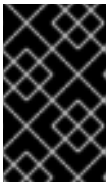
注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

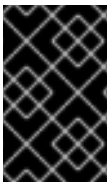
2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

注記

AdministratorAccess ポリシーが提供する昇格したパーミッションはインストール時にのみ必要です。

4.10.11. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.10.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.10.13. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.10.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

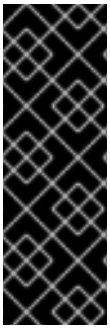
4.10.15. 次のステップ

- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.11. CLOUDFORMATION テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを独自に提供するインフラストラクチャーを使用する Amazon Web Services (AWS) にインストールできます。

このインフラストラクチャーを作成する 1 つの方法として、提供される CloudFormation テンプレートを使用できます。テンプレートを変更してインフラストラクチャーをカスタマイズしたり、それらに含まれる情報を使用し、所属する会社のポリシーに基づいて AWS オブジェクトを作成したりできます。



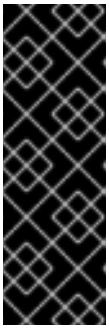
重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の CloudFormation テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

4.11.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするために AWS アカウントを設定 しています。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) を参照してください。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.11.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.11.3. 必要な AWS インフラストラクチャーコンポーネント

OpenShift Container Platform を Amazon Web Services (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、マシンとサポートするインフラストラクチャーの両方を手動で作成する必要があります。

各種プラットフォームの統合テストの詳細については、OpenShift Container Platform 4.x のテスト済みインテグレーション のページを参照してください。

提供される CloudFormation テンプレートを使用すると、以下のコンポーネントを表す AWS リソースのスタックを作成できます。

- AWS Virtual Private Cloud (VPC)
- ネットワークおよび負荷分散コンポーネント
- セキュリティーグループおよびロール
- OpenShift Container Platform ブートストラップノード
- OpenShift Container Platform コントロールプレーンノード
- OpenShift Container Platform コンピュートノード

または、コンポーネントを手動で作成するか、またはクラスターの要件を満たす既存のインフラストラクチャーを再利用できます。コンポーネントの相互関係についての詳細は、CloudFormation テンプレートを参照してください。

4.11.3.1. 他のインフラストラクチャーコンポーネント

- 1 つの VPC
- DNS エントリー
- ロードバランサー (classic または network) およびリスナー
- パブリックおよびプライベート Route 53 ゾーン
- セキュリティーグループ
- IAM ロール
- S3 バケット

非接続環境で作業している場合、またはプロキシーを使用する場合には、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これらのエンドポイントに到達するには、VPC エンドポイントを作成してクラスターが使用するサブネットに割り当てる必要があります。以

下のエンドポイントを作成します。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。</td></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明	
		443	インバウンド HTTPS トラフィック
		22	インバウンド SSH トラフィック
		1024 - 65535	インバウンド一時 (ephemeral) トラフィック
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

必要な DNS および負荷分散コンポーネント

DNS およびロードバランサー設定では、パブリックホストゾーンを使用する必要があり、クラスターのインフラストラクチャーをプロビジョニングする場合にインストールプログラムが使用するものと同様のプライベートホストゾーンを使用できます。ロードバランサーに解決する DNS エントリーを作成する必要があります。**api.<cluster_name>.<domain>** のエントリーは外部ロードバランサーを参照し、**api-int.<cluster_name>.<domain>** のエントリーは内部ロードバランサーを参照する必要があります。

またクラスターには、Kubernetes API とその拡張に必要なポート 6443、および新規マシンの Ignition 設定ファイルに必要なポート 22623 のロードバランサーおよびリスナーが必要です。ターゲットはコントロールプレーンノード (別名マスターノード) になります。ポート 6443 はクラスター外のクライアントとクラスター内のノードからもアクセスできる必要があります。ポート 22623 はクラスター内のノードからアクセスできる必要があります。

コンポーネント	AWS タイプ	説明
DNS	**AWS::Route53::HostedZone**	内部 DNS のホストゾーン。

コンポーネント	AWS タイプ	説明
etcd レコードセット	**AWS::Route53::RecordSet**	コントロールプレーンマシンの etcd の登録レコード。
パブリックロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	パブリックサブネットのロードバランサー。
外部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	外部 API サーバーのエイリアスレコード。
外部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	外部ロードバランサー用のポート 6443 のリスナー。
外部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	外部ロードバランサーのターゲットグループ。
プライベートロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	プライベートサブネットのロードバランサー。
内部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	内部 API サーバーのエイリアスレコード。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサー用のポート 22623 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサーのポート 6443 のリスナー。

コンポーネント	AWS タイプ	説明
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。

セキュリティーグループ

コントロールプレーンおよびワーカーマシンには、以下のポートへのアクセスが必要です。

グループ	タイプ	IP プロトコル	ポート範囲
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
		tcp	**6443**
		tcp	**22623**
WorkerSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
BootstrapSecurityGroup	**AWS::EC2::SecurityGroup**	**tcp**	**22**
		tcp	**19531**

コントロールプレーンの Ingress

コントロールプレーンマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressEtcd	etcd	**tcp**	**2379- 2380**
MasterIngressVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressInternal	内部クラスター通信および Kubernetes プロキシーメトリクス	**tcp**	**9000 - 9999**

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressWorkerInternal	内部クラスター通信	**tcp**	**9000** - **9999**
MasterIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250** - **10259**
MasterIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250** - **10259**
MasterIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000** - **32767**
MasterIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000** - **32767**
MasterIngressGeneve	Geneve パケット	**udp**	**6081**
MasterIngressWorkerGeneve	Geneve パケット	**udp**	**6081**
MasterIngressIpsecIke	IPsec IKE パケット	**udp**	**500**
MasterIngressWorkerIpsecIke	IPsec IKE パケット	**udp**	**500**
MasterIngressIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
MasterIngressWorkerIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
MasterIngressIpsecEsp	IPsec ESP パケット	**50**	**All**
MasterIngressWorkerIpsecEsp	IPsec ESP パケット	**50**	**All**

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressInternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
MasterIngressWorkerInternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
MasterIngressIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**
MasterIngressWorkerIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**

ワーカーの Ingress

ワーカーマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngress WorkerIngress Services	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngress Geneve	Geneve パケット	**udp**	**6081**
WorkerIngress MasterGeneve	Geneve パケット	**udp**	**6081**
WorkerIngress IpsecIke	IPsec IKE パケット	**udp**	**500**
WorkerIngress MasterIpsecIke	IPsec IKE パケット	**udp**	**500**
WorkerIngress IpsecNat	IPsec NAT-T パケット	**udp**	**4500**
WorkerIngress MasterIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
WorkerIngress IpsecEsp	IPsec ESP パケット	**50**	**All**
WorkerIngress MasterIpsecEsp	IPsec ESP パケット	**50**	**All**
WorkerIngress InternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
WorkerIngress MasterInternal UDP	内部クラスター通信	**udp**	**9000 - 9999**
WorkerIngress IngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**
WorkerIngress MasterIngress ServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**

ロールおよびインスタンスプロファイル

マシンには、AWS でのパーミッションを付与する必要があります。提供される CloudFormation テンプレートはマシンに対し、以下の **AWS::IAM::Role** オブジェクトについてのマシンの **Allow** パーミッションを付与し、それぞれのロールセットに **AWS::IAM::InstanceProfile** を指定します。テンプレートを使用しない場合、マシンには以下の広範囲のパーミッションまたは個別のパーミッションを付与することができます。

ロール	結果	アクション	リソース
マスター	**Allow**	**ec2:***	*
	Allow	**elasticloadbalancing:***	*
	Allow	**iam:PassRole**	*
	Allow	**s3:GetObject**	*
ワーカー	**Allow**	**ec2:Describe***	*
ブートストラップ	**Allow**	**ec2:Describe***	*
	Allow	**ec2:AttachVolume**	*
	Allow	**ec2:DetachVolume**	*

4.11.3.2. クラスターマシン

以下のマシンには **AWS::EC2::Instance** オブジェクトが必要になります。

- ブートストラップマシン。このマシンはインストール時に必要ですが、クラスターのデプロイ後に除去することができます。
- 3 つのコントロールプレーンマシンコントロールプレーンマシンはマシンセットによって制御されません。
- コンピュートマシン。インストール時に 2 つ以上のコンピュートマシン (ワーカーマシンとしても知られる) を作成する必要があります。これらのマシンはマシンセットによって制御されません。

4.11.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

4.11.3.4. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

例4.21 マシンのインスタンスタイプ

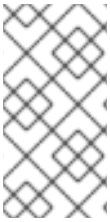
インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
t3.xlarge			x
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.11.3.5. IAM ユーザーに必要な AWS パーミッション



注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例4.22 インストールに必要な EC2 パーミッション

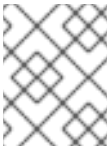
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**

- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**

- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例4.23 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例4.24 インストールに必要な Elastic Load Balancing (ELB) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**

- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

例4.25 インストールに必要な Elastic Load Balancing (ELBv2) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**

例4.26 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**

注記

AWS アカウントに Elastic Load Balancer (ELB) を作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例4.27 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**

- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例4.28 インストールに必要な S3 パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例4.29 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**

- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

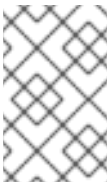
例4.30 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例4.31 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**

- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。代わりに、アカウントではネットワークリソースの削除に **tag:UntagResources** パーミッションのみが必要になります。

例4.32 共有インスタン出力ルが割り当てられたクラスターを削除するために必要なパーミッション

- **iam:UntagRole**

例4.33 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

注記

クラウドプロバイダーのクレデンシャルをミントモードで管理している場合に、IAM ユーザーには **iam:CreateAccessKey** と **iam:CreateUser** 権限も必要です。

例4.34 インスタンスのオプションのパーミッションおよびインストールのクォータチェック

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

4.11.4. AWS Marketplace イメージの取得

AWS Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に AWS を通じてサブスクライブする必要があります。オファーにサブスクライブすると、インストールプログラムがワーカーノードのデプロイに使用する AMI ID が提供されます。

注記

AWS Marketplace イメージを使用した OpenShift Container Platform クラスターのデプロイは、シークレットリージョンではサポートされていません。

前提条件

- オファーを購入するための AWS アカウントを持っている。このアカウントは、クラスターのインストールに使用されるアカウントと同じである必要はありません。

手順

1. AWS Marketplace で OpenShift Container Platform サブスクリプションを完了します。
2. 使用する特定のリージョンの AMI ID を記録します。CloudFormation テンプレートを使用してワーカーノードをデプロイする場合は、**worker0.type.properties.ImageID** パラメーターをこの値で更新する必要があります。

4.11.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

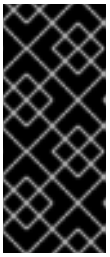
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。

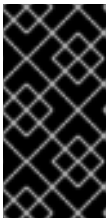
2. インフラストラクチャープロバイダーを選択します。

3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.11.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

4.11.7. AWS のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Amazon Web Services (AWS) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

4.11.7.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
```

```
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

4.11.7.2. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

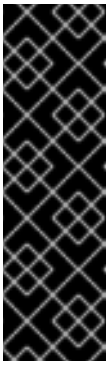
- ユーザーによってプロビジョニングされるインフラストラクチャー用の OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- Red Hat が公開している付随の Red Hat Enterprise Linux CoreOS (RHCOS) AMI のあるリージョンにクラスターをデプロイしていることを確認済みである。AWS GovCloud リージョンなどのカスタム AMI を必要とするリージョンにデプロイする場合は、**install-config.yaml** ファイルを手動で作成する必要があります。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **aws** を選択します。
      iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの **~/.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

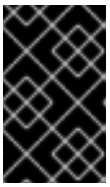
iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. オプション: **install-config.yaml** ファイルをバックアップします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

- AWS プロファイルおよび認証情報の設定についての詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

4.11.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

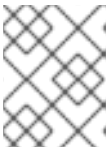
- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

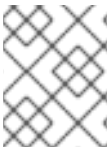


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.11.7.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

4.11.8. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Amazon Web Services (AWS) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な AWS リソースを見つけるためにも使用されます。提供される CloudFormation テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

4.11.9. AWS での VPC の作成

OpenShift Container Platform クラスターで使用する Virtual Private Cloud (VPC) を Amazon Web Services (AWS) で作成する必要があります。VPN およびルートテーブルを含む、各種要件を満たすように VPC をカスタマイズできます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、VPC を表す AWS リソースのスタックを作成できます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1. VPC の CIDR ブロック。
2. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
3. VPC をデプロイするアベイラビリティーゾーンの数。
4. **1** から **3** の間の整数を指定します。
5. 各アベイラビリティーゾーン内の各サブネットのサイズ。
6. **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このトピックの**VPC の CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

3. CloudFormation テンプレートを起動し、VPC を表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。

4.11.9.1. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例4.35 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
```

```
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
```

```
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
```

```
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
```

```
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
```

```
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
```

関連情報

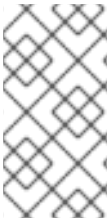
- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

4.11.10. AWS でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用できるネットワークおよび負荷分散 (classic または network) を Amazon Web Services (AWS) で設定する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なネットワークおよび負荷分散コンポーネントを表します。テンプレートは、ホストゾーンおよびサブネットタグも作成します。

単一 Virtual Private Cloud 内でテンプレートを複数回実行することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. クラスターの **install-config.yaml** ファイルに指定した Route 53 ベースドメインのホストゾーン ID を取得します。以下のコマンドを実行して、ホストゾーンの詳細を取得できます。

```
$ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
```

1 **<route53_domain>** について、クラスターの **install-config.yaml** ファイルを生成した時に作成した Route 53 ベースドメインを指定します。

出力例

```
mycluster.example.com. False 100
HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
/hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
```

この出力例では、ホストゾーン ID は **Z21IXYZ3-2Z2A4** です。

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
 {
  "ParameterKey": "ClusterName", 1
  "ParameterValue": "mycluster" 2
 },
 {
  "ParameterKey": "InfrastructureName", 3
  "ParameterValue": "mycluster-<random_string>" 4
 },
 {
  "ParameterKey": "HostedZoneId", 5
  "ParameterValue": "<random_string>" 6
 },
 {
  "ParameterKey": "HostedZoneName", 7
  "ParameterValue": "example.com" 8
 },
 {
  "ParameterKey": "PublicSubnets", 9
  "ParameterValue": "subnet-<random_string>" 10
 },
 {
  "ParameterKey": "PrivateSubnets", 11
  "ParameterValue": "subnet-<random_string>" 12
 },
 {
  "ParameterKey": "VpcId", 13
  "ParameterValue": "vpc-<random_string>" 14
 }
]
```

1 ホスト名などに使用する短いクラスター名。

2 クラスターの **install-config.yaml** ファイルを生成した時に使用したクラスター名を指定します。

3 クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前

チャーの名前。

4 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

5 ターゲットの登録に使用する Route 53 パブリックトゾーン ID。

6 **Z21IXYZABCZ2A4** に類する形式の Route 53 パブリックゾーン ID を指定します。この値は AWS コンソールから取得できます。

7 ターゲットの登録に使用する Route 53 ゾーン。

8 クラスターの **install-config.yaml** ファイルを生成した時に使用した Route 53 ベースドメインを指定します。AWS コンソールに表示される末尾のピリド (.) は含めないでください。

9 VPC 用に作成したパブリックサブネット。

10 VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

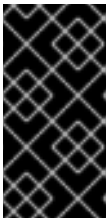
11 VPC 用に作成したプライベートサブネット。

12 VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。

13 クラスター用に作成した VPC。

14 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

3. このトピックの**ネットワークおよびロードバランサーの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。



重要

AWS government またはシークレットリージョンにクラスターをデプロイする場合は、CloudFormation テンプレートの **InternalApiServerRecord** を更新して、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。

4. CloudFormation テンプレートを起動し、ネットワークおよび負荷分散コンポーネントを提供する AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** は **cluster-dns** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4 提供されるテンプレートは一部の **AWS::IAM::Role** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-5cf0-12be5c33a183
```

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

PrivateHostedZoneId	プライベート DNS のホストゾーン ID。
ExternalApiLoadBalancerName	外部 API ロードバランサーのフルネーム。
InternalApiLoadBalancerName	内部 API ロードバランサーのフルネーム。
ApiServerDnsName	API サーバーの完全ホスト名。
RegisterNlbIpTargetsLambda	これらのロードバランサーの登録/登録解除に役立つ Lambda ARN。
ExternalApiTargetGroupArn	外部 API ターゲットグループの ARN。
InternalApiTargetGroupArn	内部 API ターゲットグループの ARN。

InternalServiceTargetGroupArn	内部サービスターゲットグループの ARN。

4.11.10.1. ネットワークおよびロードバランサーの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例4.36 ネットワークおよびロードバランサーの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
```

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
```

```
  Properties:
    HostedZoneConfig:
      Comment: "Managed by CloudFormation"
    Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
    - Key: Name
      Value: !Join ["-", [!Ref InfrastructureName, "int"]]
    - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "owned"
    VPCs:
    - VPCId: !Ref VpcId
      VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref HostedZoneId
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
```

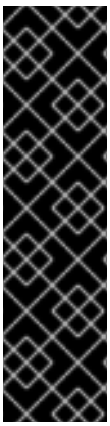
```
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalApiTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 6443
    Protocol: TCP

InternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
```

```
      Value: 60

  InternalServiceInternalListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalServiceTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 22623
      Protocol: TCP

  InternalServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/healthz"
      HealthCheckPort: 22623
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 22623
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  RegisterTargetLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
```

```
            ]
          Resource: !Ref InternalApiTargetGroup
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref InternalServiceTargetGroup
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.7"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
```

```
        - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.7"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
```

```
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```



重要

クラスターを AWS government またはシークレットリージョンにデプロイする場合は、**InternalApiServerRecord** を更新し、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。以下に例を示します。

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

- AWS Route 53 コンソール に移動して、ホストゾーンについての詳細を表示できます。
- パブリックホストゾーンの一覧表示についての詳細は、AWS ドキュメントの Listing public hosted zones を参照してください。

4.11.11. AWS でのセキュリティーグループおよびロールの作成

OpenShift Container Platform クラスターで使用するセキュリティーグループおよびロールを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なセキュリティーグループおよびロールを表します。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. VPC の CIDR ブロック。
4. **x.x.x.x/16-24** の形式で定義した VPC に使用した CIDR ブロックパラメーターを指定します。
5. VPC 用に作成したプライベートサブネット。
6. VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。
7. クラスター用に作成した VPC。
8. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

2. このトピックの**セキュリティーオブジェクトの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なセキュリティーグループおよびロールについて記述しています。

3. CloudFormation テンプレートを起動し、セキュリティーグループおよびロールを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1. **<name>** は **cluster-secs** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。
4. 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-6d7a-13fc0b61e9db
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

MasterSecurityGroupId	マスターセキュリティーグループ ID
WorkerSecurityGroupId	ワーカーセキュリティーグループ ID
MasterInstanceProfile	マスター IAM インスタンスプロファイル
WorkerInstanceProfile	ワーカー IAM インスタンスプロファイル

4.11.11.1. セキュリティーオブジェクトの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なセキュリティーオブジェクトをデプロイすることができます。

例4.37 セキュリティーオブジェクトの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
```

```
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
```

```
    SecurityGroupIngress:
    - IpProtocol: icmp
      FromPort: 0
      ToPort: 0
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

MasterIngressEtcd:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: etcd
    FromPort: 2379
    ToPort: 2380
    IpProtocol: tcp

MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressWorkerVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressWorkerGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

  MasterIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressWorkerIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressWorkerIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressWorkerIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
```

```
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressWorkerInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressWorkerKube:
```

```
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressWorkerIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
```

```
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressMasterVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressMasterGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
```

```
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

  WorkerIngressMasterIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressMasterIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressMasterIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
```

```
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal Kubernetes communication
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp
```

```
WorkerIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressMasterIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
          - "ec2:AttachVolume"
          - "ec2:AuthorizeSecurityGroupIngress"
          - "ec2:CreateSecurityGroup"
          - "ec2:CreateTags"
          - "ec2:CreateVolume"
          - "ec2:DeleteSecurityGroup"
          - "ec2:DeleteVolume"
          - "ec2:Describe*"
          - "ec2:DetachVolume"
          - "ec2:ModifyInstanceAttribute"
          - "ec2:ModifyVolume"
          - "ec2:RevokeSecurityGroupIngress"
          - "elasticloadbalancing:AddTags"
          - "elasticloadbalancing:AttachLoadBalancerToSubnets"
          - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
          - "elasticloadbalancing:CreateListener"
```

```
            - "elasticloadbalancing:CreateLoadBalancer"
            - "elasticloadbalancing:CreateLoadBalancerPolicy"
            - "elasticloadbalancing:CreateLoadBalancerListeners"
            - "elasticloadbalancing:CreateTargetGroup"
            - "elasticloadbalancing:ConfigureHealthCheck"
            - "elasticloadbalancing:DeleteListener"
            - "elasticloadbalancing:DeleteLoadBalancer"
            - "elasticloadbalancing:DeleteLoadBalancerListeners"
            - "elasticloadbalancing:DeleteTargetGroup"
            - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
            - "elasticloadbalancing:DeregisterTargets"
            - "elasticloadbalancing:Describe*"
            - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
            - "elasticloadbalancing:ModifyListener"
            - "elasticloadbalancing:ModifyLoadBalancerAttributes"
            - "elasticloadbalancing:ModifyTargetGroup"
            - "elasticloadbalancing:ModifyTargetGroupAttributes"
            - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
            - "elasticloadbalancing:RegisterTargets"
            - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
            - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
            - "kms:DescribeKey"
          Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
```

```
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

4.11.12. ストリームメタデータを使用した RHCOS AMI へのアクセス

OpenShift Container Platform では、**ストリームメタデータ** は、JSON 形式で RHCOS に関する標準化されたメタデータを 提供し、メタデータをクラスターに挿入します。ストリームメタデータは、複数のアーキテクチャーをサポートする安定した形式で、自動化を維持するための自己文書化が意図されています。

openshift-install の **coreos print-stream-json** サブコマンドを使用して、ストリームメタデータ形式のブートイメージに関する情報にアクセスできます。このコマンドは、スクリプト可能でマシン読み取り可能な形式でストリームメタデータを出力する方法を提供します。

ユーザーによってプロビジョニングされるインストールの場合、**openshift-install** バイナリーには、AWS AMI などの OpenShift Container Platform での使用がテストされている RHCOS ブートイメージのバージョンへの参照が含まれます。

手順

ストリームメタデータを解析するには、以下のいずれかの方法を使用します。

- Go プログラムから、https://github.com/coreos/stream-metadata-go の公式の **stream-metadata-go** ライブラリーを使用します。ライブラリーでサンプルコードを確認することもできます。
- Python や Ruby などの別のプログラミング言語から、お好みのプログラミング言語の JSON ライブラリーを使用します。
- **jq** などの JSON データを処理するコマンドラインユーティリティーから、以下のコマンドを実行します。
  - **us-west-1** などの、AWS リージョンの現在の **x86_64** AMI を出力します。

```
$ openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.images.aws.regions["us-west-1"].image'
```

出力例

```
ami-0d3e625f84626bbda
```

このコマンドの出力は、**us-west-1** リージョンの AWS AMI ID です。AMI はクラスターと同じリージョンに属する必要があります。

4.11.13. AWS インフラストラクチャーの RHCOS AMI

Red Hat は、OpenShift Container Platform ノードに手動で指定できるさまざまな AWS リージョンに有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を提供します。

注記

また、独自の AMI をインポートすることで、RHCOS AMI がパブリッシュされていないリージョンにインストールすることもできます。

表4.31 RHCOS AMI

AWS ゾーン	AWS AMI
af-south-1	**ami-0ce5aa99b7d576c79**
ap-east-1	**ami-0f6debc614042ce76**
ap-northeast-1	**ami-0423a1bf292f34dc3**
ap-northeast-2	**ami-0889161041cb9d77f**
ap-northeast-3	**ami-00564b0d6cbb676b1**
ap-south-1	**ami-0650f4166d12ccead**
ap-southeast-1	**ami-0b09ad848356811c7**
ap-southeast-2	**ami-013484d0474ab5860**
ca-central-1	**ami-03291c3e2b74c32b9**
eu-central-1	**ami-0510f6f15c25b29d4**
eu-north-1	**ami-03a3119ba25eb55b1**
eu-south-1	**ami-04f719435625c1313**

AWS ゾーン	AWS AMI
eu-west-1	**ami-08e20744bd1c89c8e**
eu-west-2	**ami-0c190f5d05b071c7a**
eu-west-3	**ami-0eb0bf894fdf1d416**
me-south-1	**ami-073928aa740f738bd**
sa-east-1	**ami-01242f1bac18cc0fd**
us-east-1	**ami-05ed2cc6e70392ff9**
us-east-2	**ami-00b3a5054da356288**
us-west-1	**ami-021f626622b5238f3**
us-west-2	**ami-0c9fd8b47bfd717e8**

4.11.13.1. 公開済み RHCOS AMI のない AWS リージョン

Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) または AWS software development kit (SDK) のネイティブサポートなしに、OpenShift Container Platform クラスターを Amazon Web Services (AWS) リージョンにデプロイできます。パブリッシュ済みの AMI が AWS リージョンで利用できない場合は、クラスターをインストールする前にカスタム AMI をアップロードできます。これは、クラスターを AWS government またはシークレットリージョンにデプロイする場合に必要です。AWS government およびシークレットリージョンは AWS SDK によってサポートされます。

AWS SDK によってサポートされないリージョンにデプロイしている場合で、カスタム AMI を指定しない場合、インストールプログラムは **us-east-1** AMI をユーザーアカウントに自動的にコピーします。次にインストールプログラムは、デフォルトまたはユーザー指定の Key Management Service (KMS) キーを使用して、暗号化された EBS ボリュームでコントロールプレーンマシンを作成します。これにより、AMI は、パブリッシュ済みの RHCOS AMI と同じプロセスワークフローを実施することができます。

RHCOS AMI のネイティブサポートのないリージョンはパブリッシュされないため、クラスターの作成時にターミナルから選択することはできません。ただし、**install-config.yaml** ファイルでカスタム AMI を設定して、このリージョンにインストールすることができます。

4.11.13.2. AWS でのカスタム RHCOS AMI のアップロード

カスタム Amazon Web Services (AWS) リージョンにデプロイする場合、そのリージョンに属するカスタム Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) をアップロードする必要があります。

前提条件

- AWS アカウントを設定している。
- 必要な IAM サービス出力 で、Amazon S3 バケットを作成している。

- RHCOS VMDK ファイルを Amazon S3 にアップロードしている。RHCOS VMDK ファイルは、インストールする OpenShift Container Platform のバージョンと同じか、またはそれ以下のバージョンである必要があります。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer を参照してください。

手順

1. AWS プロファイルを環境変数としてエクスポートします。

```
$ export AWS_PROFILE=<aws_profile> 1
```

1 **govcloud** などの AWS 認証情報を保持する AWS プロファイル名。

2. カスタム AMI に関連付けるリージョンを環境変数としてエクスポートします。

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

1 **us-gov-east-1** などの AWS リージョン。

3. 環境変数として Amazon S3 にアップロードした RHCOS のバージョンをエクスポートします。

```
$ export RHCOS_VERSION=<version> 1
```

1 **4.8.0** などの RHCOS VMDK バージョン。

4. Amazon S3 バケット名を環境変数としてエクスポートします。

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. **containers.json** ファイルを作成し、RHCOS VMDK ファイルを定義します。

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. RHCOS ディスクを Amazon EBS スナップショットとしてインポートします。

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1. **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64** などの RHCOS ディスクがインポートされていることの説明。
2. RHCOS ディスクを説明する JSON ファイルへのファイルパス。JSON ファイルには、Amazon S3 バケット名とキーが含まれている必要があります。

7. イメージインポートのステータスを確認します。

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

出力例

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

SnapshotId をコピーして、イメージを登録します。

8. RHCOS スナップショットからカスタム RHCOS AMI を作成します。

```
$ aws ec2 register-image \
  --region ${AWS_DEFAULT_REGION} \
  --architecture x86_64 \ 1
  --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
  --ena-support \
  --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
  --virtualization-type hvm \
  --root-device-name '/dev/xvda' \
  --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. **x86_64**、**s390x**、または **ppc64le** などの RHCOS VMDK アーキテクチャータイプ。
2. インポートされたスナップショットの **Description**。
3. RHCOS AMI の名前。

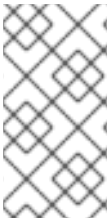
4 インポートされたスナップショットからの **SnapshotID**。

これらの API の詳細は、AWS ドキュメントの Importing a Disk as a Snapshot Using VM Import/Export および Creating a Linux AMI from a snapshot を参照してください。

4.11.14. AWS でのブートストラップノードの作成

OpenShift Container Platform クラスターの初期化で使用するブートストラップノードを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform インストールに必要なブートストラップノードを表します。



注記

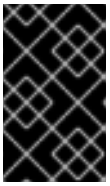
提供される CloudFormation テンプレートを使用してブートストラップノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。

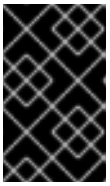
手順

1. **bootstrap.ign** Ignition 設定ファイルをクラスターに送るための場所を指定します。このファイルはインストールディレクトリーに置かれます。これを実行するための 1 つの方法として、クラスターのリージョンに S3 バケットを作成し、Ignition 設定ファイルをこれにアップロードします。



重要

提供される CloudFormation テンプレートでは、クラスターの Ignition 設定ファイルは S3 バケットから送られることを前提としています。このファイルを別の場所から送ることを選択する場合は、テンプレートを変更する必要があります。



重要

AWS SDK とは異なるエンドポイントを持つリージョンにデプロイする場合や、独自のカスタムエンドポイントを提供する場合は、**s3://** スキーマではなく、事前に署名済みの URL を S3 バケットに使用する必要があります。

注記

ブートストラップ Ignition 設定ファイルには、X.509 キーのようなシークレットが含まれません。以下の手順では、S3 バケットの基本的なセキュリティーを提供します。追加のセキュリティーを提供するには、OpenShift IAM ユーザーなどの特定のユーザーのみがバケットに含まれるオブジェクトにアクセスできるように S3 バケットポリシーを有効にできます。S3 を完全に回避し、ブートストラップマシンが到達できるアドレスからブートストラップ Ignition 設定ファイルを送ることができます。

a. バケットを作成します。

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** はバケット名です。**install-config.yaml** ファイルを作成する際に、**<cluster-name>** をクラスターに指定された名前に置き換えます。

b. **bootstrap.ign** Ignition 設定ファイルをバケットにアップロードします。

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-
infra/bootstrap.ign 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

c. ファイルがアップロードされていることを確認します。

```
$ aws s3 ls s3://<cluster-name>-infra/
```

出力例

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
 {
  "ParameterKey": "InfrastructureName", 1
  "ParameterValue": "mycluster-<random_string>" 2
 },
 {
  "ParameterKey": "RhcosAmi", 3
  "ParameterValue": "ami-<random_string>" 4
 },
 {
  "ParameterKey": "AllowedBootstrapSshCidr", 5
  "ParameterValue": "0.0.0.0/0" 6
 },
 {
  "ParameterKey": "PublicSubnet", 7
  "ParameterValue": "subnet-<random_string>" 8
```

```
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. ブートストラップノードに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. 有効な **AWS::EC2::Image::Id** 値を指定します。
5. ブートストラップノードへの SSH アクセスを許可する CIDR ブロック。
6. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。

7 ブートストラップを起動するために VPC に関連付けられるパブリックサブネット。

8 VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

9 マスターセキュリティーグループ ID (一時ルールの登録用)。

10 セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。

11 作成されたリソースが属する VPC。

12 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

13 ブートストラップの Ignition 設定ファイルをフェッチする場所。

14 **s3://<bucket_name>/bootstrap.ign** の形式で S3 バケットおよびファイル名を指定します。

15 ネットワークロードバランサー (NLB) を登録するかどうか。

16 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

17 NLB IP ターゲット登録 lambda グループの ARN。

18 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

19 外部 API ロードバランサーのターゲットグループの ARN。

20 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

21 内部 API ロードバランサーのターゲットグループの ARN。

22 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

23 内部サービスバランサーのターゲットグループの ARN。

24 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

3. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

4. CloudFormation テンプレートを起動し、ブートストラップノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml (2)
    --parameters file://<parameters>.json (3)
    --capabilities CAPABILITY_NAMED_IAM (4)
```

(1) **<name>** は **cluster-bootstrap** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

(2) **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

(3) **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

(4) 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-11eb-9dee-12dace8e3a83
```

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

Bootstrap InstanceId	ブートストラップインスタンス ID。
Bootstrap PublicIp	ブートストラップノードのパブリック IP アドレス。
Bootstrap PrivateIp	ブートストラップノードのプライベート IP アドレス。

4.11.14.1. ブートストラップマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイできます。

例4.38 ブートストラップマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String
```

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
```

```
        Principal:
          Service:
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action: "ec2:Describe*"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:AttachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:DetachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "s3:GetObject"
          Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
      VpcId: !Ref VpcId

  BootstrapInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: "i3.large"
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
```

```
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "PublicSubnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
        - {
          S3Loc: !Ref BootstrapIgnitionLocation
        }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
    Description: The bootstrap node public IP address.
    Value: !GetAtt BootstrapInstance.PublicIp

  BootstrapPrivateIp:
    Description: The bootstrap node private IP address.
    Value: !GetAtt BootstrapInstance.PrivateIp
```

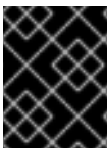
関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。
- AWS ゾーンの Red Hat Enterprise Linux CoreOS (RHCOS) AMI の詳細は、AWS インフラストラクチャーの RHCOS AMI を参照してください。

4.11.15. AWS でのコントロールプレーンの作成

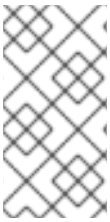
クラスターで使用するコントロールプレーンマシンを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、コントロールプレーンノードを表す AWS リソースのスタックを作成できます。



重要

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。



注記

提供される CloudFormation テンプレートを使用してコントロールプレーンノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
```

```
  "ParameterKey": "PrivateHostedZoneId", 7
  "ParameterValue": "<random_string>" 8
 },
 {
  "ParameterKey": "PrivateHostedZoneName", 9
  "ParameterValue": "mycluster.example.com" 10
 },
 {
  "ParameterKey": "Master0Subnet", 11
  "ParameterValue": "subnet-<random_string>" 12
 },
 {
  "ParameterKey": "Master1Subnet", 13
  "ParameterValue": "subnet-<random_string>" 14
 },
 {
  "ParameterKey": "Master2Subnet", 15
  "ParameterValue": "subnet-<random_string>" 16
 },
 {
  "ParameterKey": "MasterSecurityGroupId", 17
  "ParameterValue": "sg-<random_string>" 18
 },
 {
  "ParameterKey": "IgnitionLocation", 19
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
 },
 {
  "ParameterKey": "CertificateAuthorities", 21
  "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
 },
 {
  "ParameterKey": "MasterInstanceProfileName", 23
  "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
 },
 {
  "ParameterKey": "MasterInstanceType", 25
  "ParameterValue": "m5.xlarge" 26
 },
 {
  "ParameterKey": "AutoRegisterELB", 27
  "ParameterValue": "yes" 28
 },
 {
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
 },
 {
  "ParameterKey": "ExternalApiTargetGroupArn", 31
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
```

```
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. コントロールプレーンマシンに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. DNS etcd 登録を実行するかどうか。
6. **yes** または **no** を指定します。**yes** を指定する場合、ホストゾーンの情報を指定する必要があります。
7. etcd ターゲットの登録に使用する Route 53 プライベートゾーン ID。
8. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateHostedZoneId** 値を指定します。
9. ターゲットの登録に使用する Route 53 ゾーン。
10. **<cluster_name>.<domain_name>** を指定します。ここで、**<domain_name>** はクラスターの **install-config.yaml** ファイルの生成時に使用した Route 53 ベースドメインです。AWS コンソールに表示される末尾のピリド (.) は含めないでください。
11. 13. 15. コントロールプレーンマシンの起動に使用するサブネット (プライベートが望ましい)。
12. 14. 16. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
17. コントロールプレーンノード (別名マスターノード) に関連付けるマスターセキュリティーグループ ID。
18. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。
19. コントロールプレーンの Ignition 設定ファイルをフェッチする場所。
20. 生成される Ignition 設定ファイルの場所を指定します (**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)。

21 使用する base64 でエンコードされた認証局の文字列。

22 インストールディレクトリーにある **master.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

23 コントロールプレーンノードに関連付ける IAM プロファイル。

24 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **MasterInstanceProfile** パラメーターの値を指定します。

25 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

26 許可される値:

- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **c5.2xlarge**
- **c5.4xlarge**

- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**

27 ネットワークロードバランサー (NLB) を登録するかどうか。

28 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

29 NLB IP ターゲット登録 lambda グループの ARN。

30 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

31 外部 API ロードバランサーのターゲットグループの ARN。

32 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

33 内部 API ロードバランサーのターゲットグループの ARN。

34 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

35 内部サービスバランサーのターゲットグループの ARN。

36 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

2. このトピックの**コントロールプレーンマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。

3. **m5** インスタンスタイプを **MasterInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **MasterInstanceType.AllowedValues** パラメーターに追加します。

4. CloudFormation テンプレートを起動し、コントロールプレーンノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-control-plane** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-11eb-c6f6-0aa34627df4b
```

注記

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

4.11.15.1. コントロールプレーンマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例4.39 コントロールプレーンマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone information?
    Type: String
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
```

```
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String
    AllowedValues:
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "m5.xlarge"
    - "m5.2xlarge"
    - "m5.4xlarge"
    - "m5.8xlarge"
    - "m5.12xlarge"
    - "m5.16xlarge"
    - "m5a.xlarge"
    - "m5a.2xlarge"
    - "m5a.4xlarge"
    - "m5a.8xlarge"
    - "m5a.12xlarge"
    - "m5a.16xlarge"
    - "c4.2xlarge"
    - "c4.4xlarge"
    - "c4.8xlarge"
    - "c5.2xlarge"
    - "c5.4xlarge"
    - "c5.9xlarge"
    - "c5.12xlarge"
    - "c5.18xlarge"
    - "c5.24xlarge"
    - "c5a.2xlarge"
```

```
    - "c5a.4xlarge"
    - "c5a.8xlarge"
    - "c5a.12xlarge"
    - "c5a.16xlarge"
    - "c5a.24xlarge"
    - "r4.xlarge"
    - "r4.2xlarge"
    - "r4.4xlarge"
    - "r4.8xlarge"
    - "r4.16xlarge"
    - "r5.xlarge"
    - "r5.2xlarge"
    - "r5.4xlarge"
    - "r5.8xlarge"
    - "r5.12xlarge"
    - "r5.16xlarge"
    - "r5.24xlarge"
    - "r5a.xlarge"
    - "r5a.2xlarge"
    - "r5a.4xlarge"
    - "r5a.8xlarge"
    - "r5a.12xlarge"
    - "r5a.16xlarge"
    - "r5a.24xlarge"

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
```

```
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "DNS"
      Parameters:
      - AutoRegisterDNS
      - PrivateHostedZoneName
      - PrivateHostedZoneId
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
  ParameterLabels:
    InfrastructureName:
      default: "Infrastructure Name"
    VpcId:
      default: "VPC ID"
    Master0Subnet:
      default: "Master-0 Subnet"
    Master1Subnet:
      default: "Master-1 Subnet"
    Master2Subnet:
      default: "Master-2 Subnet"
    MasterInstanceType:
      default: "Master Instance Type"
    MasterInstanceProfileName:
      default: "Master Instance Profile Name"
    RhcosAmi:
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
      default: "Master Ignition Source"
    CertificateAuthorities:
      default: "Ignition CA String"
    MasterSecurityGroupId:
      default: "Master Security Group ID"
    AutoRegisterDNS:
      default: "Use Provided DNS Automation"
```

```
      AutoRegisterELB:
        default: "Use Provided ELB Automation"
      PrivateHostedZoneName:
        default: "Private Hosted Zone Name"
      PrivateHostedZoneId:
        default: "Private Hosted Zone ID"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
  DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp
```

```
  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
```

```
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
```

```
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  EtcdSrvRecords:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
      ]
      TTL: 60
      Type: SRV

  Etcd0Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master0.PrivateIp
      TTL: 60
      Type: A

  Etcd1Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master1.PrivateIp
      TTL: 60
      Type: A

  Etcd2Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master2.PrivateIp
      TTL: 60
```

```
      Type: A

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

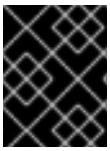
関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

4.11.16. AWS でのワーカーノードの作成

クラスターで使用するワーカーノードを Amazon Web Services (AWS) で作成できます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、ワーカーノードを表す AWS リソースのスタックを作成できます。



重要

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。それぞれのワーカーノードにスタックを作成する必要があります。

注記

提供される CloudFormation テンプレートを使用してワーカーノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.2xlarge" 16
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. ワーカーノードに使用する最新の Red Hat Enterprise Linux CoreOS(RHCOS)AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. ワーカーノードを起動するためのサブネット (プライベートであることが望ましい)。
6.

DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。

7 ワーカーノードに関連付けるワーカーセキュリティーグループ ID。

8 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WorkerSecurityGroupId** 値を指定します。

9 ブートストラップ Ignition 設定ファイルを取得する場所。

10 生成される Ignition 設定の場所を指定します。 **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**

11 使用する base64 でエンコードされた認証局の文字列。

12 インストールディレクトリーにある **worker.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

13 ワーカーロールに関連付ける IAM プロファイル。

14 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WokerInstanceProfile** パラメーターの値を指定します。

15 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

16 許可される値:

- **m4.large**
- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.large**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.large**
- **m5a.xlarge**
- **m5a.2xlarge**

- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.large**
- **c4.xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **c5.large**
- **c5.xlarge**
- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.large**
- **c5a.xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.large**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**

- **r4.8xlarge**
- **r4.16xlarge**
- **r5.large**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.large**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**
- **t3.large**
- **t3.xlarge**
- **t3.2xlarge**
- **t3a.large**
- **t3a.xlarge**
- **t3a.2xlarge**

2. このトピックの**ワーカーマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークオブジェクトおよびロードバランサーについて記述しています。

3. オプション: **m5** インスタンスタイプを **WorkerInstanceType** の値として指定した場合は、そのインスタンスタイプを CloudFormation テンプレートの **WorkerInstanceType.AllowedValues** パラメーターに追加します。

4. オプション: AWS Marketplace イメージを使用してデプロイする場合は、サブスクリプションから取得した AMI ID で **Worker0.type.properties.ImageID** パラメーターを更新します。

5. CloudFormation テンプレートを起動し、ワーカーノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml \ (2)
    --parameters file://<parameters>.json (3)
```

(1) **<name>** は **cluster-worker-1** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

(2) **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

(3) **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-11eb-348f-sd9888c65b59
```

注記

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。

6. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. クラスターに作成するワーカーマシンが十分な数に達するまでワーカースタックの作成を継続します。同じテンプレートおよびパラメーターファイルを参照し、異なるスタック名を指定してワーカースタックをさらに作成することができます。

重要

2 つ以上のワーカーマシンを作成する必要があるため、この CloudFormation テンプレートを使用する 2 つ以上のスタックを作成する必要があります。

4.11.16.1. ワーカーマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例4.40 ワーカーマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String
    AllowedValues:
    - "m4.large"
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "m5.large"
    - "m5.xlarge"
    - "m5.2xlarge"
    - "m5.4xlarge"
    - "m5.8xlarge"
    - "m5.12xlarge"
    - "m5.16xlarge"
    - "m5a.large"
    - "m5a.xlarge"
    - "m5a.2xlarge"
    - "m5a.4xlarge"
    - "m5a.8xlarge"
```

```
      - "m5a.12xlarge"
      - "m5a.16xlarge"
      - "c4.large"
      - "c4.xlarge"
      - "c4.2xlarge"
      - "c4.4xlarge"
      - "c4.8xlarge"
      - "c5.large"
      - "c5.xlarge"
      - "c5.2xlarge"
      - "c5.4xlarge"
      - "c5.9xlarge"
      - "c5.12xlarge"
      - "c5.18xlarge"
      - "c5.24xlarge"
      - "c5a.large"
      - "c5a.xlarge"
      - "c5a.2xlarge"
      - "c5a.4xlarge"
      - "c5a.8xlarge"
      - "c5a.12xlarge"
      - "c5a.16xlarge"
      - "c5a.24xlarge"
      - "r4.large"
      - "r4.xlarge"
      - "r4.2xlarge"
      - "r4.4xlarge"
      - "r4.8xlarge"
      - "r4.16xlarge"
      - "r5.large"
      - "r5.xlarge"
      - "r5.2xlarge"
      - "r5.4xlarge"
      - "r5.8xlarge"
      - "r5.12xlarge"
      - "r5.16xlarge"
      - "r5.24xlarge"
      - "r5a.large"
      - "r5a.xlarge"
      - "r5a.2xlarge"
      - "r5a.4xlarge"
      - "r5a.8xlarge"
      - "r5a.12xlarge"
      - "r5a.16xlarge"
      - "r5a.24xlarge"
      - "t3.large"
      - "t3.xlarge"
      - "t3.2xlarge"
      - "t3a.large"
      - "t3a.xlarge"
      - "t3a.2xlarge"

    Metadata:
      AWS::CloudFormation::Interface:
        ParameterGroups:
        - Label:
```

```
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
```

```
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

関連情報

- AWS CloudFormation コンソール に移動し、作成する CloudFormation スタックについての詳細を表示できます。

4.11.17. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS でのブートストラップシーケンスの初期化

Amazon Web Services (AWS) ですべての必要なインフラストラクチャーを作成した後に、OpenShift Container Platform コントロールプレーンを初期化するブートストラップシーケンスを開始できます。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。
- ワーカーノードを作成している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、OpenShift Container Platform コントロールプレーンを初期化するブートストラッププロセスを開始します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.19.0+9f84db3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

コマンドが **FATAL** 警告を出さずに終了する場合、OpenShift Container Platform コントロールプレーンは初期化されています。

注記

コントロールプレーンの初期化後に、コンピュートノードを設定し、Operator の形式で追加のサービスをインストールします。

関連情報

- OpenShift Container Platform インストールの進捗としてインストール、ブートストラップ、およびコントロールプレーンのログをモニターリングする方法についての詳細は、インストールの進捗のモニターリング を参照してください。
- ブートストラッププロセスに関する問題のトラブルシューティングの詳細は、ブートストラップノードの診断データの収集 を参照してください。
- AWS EC2 コンソールを使用して、作成される実行中のインスタンスについての詳細を表示できます。

4.11.18. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

4.11.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.11.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

4.11.21. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

4.11.21.1. イメージレジストリーストレージの設定

Amazon Web Services はデフォルトのストレージを提供します。つまり、Image Registry Operator はインストール後に利用可能になります。ただし、レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、レジストリーストレージを手動で設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

AWS のユーザーによってプロビジョニングされるインフラストラクチャーのレジストリーストレージを設定し、OpenShift Container Platform を非表示のリージョンにデプロイできます。詳細は、Configuring the registry for AWS user-provisioned infrastructure を参照してください。

4.11.21.1.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS 上にクラスターがある。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

4.11.21.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

4.11.22. ブートストラップリソースの削除

クラスターの初期 Operator 設定の完了後に、Amazon Web Services (AWS) からブートストラップリソースを削除します。

前提条件

- クラスターの初期 Operator 設定が完了済みです。

手順

1. ブートストラップリソースを削除します。CloudFormation テンプレートを使用した場合は、そのスタックを削除 します。
   - AWS CLI を使用してスタックを削除します。

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** は、ブートストラップスタックの名前です。

   - AWS CloudFormation コンソール を使用してスタックを削除します。

4.11.23. Ingress DNS レコードの作成

DNS ゾーン設定を削除した場合には、Ingress ロードバランサーを参照する DNS レコードを手動で作成します。ワイルドカードレコードまたは特定のレコードのいずれかを作成できます。以下の手順では A レコードを使用しますが、CNAME やエイリアスなどの必要な他のレコードタイプを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用する OpenShift Container Platform クラスターを Amazon Web Services (AWS) にデプロイしています。
- OpenShift CLI (**oc**) がインストールされている。
- **jq** パッケージをインストールしている。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。

手順

1. 作成するルートを決定します。
   - ワイルドカードレコードを作成するには、***.apps.<cluster_name>.<domain_name>** を使用します。ここで、**<cluster_name>** はクラスター名で、**<domain_name>** は OpenShift Container Platform クラスターの Route 53 ベースドメインです。
   - 特定のレコードを作成するには、以下のコマンドの出力にあるように、クラスターが使用する各ルートにレコードを作成する必要があります。

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     出力例

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Ingress Operator ロードバランサーのステータスを取得し、使用する外部 IP アドレスの値をメモします。これは **EXTERNAL-IP** 列に表示されます。

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   出力例

   ```
   NAME             TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
   AGE
   router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
   80:31499/TCP,443:30693/TCP   5m
   ```

3. ロードバランサーのホストゾーン ID を見つけます。

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
"<external_ip>").CanonicalHostedZoneNameID' 1
```

1 **<external_ip>** については、取得した Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。

出力例

```
Z3AADJGX6KTTL2
```

このコマンドの出力は、ロードバランサーのホストゾーン ID です。

4. クラスターのドメインのパブリックホストゾーン ID を取得します。

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>** については、OpenShift Container Platform クラスターの Route 53 ベースドメインを指定します。

出力例

```
/hostedzone/Z3URY6TWQ91KVV
```

ドメインのパブリックホストゾーン ID がコマンド出力に表示されます。この例では、これは **Z3URY6TWQ91KVV** になります。

5. プライベートゾーンにエイリアスレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1. **<private_hosted_zone_id>** については、DNS および負荷分散の CloudFormation テンプレートの出力から値を指定します。
2. **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。
3. **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。
4. **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6. パブリックゾーンにレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1. **<public_hosted_zone_id>** については、ドメインのパブリックホストゾーンを指定します。
2. **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。
3. **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。
4. **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

4.11.24. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行

Amazon Web Service (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後に、デプロイメントを完了するまでモニターします。

前提条件

前提条件

- OpenShift Container Platform クラスターのブートストラップノードを、ユーザーによってプロビジョニングされた AWS インフラストラクチャーで削除している。
- **oc** CLI をインストールしていること。

手順

- インストールプログラムが含まれるディレクトリーから、クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Fe5en-ymBEc-
Wt6NL"
INFO Time elapsed: 1s
```

重要

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

4.11.25. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

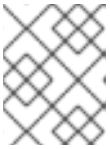
前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```



注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>           console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.11.26. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.11.27. 関連情報

- AWS CloudFormation スタックについての詳細は、Working with stacks を参照してください。

4.11.28. 次のステップ

- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.12. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、独自に提供するインフラストラクチャーおよびインストールリリースコンテンツの内部ミラーを使用して、クラスターを Amazon Web Services (AWS) にインストールできます。

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが AWS API を使用するにはインターネットへのアクセスが必要になります。

このインフラストラクチャーを作成する 1 つの方法として、提供される CloudFormation テンプレートを使用できます。テンプレートを変更してインフラストラクチャーをカスタマイズしたり、それらに含まれる情報を使用し、所属する会社のポリシーに基づいて AWS オブジェクトを作成したりできます。

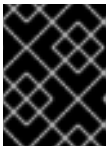
重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の CloudFormation テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

4.12.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

- ミラーホストでミラーレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得しています。



重要

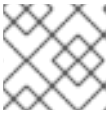
インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターをホストするために AWS アカウントを設定 しています。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

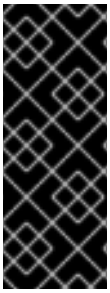
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

4.12.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

4.12.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

4.12.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.12.4. 必要な AWS インフラストラクチャーコンポーネント

OpenShift Container Platform を Amazon Web Services (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、マシンとサポートするインフラストラクチャーの両方を手動で作成する必要があります。

各種プラットフォームの統合テストの詳細については、OpenShift Container Platform 4.x のテスト済みインテグレーション のページを参照してください。

提供される CloudFormation テンプレートを使用すると、以下のコンポーネントを表す AWS リソースのスタックを作成できます。

- AWS Virtual Private Cloud (VPC)
- ネットワークおよび負荷分散コンポーネント
- セキュリティーグループおよびロール
- OpenShift Container Platform ブートストラップノード
- OpenShift Container Platform コントロールプレーンノード
- OpenShift Container Platform コンピュートノード

または、コンポーネントを手動で作成するか、またはクラスターの要件を満たす既存のインフラストラクチャーを再利用できます。コンポーネントの相互関係についての詳細は、CloudFormation テンプレートを参照してください。

4.12.4.1. 他のインフラストラクチャーコンポーネント

- 1 つの VPC
- DNS エントリー
- ロードバランサー (classic または network) およびリスナー
- パブリックおよびプライベート Route 53 ゾーン
- セキュリティーグループ
- IAM ロール
- S3 バケット

非接続環境で作業している場合、またはプロキシーを使用する場合には、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これらのエンドポイントに到達するには、VPC エンドポイントを作成してクラスターが使用するサブネットに割り当てる必要があります。以下のエンドポイントを作成します。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

必要な DNS および負荷分散コンポーネント

DNS およびロードバランサー設定では、パブリックホストゾーンを使用する必要があり、クラスターのインフラストラクチャーをプロビジョニングする場合にインストールプログラムが使用するものと同様のプライベートホストゾーンを使用できます。ロードバランサーに解決する DNS エントリーを作成する必要があります。**api.<cluster_name>.<domain>** のエントリーは外部ロードバランサーを参照し、**api-int.<cluster_name>.<domain>** のエントリーは内部ロードバランサーを参照する必要があります。

またクラスターには、Kubernetes API とその拡張に必要なポート 6443、および新規マシンの Ignition 設定ファイルに必要なポート 22623 のロードバランサーおよびリスナーが必要です。ターゲットはコントロールプレーンノード (別名マスターノード) になります。ポート 6443 はクラスター外のクライアントとクラスター内のノードからもアクセスできる必要があります。ポート 22623 はクラスター内のノードからアクセスできる必要があります。

コンポーネント	AWS タイプ	説明
DNS	**AWS::Route53::HostedZone**	内部 DNS のホストゾーン。
etcd レコードセット	**AWS::Route53::RecordSet**	コントロールプレーンマシンの etcd の登録レコード。
パブリックロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	パブリックサブネットのロードバランサー。
外部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	外部 API サーバーのエイリアスレコード。
外部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	外部ロードバランサー用のポート 6443 のリスナー。

コンポーネント	AWS タイプ	説明
外部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	外部ロードバランサーのターゲットグループ。
プライベートロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	プライベートサブネットのロードバランサー。
内部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	内部 API サーバーのエイリアスレコード。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサー用のポート 22623 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサーのポート 6443 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。

セキュリティーグループ

コントロールプレーンおよびワーカーマシンには、以下のポートへのアクセスが必要です。

グループ	タイプ	IP プロトコル	ポート範囲
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
		tcp	**6443**

グループ	タイプ	IP プロトコル	ポート範囲
		tcp	**22623**
WorkerSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
BootstrapSecurityGroup	**AWS::EC2::SecurityGroup**	**tcp**	**22**
		tcp	**19531**

コントロールプレーンの Ingress

コントロールプレーンマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressEtcd	etcd	**tcp**	**2379- 2380**
MasterIngressVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressInternal	内部クラスター通信および Kubernetes プロキシーメトリクス	**tcp**	**9000 - 9999**
MasterIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
MasterIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250 - 10259**
MasterIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250 - 10259**
MasterIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
MasterIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressGeneve	Geneve パケット	**udp**	**6081**
MasterIngressWorkerGeneve	Geneve パケット	**udp**	**6081**
MasterIngressIpsecIke	IPsec IKE パケット	**udp**	**500**
MasterIngressWorkerIpsecIke	IPsec IKE パケット	**udp**	**500**
MasterIngressIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
MasterIngressWorkerIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
MasterIngressIpsecEsp	IPsec ESP パケット	**50**	**All**
MasterIngressWorkerIpsecEsp	IPsec ESP パケット	**50**	**All**
MasterIngressInternalUDP	内部クラスター通信	**udp**	**9000** - **9999**
MasterIngressWorkerInternalUDP	内部クラスター通信	**udp**	**9000** - **9999**
MasterIngressIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000** - **32767**
MasterIngressWorkerIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000** - **32767**

ワーカーの Ingress

ワーカーマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngressGeneve	Geneve パケット	**udp**	**6081**
WorkerIngressMasterGeneve	Geneve パケット	**udp**	**6081**
WorkerIngressIpsecIke	IPsec IKE パケット	**udp**	**500**
WorkerIngressMasterIpsecIke	IPsec IKE パケット	**udp**	**500**
WorkerIngressIpsecNat	IPsec NAT-T パケット	**udp**	**4500**
WorkerIngressMasterIpsecNat	IPsec NAT-T パケット	**udp**	**4500**

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressIpsecEsp	IPsec ESP パケット	**50**	**All**
WorkerIngressMasterIpsecEsp	IPsec ESP パケット	**50**	**All**
WorkerIngressInternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
WorkerIngressMasterInternalUDP	内部クラスター通信	**udp**	**9000 - 9999**
WorkerIngressIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**
WorkerIngressMasterIngressServicesUDP	Kubernetes Ingress サービス	**udp**	**30000 - 32767**

ロールおよびインスタンスプロファイル

マシンには、AWS でのパーミッションを付与する必要があります。提供される CloudFormation テンプレートはマシンに対し、以下の **AWS::IAM::Role** オブジェクトについてのマシンの **Allow** パーミッションを付与し、それぞれのロールセットに **AWS::IAM::InstanceProfile** を指定します。テンプレートを使用しない場合、マシンには以下の広範囲のパーミッションまたは個別のパーミッションを付与することができます。

ロール	結果	アクション	リソース
マスター	**Allow**	**ec2:***	*
	Allow	**elasticloadbalancing:***	*
	Allow	**iam:PassRole**	*
	Allow	**s3:GetObject**	*
ワーカー	**Allow**	**ec2:Describe***	*
ブートストラップ	**Allow**	**ec2:Describe***	*
	Allow	**ec2:AttachVolume**	*

ロール	結果	アクション	リソース
	Allow	**ec2:DetachVolume**	*

4.12.4.2. クラスターマシン

以下のマシンには **AWS::EC2::Instance** オブジェクトが必要になります。

- ブートストラップマシン。このマシンはインストール時に必要ですが、クラスターのデプロイ後に除去することができます。
- 3 つのコントロールプレーンマシンコントロールプレーンマシンはマシンセットによって制御されません。
- コンピュートマシン。インストール時に 2 つ以上のコンピュートマシン (ワーカーマシンとしても知られる) を作成する必要があります。これらのマシンはマシンセットによって制御されません。

4.12.4.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

4.12.4.4. サポートされる AWS マシンタイプ

以下の Amazon Web Services (AWS) イ ンスタンスタイプは、OpenShift Container Platform でサポートされています。

例4.41 マシンのインスタンスタイプ

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large			x
m4.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m5.large			x
m5.xlarge		x	x
m5.2xlarge		x	x
m5.4xlarge		x	x
m5.8xlarge		x	x
m5.12xlarge		x	x
m5.16xlarge		x	x
m5a.large			x
m5a.xlarge		x	x
m5a.2xlarge		x	x
m5a.4xlarge		x	x
m5a.8xlarge		x	x
m5a.12xlarge		x	x
m5a.16xlarge		x	x
m6i.xlarge		x	x
m6i.2xlarge		x	x
m6i.4xlarge		x	x
m6i.8xlarge		x	x
m6i.16xlarge		x	x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
c5.xlarge			x
c5.2xlarge		x	x
c5.4xlarge		x	x
c5.9xlarge		x	x
c5.12xlarge		x	x
c5.18xlarge		x	x
c5.24xlarge		x	x
c5a.xlarge			x
c5a.2xlarge		x	x
c5a.4xlarge		x	x
c5a.8xlarge		x	x
c5a.12xlarge		x	x
c5a.16xlarge		x	x
c5a.24xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x
r5.large			x
r5.xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r5.2xlarge		x	x
r5.4xlarge		x	x
r5.8xlarge		x	x
r5.12xlarge		x	x
r5.16xlarge		x	x
r5.24xlarge		x	x
r5a.large			x
r5a.xlarge		x	x
r5a.2xlarge		x	x
r5a.4xlarge		x	x
r5a.8xlarge		x	x
r5a.12xlarge		x	x
r5a.16xlarge		x	x
r5a.24xlarge		x	x
t3.large			x
t3.xlarge			x
t3.2xlarge			x
t3a.large			x
t3a.xlarge			x
t3a.2xlarge			x

4.12.4.5. IAM ユーザーに必要な AWS パーミッション

注記

ベースクラスターリソースを削除するには、IAM ユーザーが領域 **us-east-1** にアクセス許可 **tag:GetResources** を持っている必要があります。AWS API 要件の一部として、OpenShift Container Platform インストールプログラムはこのリージョンでさまざまなアクションを実行します。

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例4.42 インストールに必要な EC2 パーミッション

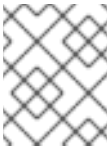
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInternetGateways**

- ec2:DescribeKeyPairs
- ec2:DescribeNatGateways
- ec2:DescribeNetworkAcls
- ec2:DescribeNetworkInterfaces
- ec2:DescribePrefixLists
- ec2:DescribeRegions
- ec2:DescribeRouteTables
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeTags
- ec2:DescribeVolumes
- ec2:DescribeVpcAttribute
- ec2:DescribeVpcClassicLink
- ec2:DescribeVpcClassicLinkDnsSupport
- ec2:DescribeVpcEndpoints
- ec2:DescribeVpcs
- ec2:GetEbsDefaultKmsKeyId
- ec2:ModifyInstanceAttribute
- ec2:ModifyNetworkInterfaceAttribute
- ec2:RevokeSecurityGroupEgress
- ec2:RevokeSecurityGroupIngress
- ec2:RunInstances
- ec2:TerminateInstances

例4.43 インストール時のネットワークリソースの作成に必要なパーミッション

- ec2:AllocateAddress
- ec2:AssociateAddress
- ec2:AssociateDhcpOptions
- ec2:AssociateRouteTable
- ec2:AttachInternetGateway

- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例4.44 インストールに必要な Elastic Load Balancing (ELB) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

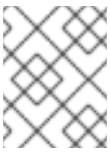
例4.45 インストールに必要な Elastic Load Balancing (ELBv2) のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**

例4.46 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**

- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



注記

AWS アカウントに Elastic Load Balancer (ELB) を作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例4.47 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例4.48 インストールに必要な S3 パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**

- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例4.49 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

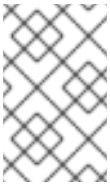
例4.50 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**

- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例4.51 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。代わりに、アカウントではネットワークリソースの削除に **tag:UntagResources** パーミッションのみが必要になります。

例4.52 共有インスタン出力ルが割り当てられたクラスターを削除するために必要なパーミッション

- **iam:UntagRole**

例4.53 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

注記

クラウドプロバイダーのクレデンシャルをミントモードで管理している場合に、IAM ユーザーには **iam:CreateAccessKey** と **iam:CreateUser** 権限も必要です。

例4.54 インスタンスのオプションのパーミッションおよびインストールのクォータチェック

- **ec2:DescribeInstanceTypeOfferings**
- **servicequotas:ListAWSDefaultServiceQuotas**

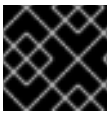
4.12.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS

(RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

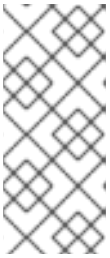
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

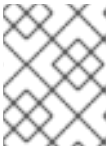
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

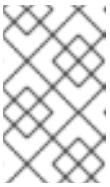
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

4.12.6. AWS のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Amazon Web Services (AWS) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

4.12.6.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

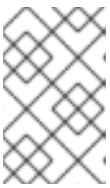
```
99_kubeadmin-password-secret.yaml
```

```
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、**manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

4.12.6.2. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

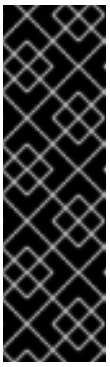
- ユーザーによってプロビジョニングされるインフラストラクチャー用の OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- Red Hat が公開している付随の Red Hat Enterprise Linux CoreOS (RHCOS) AMI のあるリージョンにクラスターをデプロイしていることを確認済みである。AWS GovCloud リージョンなどのカスタム AMI を必要とするリージョンにデプロイする場合は、**install-config.yaml** ファイルを手動で作成する必要があります。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **aws** を選択します。

iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

注記

AWS アクセスキー ID およびシークレットアクセスキーは、インストールホストの現行ユーザーのホームディレクトリーの **~/.aws/credentials** に保存されます。エクスポートされたプロファイルの認証情報がファイルにない場合は、インストールプログラムにより認証情報の入力が求めるプロンプトが出されます。インストールプログラムに指定する認証情報は、ファイルに保存されます。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<local_registry> については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

```
additionalTrustBundle: |
 -----BEGIN CERTIFICATE-----
```

```
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
-----END CERTIFICATE-----
```

c. イメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

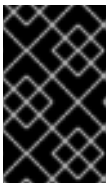
コマンドの出力の **imageContentSources** セクションを使用して、リポジトリー、またはネットワークが制限されたネットワークに取り込んだメディアからのコンテンツをミラーリングする際に使用した値をミラーリングします。

d. オプション: パブリッシュストラテジーを **Internal** に設定します。

```
publish: Internal
```

このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

3. オプション: **install-config.yaml** ファイルをバックアップします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

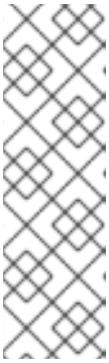
- AWS プロファイルおよび認証情報の設定についての詳細は、AWS ドキュメントの Configuration and credential file settings を参照してください。

4.12.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

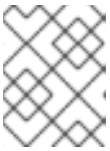
- クラスターが AWS にある場合は、**ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com** および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシーはノードレベルではなくコンテナーレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシーサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

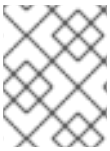


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



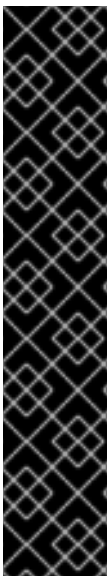
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.12.6.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

4.12.7. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Amazon Web Services (AWS) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な AWS リソースを見つけるためにも使用されます。提供される CloudFormation テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

4.12.8. AWS での VPC の作成

OpenShift Container Platform クラスターで使用する Virtual Private Cloud (VPC) を Amazon Web Services (AWS) で作成する必要があります。VPN およびルートテーブルを含む、各種要件を満たすように VPC をカスタマイズできます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、VPC を表す AWS リソースのスタックを作成できます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1 VPC の CIDR ブロック。

2 **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。

3 VPC をデプロイするアベイラビリティーゾーンの数。

4 **1** から **3** の間の整数を指定します。

5 各アベイラビリティーゾーン内の各サブネットのサイズ。

6 **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このトピックの**VPC の CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

3. CloudFormation テンプレートを起動し、VPC を表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-820e-12a48460849f
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。

4.12.8.1. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例4.55 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]
```

```
Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
```

```
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
```

```
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP2
        - AllocationId
      SubnetId: !Ref PublicSubnet2
  EIP2:
    Type: "AWS::EC2::EIP"
    Condition: DoAz2
    Properties:
      Domain: vpc
  Route2:
    Type: "AWS::EC2::Route"
    Condition: DoAz2
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT2
  PrivateSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable3:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation3:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
```

```
    Condition: DoAz3
    Properties:
      SubnetId: !Ref PrivateSubnet3
      RouteTableId: !Ref PrivateRouteTable3
  NAT3:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
```

```
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
```

4.12.9. AWS でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用できるネットワークおよび負荷分散 (classic または network) を Amazon Web Services (AWS) で設定する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なネットワークおよび負荷分散コンポーネントを表します。テンプレートは、ホストゾーンおよびサブネットタグも作成します。

単一 Virtual Private Cloud 内でテンプレートを複数回実行することができます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. クラスターの **install-config.yaml** ファイルに指定した Route 53 ベースドメインのホストゾーン ID を取得します。以下のコマンドを実行して、ホストゾーンの詳細を取得できます。

```
$ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
```

■

1 **<route53_domain>** について、クラスターの **install-config.yaml** ファイルを生成した時に作成した Route 53 ベースドメインを指定します。

出力例

```
mycluster.example.com. False 100
HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
/hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
```

この出力例では、ホストゾーン ID は **Z21IXYZ3-2Z2A4** です。

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

1 ホスト名などに使用する短いクラスター名。

2 クラスターの **install-config.yaml** ファイルを生成した時に使用したクラスター名を指定します。

3 クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

4 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

5 ターゲットの登録に使用する Route 53 パブリックトゾーン ID。

6 **Z21IXYZABCZ2A4** に類する形式の Route 53 パブリックゾーン ID を指定します。この値は AWS コンソールから取得できます。

7 ターゲットの登録に使用する Route 53 ゾーン。

8 クラスターの **install-config.yaml** ファイルを生成した時に使用した Route 53 ベースドメインを指定します。AWS コンソールに表示される末尾のピリド (.) は含めないでください。

9 VPC 用に作成したパブリックサブネット。

10 VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

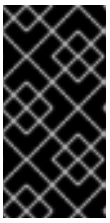
11 VPC 用に作成したプライベートサブネット。

12 VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。

13 クラスター用に作成した VPC。

14 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

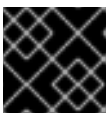
3. このトピックの**ネットワークおよびロードバランサーの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。



重要

AWS government またはシークレットリージョンにクラスターをデプロイする場合は、CloudFormation テンプレートの **InternalApiServerRecord** を更新して、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。

4. CloudFormation テンプレートを起動し、ネットワークおよび負荷分散コンポーネントを提供する AWS リソースのスタックを作成します。



重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** は **cluster-dns** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4 提供されるテンプレートは一部の **AWS::IAM::Role** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-5cf0-12be5c33a183
```

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

PrivateHostedZoneId	プライベート DNS のホストゾーン ID。
ExternalApiLoadBalancerName	外部 API ロードバランサーのフルネーム。
InternalApiLoadBalancerName	内部 API ロードバランサーのフルネーム。
ApiServerDnsName	API サーバーの完全ホスト名。
RegisterNlbIpTargetsLambda	これらのロードバランサーの登録/登録解除に役立つ Lambda ARN。
ExternalApiTargetGroupArn	外部 API ターゲットグループの ARN。
InternalApiTargetGroupArn	内部 API ターゲットグループの ARN。

InternalServiceTargetGroupArn	内部サービスターゲットグループの ARN。

4.12.9.1. ネットワークおよびロードバランサーの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例4.56 ネットワークおよびロードバランサーの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
```

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
```

```
  Properties:
    HostedZoneConfig:
      Comment: "Managed by CloudFormation"
    Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
    - Key: Name
      Value: !Join ["-", [!Ref InfrastructureName, "int"]]
    - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "owned"
    VPCs:
    - VPCId: !Ref VpcId
      VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref HostedZoneId
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
```

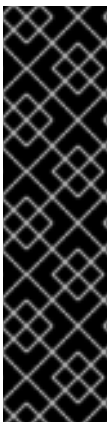
```
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalApiTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 6443
    Protocol: TCP

InternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
```

```
      Value: 60

InternalServiceInternalListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalServiceTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 22623
    Protocol: TCP

InternalServiceTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/healthz"
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

RegisterTargetLambdaIamRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "lambda.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
```

```
          ]
        Resource: !Ref InternalApiTargetGroup
      - Effect: "Allow"
        Action:
          [
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:DeregisterTargets",
          ]
        Resource: !Ref InternalServiceTargetGroup
      - Effect: "Allow"
        Action:
          [
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:DeregisterTargets",
          ]
        Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.7"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
```

```
        - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.7"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
```

```
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```



重要

クラスターを AWS government またはシークレットリージョンにデプロイする場合は、**InternalApiServerRecord** を更新し、**CNAME** レコードを使用する必要があります。**ALIAS** タイプのレコードは、AWS 政府リージョンではサポートされません。以下に例を示します。

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

関連情報

- パブリックホストゾーンの一覧表示についての詳細は、AWS ドキュメントの Listing public hosted zones を参照してください。

4.12.10. AWS でのセキュリティーグループおよびロールの作成

OpenShift Container Platform クラスターで使用するセキュリティーグループおよびロールを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform クラスターに必要なセキュリティーグループおよびロールを表します。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

3 VPC の CIDR ブロック。

4 **x.x.x.x/16-24** の形式で定義した VPC に使用した CIDR ブロックパラメーターを指定します。

5 VPC 用に作成したプライベートサブネット。

6 VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。

7 クラスター用に作成した VPC。

8 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

2. このトピックの**セキュリティーオブジェクトの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なセキュリティーグループおよびロールについて記述しています。

3. CloudFormation テンプレートを起動し、セキュリティーグループおよびロールを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** は **cluster-secs** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-6d7a-13fc0b61e9db
```

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

MasterSecurityGroupId	マスターセキュリティーグループ ID
WorkerSecurityGroupId	ワーカーセキュリティーグループ ID
MasterInstanceProfile	マスター IAM インスタンスプロファイル
WorkerInstanceProfile	ワーカー IAM インスタンスプロファイル

4.12.10.1. セキュリティーオブジェクトの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なセキュリティーオブジェクトをデプロイすることができます。

例4.57 セキュリティーオブジェクトの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
```

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
```

```
      CidrIp: !Ref VpcCidr
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

MasterIngressEtcd:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: etcd
    FromPort: 2379
    ToPort: 2380
    IpProtocol: tcp

MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressWorkerVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressWorkerGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp
```

```
MasterIngressIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressWorkerIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressWorkerIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressWorkerIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50
```

```
  MasterIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressWorkerInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

  MasterIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressWorkerIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIngressWorkerIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressMasterVxlan:
```

```
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressMasterGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

WorkerIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50
```

```
WorkerIngressMasterIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

WorkerIngressMasterIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressMasterIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

WorkerIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

WorkerIngressMasterInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

WorkerIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
```

```
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal Kubernetes communication
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
```

```
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:AttachVolume"
            - "ec2:AuthorizeSecurityGroupIngress"
            - "ec2:CreateSecurityGroup"
            - "ec2:CreateTags"
            - "ec2:CreateVolume"
            - "ec2:DeleteSecurityGroup"
            - "ec2:DeleteVolume"
            - "ec2:Describe*"
            - "ec2:DetachVolume"
            - "ec2:ModifyInstanceAttribute"
            - "ec2:ModifyVolume"
            - "ec2:RevokeSecurityGroupIngress"
            - "elasticloadbalancing:AddTags"
            - "elasticloadbalancing:AttachLoadBalancerToSubnets"
            - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
            - "elasticloadbalancing:CreateListener"
            - "elasticloadbalancing:CreateLoadBalancer"
            - "elasticloadbalancing:CreateLoadBalancerPolicy"
            - "elasticloadbalancing:CreateLoadBalancerListeners"
            - "elasticloadbalancing:CreateTargetGroup"
```

```
            - "elasticloadbalancing:ConfigureHealthCheck"
            - "elasticloadbalancing:DeleteListener"
            - "elasticloadbalancing:DeleteLoadBalancer"
            - "elasticloadbalancing:DeleteLoadBalancerListeners"
            - "elasticloadbalancing:DeleteTargetGroup"
            - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
            - "elasticloadbalancing:DeregisterTargets"
            - "elasticloadbalancing:Describe*"
            - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
            - "elasticloadbalancing:ModifyListener"
            - "elasticloadbalancing:ModifyLoadBalancerAttributes"
            - "elasticloadbalancing:ModifyTargetGroup"
            - "elasticloadbalancing:ModifyTargetGroupAttributes"
            - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
            - "elasticloadbalancing:RegisterTargets"
            - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
            - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
            - "kms:DescribeKey"
            Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
```

```
MasterSecurityGroupId:
  Description: Master Security Group ID
  Value: !GetAtt MasterSecurityGroup.GroupId

WorkerSecurityGroupId:
  Description: Worker Security Group ID
  Value: !GetAtt WorkerSecurityGroup.GroupId

MasterInstanceProfile:
  Description: Master IAM Instance Profile
  Value: !Ref MasterInstanceProfile

WorkerInstanceProfile:
  Description: Worker IAM Instance Profile
  Value: !Ref WorkerInstanceProfile
```

4.12.11. ストリームメタデータを使用した RHCOS AMI へのアクセス

OpenShift Container Platform では、**ストリームメタデータ** は、JSON 形式で RHCOS に関する標準化されたメタデータを 提供し、メタデータをクラスターに挿入します。ストリームメタデータは、複数のアーキテクチャーをサポートする安定した形式で、自動化を維持するための自己文書化が意図されています。

openshift-install の **coreos print-stream-json** サブコマンドを使用して、ストリームメタデータ形式のブートイメージに関する情報にアクセスできます。このコマンドは、スクリプト可能でマシン読み取り可能な形式でストリームメタデータを出力する方法を提供します。

ユーザーによってプロビジョニングされるインストールの場合、**openshift-install** バイナリーには、AWS AMI などの OpenShift Container Platform での使用がテストされている RHCOS ブートイメージのバージョンへの参照が含まれます。

手順

ストリームメタデータを解析するには、以下のいずれかの方法を使用します。

- Go プログラムから、https://github.com/coreos/stream-metadata-go の公式の **stream-metadata-go** ライブラリーを使用します。ライブラリーでサンプルコードを確認することもできます。
- Python や Ruby などの別のプログラミング言語から、お好みのプログラミング言語の JSON ライブラリーを使用します。
- **jq** などの JSON データを処理するコマンドラインユーティリティーから、以下のコマンドを実行します。
  - **us-west-1** などの、AWS リージョンの現在の **x86_64** AMI を出力します。

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.x86_64.images.aws.regions["us-west-1"].image'
    ```

    出力例

    ```
    ami-0d3e625f84626bbda
    ```

このコマンドの出力は、**us-west-1** リージョンの AWS AMI ID です。AMI はクラスターと同じリージョンに属する必要があります。

4.12.12. AWS インフラストラクチャーの RHCOS AMI

Red Hat は、OpenShift Container Platform ノードに手動で指定できるさまざまな AWS リージョンに有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を提供します。

注記

また、独自の AMI をインポートすることで、RHCOS AMI がパブリッシュされていないリージョンにインストールすることもできます。

表4.32 RHCOS AMI

AWS ゾーン	AWS AMI
af-south-1	**ami-0ce5aa99b7d576c79**
ap-east-1	**ami-0f6debc614042ce76**
ap-northeast-1	**ami-0423a1bf292f34dc3**
ap-northeast-2	**ami-0889161041cb9d77f**
ap-northeast-3	**ami-00564b0d6cbb676b1**
ap-south-1	**ami-0650f4166d12ccead**
ap-southeast-1	**ami-0b09ad848356811c7**
ap-southeast-2	**ami-013484d0474ab5860**
ca-central-1	**ami-03291c3e2b74c32b9**
eu-central-1	**ami-0510f6f15c25b29d4**
eu-north-1	**ami-03a3119ba25eb55b1**
eu-south-1	**ami-04f719435625c1313**
eu-west-1	**ami-08e20744bd1c89c8e**
eu-west-2	**ami-0c190f5d05b071c7a**
eu-west-3	**ami-0eb0bf894fdf1d416**
me-south-1	**ami-073928aa740f738bd**

AWS ゾーン	AWS AMI
sa-east-1	**ami-01242f1bac18cc0fd**
us-east-1	**ami-05ed2cc6e70392ff9**
us-east-2	**ami-00b3a5054da356288**
us-west-1	**ami-021f626622b5238f3**
us-west-2	**ami-0c9fd8b47bfd717e8**

4.12.13. AWS でのブートストラップノードの作成

OpenShift Container Platform クラスターの初期化で使用するブートストラップノードを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、AWS リソースのスタックを作成できます。スタックは、OpenShift Container Platform インストールに必要なブートストラップノードを表します。

注記

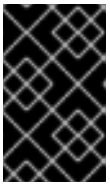
提供される CloudFormation テンプレートを使用してブートストラップノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。

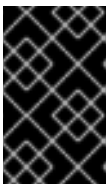
手順

1. **bootstrap.ign** Ignition 設定ファイルをクラスターに送るための場所を指定します。このファイルはインストールディレクトリーに置かれます。これを実行するための 1 つの方法として、クラスターのリージョンに S3 バケットを作成し、Ignition 設定ファイルをこれにアップロードします。



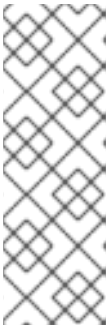
重要

提供される CloudFormation テンプレートでは、クラスターの Ignition 設定ファイルは S3 バケットから送られることを前提としています。このファイルを別の場所から送ることを選択する場合は、テンプレートを変更する必要があります。



重要

AWS SDK とは異なるエンドポイントを持つリージョンにデプロイする場合や、独自のカスタムエンドポイントを提供する場合は、**s3://** スキーマではなく、事前に署名済みの URL を S3 バケットに使用する必要があります。



注記

ブートストラップ Ignition 設定ファイルには、X.509 キーのようなシークレットが含まれません。以下の手順では、S3 バケットの基本的なセキュリティーを提供します。追加のセキュリティーを提供するには、OpenShift IAM ユーザーなどの特定のユーザーのみがバケットに含まれるオブジェクトにアクセスできるように S3 バケットポリシーを有効にできます。S3 を完全に回避し、ブートストラップマシンが到達できるアドレスからブートストラップ Ignition 設定ファイルを送ることができます。

a. バケットを作成します。

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** はバケット名です。**install-config.yaml** ファイルを作成する際に、**<cluster-name>** をクラスターに指定された名前に置き換えます。

b. **bootstrap.ign** Ignition 設定ファイルをバケットにアップロードします。

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-
infra/bootstrap.ign 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

c. ファイルがアップロードされていることを確認します。

```
$ aws s3 ls s3://<cluster-name>-infra/
```

出力例

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
```

```
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1 クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

2 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

3 ブートストラップノードに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。

4 有効な **AWS::EC2::Image::Id** 値を指定します。

5 ブートストラップノードへの SSH アクセスを許可する CIDR ブロック。

6 **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。

7 ブートストラップを起動するために VPC に関連付けられるパブリックサブネット。

8 VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

9 マスターセキュリティーグループ ID (一時ルールの登録用)。

10 セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。

11 作成されたリソースが属する VPC。

12 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

13 ブートストラップの Ignition 設定ファイルをフェッチする場所。

14 **s3://<bucket_name>/bootstrap.ign** の形式で S3 バケットおよびファイル名を指定します。

15 ネットワークロードバランサー (NLB) を登録するかどうか。

16 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

17 NLB IP ターゲット登録 lambda グループの ARN。

18 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

19 外部 API ロードバランサーのターゲットグループの ARN。

20 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

21 内部 API ロードバランサーのターゲットグループの ARN。

22 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

23 内部サービスバランサーのターゲットグループの ARN。

24 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

3. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

4. CloudFormation テンプレートを起動し、ブートストラップノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml (2)
    --parameters file://<parameters>.json (3)
    --capabilities CAPABILITY_NAMED_IAM (4)
```

(1) **<name>** は **cluster-bootstrap** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

(2) **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

(3) **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

(4) 提供されるテンプレートは一部の **AWS::IAM::Role** および **AWS::IAM::InstanceProfile** リソースを作成するため、**CAPABILITY_NAMED_IAM** 機能を明示的に宣言する必要があります。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-11eb-9dee-12dace8e3a83
```

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

BootstrapInstanceId	ブートストラップインスタンス ID。
BootstrapPublicIp	ブートストラップノードのパブリック IP アドレス。
BootstrapPrivateIp	ブートストラップノードのプライベート IP アドレス。

4.12.13.1. ブートストラップマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイできます。

例4.58 ブートストラップマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
```

```
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
```

```
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action: "ec2:Describe*"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:AttachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:DetachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "s3:GetObject"
            Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
      VpcId: !Ref VpcId

  BootstrapInstance:
    Type: AWS::EC2::Instance
    Properties:
```

```
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: "i3.large"
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "PublicSubnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
        - {
          S3Loc: !Ref BootstrapIgnitionLocation
        }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
    Description: The bootstrap node public IP address.
    Value: !GetAtt BootstrapInstance.PublicIp

  BootstrapPrivateIp:
    Description: The bootstrap node private IP address.
    Value: !GetAtt BootstrapInstance.PrivateIp
```

関連情報

関連情報

- AWS ゾーンの Red Hat Enterprise Linux CoreOS (RHCOS) AMI の詳細は、AWS インフラストラクチャーの RHCOS AMI を参照してください。

4.12.14. AWS でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Amazon Web Services (AWS) で作成する必要があります。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、コントロールプレーンノードを表す AWS リソースのスタックを作成できます。

重要

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。

注記

提供される CloudFormation テンプレートを使用してコントロールプレーンノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
```

```
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "m5.xlarge" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
```

```
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. コントロールプレーンマシンに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. DNS etcd 登録を実行するかどうか。
6. **yes** または **no** を指定します。**yes** を指定する場合、ホストゾーンの情報を指定する必要があります。
7. etcd ターゲットの登録に使用する Route 53 プライベートゾーン ID。
8. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateHostedZoneId** 値を指定します。
9. ターゲットの登録に使用する Route 53 ゾーン。
10. **<cluster_name>.<domain_name>** を指定します。ここで、**<domain_name>** はクラスターの **install-config.yaml** ファイルの生成時に使用した Route 53 ベースドメインです。AWS コンソールに表示される末尾のピリド (.) は含めないでください。
11. 13. 15. コントロールプレーンマシンの起動に使用するサブネット (プライベートが望ましい)。
12. 14. 16. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
17. コントロールプレーンノード (別名マスターノード) に関連付けるマスターセキュリティーグループ ID。
18. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。

19 コントロールプレーンの Ignition 設定ファイルをフェッチする場所。

20 生成される Ignition 設定ファイルの場所を指定します (**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)。

21 使用する base64 でエンコードされた認証局の文字列。

22 インストールディレクトリーにある **master.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

23 コントロールプレーンノードに関連付ける IAM プロファイル。

24 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **MasterInstanceProfile** パラメーターの値を指定します。

25 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

26 許可される値:

- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**
- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**

- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**

- **r5a.16xlarge**
- **r5a.24xlarge**

27 ネットワークロードバランサー (NLB) を登録するかどうか。

28 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

29 NLB IP ターゲット登録 lambda グループの ARN。

30 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

31 外部 API ロードバランサーのターゲットグループの ARN。

32 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

33 内部 API ロードバランサーのターゲットグループの ARN。

34 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

35 内部サービスバランサーのターゲットグループの ARN。

36 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。クラスターを AWS GovCloud リージョンにデプロイする場合は、**arn:aws-us-gov** を使用します。

2. このトピックの**コントロールプレーンマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。

3. **m5** インスタンスタイプを **MasterInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **MasterInstanceType.AllowedValues** パラメーターに追加します。

4. CloudFormation テンプレートを起動し、コントロールプレーンノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1. **<name>** は **cluster-control-plane** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-11eb-c6f6-0aa34627df4b
```

注記

CloudFormation テンプレートは、3 つのコントロールプレーンノードを表すスタックを作成します。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

4.12.14.1. コントロールプレーンマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例4.59 コントロールプレーンマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone information?
    Type: String
```

```
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String
    AllowedValues:
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "m5.xlarge"
    - "m5.2xlarge"
    - "m5.4xlarge"
    - "m5.8xlarge"
    - "m5.12xlarge"
    - "m5.16xlarge"
    - "m5a.xlarge"
    - "m5a.2xlarge"
    - "m5a.4xlarge"
    - "m5a.8xlarge"
    - "m5a.12xlarge"
    - "m5a.16xlarge"
    - "c4.2xlarge"
    - "c4.4xlarge"
    - "c4.8xlarge"
    - "c5.2xlarge"
```

```
    - "c5.4xlarge"
    - "c5.9xlarge"
    - "c5.12xlarge"
    - "c5.18xlarge"
    - "c5.24xlarge"
    - "c5a.2xlarge"
    - "c5a.4xlarge"
    - "c5a.8xlarge"
    - "c5a.12xlarge"
    - "c5a.16xlarge"
    - "c5a.24xlarge"
    - "r4.xlarge"
    - "r4.2xlarge"
    - "r4.4xlarge"
    - "r4.8xlarge"
    - "r4.16xlarge"
    - "r5.xlarge"
    - "r5.2xlarge"
    - "r5.4xlarge"
    - "r5.8xlarge"
    - "r5.12xlarge"
    - "r5.16xlarge"
    - "r5.24xlarge"
    - "r5a.xlarge"
    - "r5a.2xlarge"
    - "r5a.4xlarge"
    - "r5a.8xlarge"
    - "r5a.12xlarge"
    - "r5a.16xlarge"
    - "r5a.24xlarge"

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
```

```
AWS::CloudFormation::Interface:
  ParameterGroups:
  - Label:
      default: "Cluster Information"
    Parameters:
    - InfrastructureName
  - Label:
      default: "Host Information"
    Parameters:
    - MasterInstanceType
    - RhcosAmi
    - IgnitionLocation
    - CertificateAuthorities
    - MasterSecurityGroupId
    - MasterInstanceProfileName
  - Label:
      default: "Network Configuration"
    Parameters:
    - VpcId
    - AllowedBootstrapSshCidr
    - Master0Subnet
    - Master1Subnet
    - Master2Subnet
  - Label:
      default: "DNS"
    Parameters:
    - AutoRegisterDNS
    - PrivateHostedZoneName
    - PrivateHostedZoneId
  - Label:
      default: "Load Balancer Automation"
    Parameters:
    - AutoRegisterELB
    - RegisterNlbIpTargetsLambdaArn
    - ExternalApiTargetGroupArn
    - InternalApiTargetGroupArn
    - InternalServiceTargetGroupArn
  ParameterLabels:
    InfrastructureName:
      default: "Infrastructure Name"
    VpcId:
      default: "VPC ID"
    Master0Subnet:
      default: "Master-0 Subnet"
    Master1Subnet:
      default: "Master-1 Subnet"
    Master2Subnet:
      default: "Master-2 Subnet"
    MasterInstanceType:
      default: "Master Instance Type"
    MasterInstanceProfileName:
      default: "Master Instance Profile Name"
    RhcosAmi:
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
      default: "Master Ignition Source"
```

```
      CertificateAuthorities:
        default: "Ignition CA String"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterDNS:
        default: "Use Provided DNS Automation"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"
      PrivateHostedZoneName:
        default: "Private Hosted Zone Name"
      PrivateHostedZoneId:
        default: "Private Hosted Zone ID"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
  DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
```

```
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
```

```
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp
```

```
RegisterMaster2InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

EtcdSrvRecords:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join ["", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
    ResourceRecords:
    - !Join [
      " ",
      ["0 10 2380", !Join ["", ["etcd-0", !Ref PrivateHostedZoneName]]],
    ]
    - !Join [
      " ",
      ["0 10 2380", !Join ["", ["etcd-1", !Ref PrivateHostedZoneName]]],
    ]
    - !Join [
      " ",
      ["0 10 2380", !Join ["", ["etcd-2", !Ref PrivateHostedZoneName]]],
    ]
    TTL: 60
    Type: SRV

Etcd0Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join ["", ["etcd-0", !Ref PrivateHostedZoneName]]
    ResourceRecords:
    - !GetAtt Master0.PrivateIp
    TTL: 60
    Type: A

Etcd1Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join ["", ["etcd-1", !Ref PrivateHostedZoneName]]
    ResourceRecords:
    - !GetAtt Master1.PrivateIp
    TTL: 60
    Type: A

Etcd2Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
```

```
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master2.PrivateIp
      TTL: 60
      Type: A

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

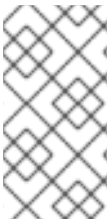
4.12.15. AWS でのワーカーノードの作成

クラスターで使用するワーカーノードを Amazon Web Services (AWS) で作成できます。

提供される CloudFormation テンプレートおよびカスタムパラメーターファイルを使用して、ワーカーノードを表す AWS リソースのスタックを作成できます。

重要

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。それぞれのワーカーノードにスタックを作成する必要があります。



注記

提供される CloudFormation テンプレートを使用してワーカーノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", (1)
    "ParameterValue": "mycluster-<random_string>" (2)
  },
  {
    "ParameterKey": "RhcosAmi", (3)
    "ParameterValue": "ami-<random_string>" (4)
  },
  {
    "ParameterKey": "Subnet", (5)
    "ParameterValue": "subnet-<random_string>" (6)
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", (7)
    "ParameterValue": "sg-<random_string>" (8)
  },
  {
    "ParameterKey": "IgnitionLocation", (9)
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker" (10)
  },
  {
    "ParameterKey": "CertificateAuthorities", (11)
    "ParameterValue": "" (12)
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", (13)
    "ParameterValue": "" (14)
  },
  {
    "ParameterKey": "WorkerInstanceType", (15)
    "ParameterValue": "m4.2xlarge" (16)
  }
]
```

(1) クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

(2) 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

(3) ワーカーノードに使用する最新の Red Hat Enterprise Linux CoreOS(RHCOS)AMI。

(4) **AWS::EC2::Image::Id** 値を指定します。

(5) ワーカーノードを起動するためのサブネット (プライベートであることが望ましい)。

(6) DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。

ターゲットを指定します。

7 ワーカーノードに関連付けるワーカーセキュリティーグループ ID。

8 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WorkerSecurityGroupId** 値を指定します。

9 ブートストラップ Ignition 設定ファイルを取得する場所。

10 生成される Ignition 設定の場所を指定します。 **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**

11 使用する base64 でエンコードされた認証局の文字列。

12 インストールディレクトリーにある **worker.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

13 ワーカーロールに関連付ける IAM プロファイル。

14 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WokerInstanceProfile** パラメーターの値を指定します。

15 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

16 許可される値:

- **m4.large**
- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **m5.large**
- **m5.xlarge**
- **m5.2xlarge**
- **m5.4xlarge**
- **m5.8xlarge**
- **m5.12xlarge**
- **m5.16xlarge**
- **m5a.large**
- **m5a.xlarge**
- **m5a.2xlarge**
- **m5a.4xlarge**

- **m5a.8xlarge**
- **m5a.12xlarge**
- **m5a.16xlarge**
- **c4.large**
- **c4.xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **c5.large**
- **c5.xlarge**
- **c5.2xlarge**
- **c5.4xlarge**
- **c5.9xlarge**
- **c5.12xlarge**
- **c5.18xlarge**
- **c5.24xlarge**
- **c5a.large**
- **c5a.xlarge**
- **c5a.2xlarge**
- **c5a.4xlarge**
- **c5a.8xlarge**
- **c5a.12xlarge**
- **c5a.16xlarge**
- **c5a.24xlarge**
- **r4.large**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**

- **r4.16xlarge**
- **r5.large**
- **r5.xlarge**
- **r5.2xlarge**
- **r5.4xlarge**
- **r5.8xlarge**
- **r5.12xlarge**
- **r5.16xlarge**
- **r5.24xlarge**
- **r5a.large**
- **r5a.xlarge**
- **r5a.2xlarge**
- **r5a.4xlarge**
- **r5a.8xlarge**
- **r5a.12xlarge**
- **r5a.16xlarge**
- **r5a.24xlarge**
- **t3.large**
- **t3.xlarge**
- **t3.2xlarge**
- **t3a.large**
- **t3a.xlarge**
- **t3a.2xlarge**

2. このトピックの**ワーカーマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークオブジェクトおよびロードバランサーについて記述しています。

3. オプション: **m5** インスタンスタイプを **WorkerInstanceType** の値として指定した場合は、そのインスタンスタイプを CloudFormation テンプレートの **WorkerInstanceType.AllowedValues** パラメーターに追加します。

4. オプション: AWS Marketplace イメージを使用してデプロイする場合は、サブスクリプションから取得した AMI ID で **Worker0.type.properties.ImageID** パラメーターを更新します。

5. CloudFormation テンプレートを起動し、ワーカーノードを表す AWS リソースのスタックを作成します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-worker-1** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

出力例

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-11eb-348f-sd9888c65b59
```

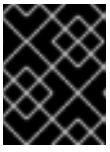
注記

CloudFormation テンプレートは、1 つのワーカーノードを表すスタックを作成します。

6. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. クラスターに作成するワーカーマシンが十分な数に達するまでワーカースタックの作成を継続します。同じテンプレートおよびパラメーターファイルを参照し、異なるスタック名を指定してワーカースタックをさらに作成することができます。



重要

2 つ以上のワーカーマシンを作成する必要があるため、この CloudFormation テンプレートを使用する 2 つ以上のスタックを作成する必要があります。

4.12.15.1. ワーカーマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例4.60 ワーカーマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String
    AllowedValues:
    - "m4.large"
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "m5.large"
    - "m5.xlarge"
    - "m5.2xlarge"
    - "m5.4xlarge"
    - "m5.8xlarge"
    - "m5.12xlarge"
    - "m5.16xlarge"
    - "m5a.large"
    - "m5a.xlarge"
    - "m5a.2xlarge"
    - "m5a.4xlarge"
    - "m5a.8xlarge"
    - "m5a.12xlarge"
    - "m5a.16xlarge"
    - "c4.large"
```

```
    - "c4.xlarge"
    - "c4.2xlarge"
    - "c4.4xlarge"
    - "c4.8xlarge"
    - "c5.large"
    - "c5.xlarge"
    - "c5.2xlarge"
    - "c5.4xlarge"
    - "c5.9xlarge"
    - "c5.12xlarge"
    - "c5.18xlarge"
    - "c5.24xlarge"
    - "c5a.large"
    - "c5a.xlarge"
    - "c5a.2xlarge"
    - "c5a.4xlarge"
    - "c5a.8xlarge"
    - "c5a.12xlarge"
    - "c5a.16xlarge"
    - "c5a.24xlarge"
    - "r4.large"
    - "r4.xlarge"
    - "r4.2xlarge"
    - "r4.4xlarge"
    - "r4.8xlarge"
    - "r4.16xlarge"
    - "r5.large"
    - "r5.xlarge"
    - "r5.2xlarge"
    - "r5.4xlarge"
    - "r5.8xlarge"
    - "r5.12xlarge"
    - "r5.16xlarge"
    - "r5.24xlarge"
    - "r5a.large"
    - "r5a.xlarge"
    - "r5a.2xlarge"
    - "r5a.4xlarge"
    - "r5a.8xlarge"
    - "r5a.12xlarge"
    - "r5a.16xlarge"
    - "r5a.24xlarge"
    - "t3.large"
    - "t3.xlarge"
    - "t3.2xlarge"
    - "t3a.large"
    - "t3a.xlarge"
    - "t3a.2xlarge"

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
```

```
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
```

```
      }
    Tags:
    - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

4.12.16. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS でのブートストラップシーケンスの初期化

Amazon Web Services (AWS) ですべての必要なインフラストラクチャーを作成した後に、OpenShift Container Platform コントロールプレーンを初期化するブートストラップシーケンスを開始できます。

前提条件

- AWS アカウントを設定している。
- **aws configure** を実行して、AWS キーおよびリージョンをローカルの AWS プロファイルに追加している。
- クラスターの Ignition 設定ファイルを生成している。
- AWS で VPC および関連するサブネットを作成し、設定している。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定している。
- AWS でクラスターに必要なセキュリティーグループおよびロールを作成している。
- ブートストラップマシンを作成している。
- コントロールプレーンマシンを作成している。
- ワーカーノードを作成している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、OpenShift Container Platform コントロールプレーンを初期化するブートストラッププロセスを開始します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

-

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.19.0+9f84db3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

コマンドが **FATAL** 警告を出さずに終了する場合、OpenShift Container Platform コントロールプレーンは初期化されています。

注記

コントロールプレーンの初期化後に、コンピュートノードを設定し、Operator の形式で追加のサービスをインストールします。

関連情報

- OpenShift Container Platform インストールの進捗としてインストール、ブートストラップ、およびコントロールプレーンのログをモニターリングする方法についての詳細は、インストールの進捗のモニターリング を参照してください。
- ブートストラッププロセスに関する問題のトラブルシューティングの詳細は、ブートストラップノードの診断データの収集 を参照してください。

4.12.17. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.12.18. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

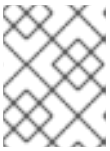
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

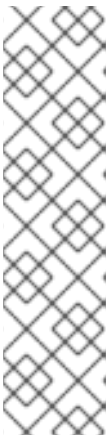
```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

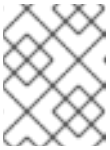
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

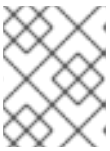
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

4.12.19. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

4.12.19.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

4.12.19.2. イメージレジストリーストレージの設定

Amazon Web Services はデフォルトのストレージを提供します。つまり、Image Registry Operator はインストール後に利用可能になります。ただし、レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、レジストリーストレージを手動で設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

4.12.19.2.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS 上にクラスターがある。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

4.12.19.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

4.12.20. ブートストラップリソースの削除

クラスターの初期 Operator 設定の完了後に、Amazon Web Services (AWS) からブートストラップリソースを削除します。

前提条件

- クラスターの初期 Operator 設定が完了済みです。

手順

1. ブートストラップリソースを削除します。CloudFormation テンプレートを使用した場合は、そのスタックを削除 します。
   - AWS CLI を使用してスタックを削除します。

     ```
     $ aws cloudformation delete-stack --stack-name <name> 1
     ```

     1 **<name>** は、ブートストラップスタックの名前です。

   - AWS CloudFormation コンソール を使用してスタックを削除します。

4.12.21. Ingress DNS レコードの作成

DNS ゾーン設定を削除した場合には、Ingress ロードバランサーを参照する DNS レコードを手動で作成します。ワイルドカードレコードまたは特定のレコードのいずれかを作成できます。以下の手順では A レコードを使用しますが、CNAME やエイリアスなどの必要な他のレコードタイプを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用する OpenShift Container Platform クラスターを Amazon Web Services (AWS) にデプロイしています。
- OpenShift CLI (**oc**) がインストールされている。
- **jq** パッケージをインストールしている。
- AWS CLI をダウンロードし、これをコンピューターにインストールしている。Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。

手順

1. 作成するルートを決定します。
   - ワイルドカードレコードを作成するには、***.apps.<cluster_name>.<domain_name>** を使用します。ここで、**<cluster_name>** はクラスター名で、**<domain_name>** は OpenShift Container Platform クラスターの Route 53 ベースドメインです。
   - 特定のレコードを作成するには、以下のコマンドの出力にあるように、クラスターが使用する各ルートにレコードを作成する必要があります。

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     出力例

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     ```

```
alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
```

2. Ingress Operator ロードバランサーのステータスを取得し、使用する外部 IP アドレスの値をメモします。これは **EXTERNAL-IP** 列に表示されます。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                         PORT(S)
AGE
router-default  LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
80:31499/TCP,443:30693/TCP   5m
```

3. ロードバランサーのホストゾーン ID を見つけます。

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
"<external_ip>").CanonicalHostedZoneNameID' 1
```

1 **<external_ip>** については、取得した Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。

出力例

```
Z3AADJGX6KTTL2
```

このコマンドの出力は、ロードバランサーのホストゾーン ID です。

4. クラスターのドメインのパブリックホストゾーン ID を取得します。

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>** については、OpenShift Container Platform クラスターの Route 53 ベースドメインを指定します。

出力例

```
/hostedzone/Z3URY6TWQ91KVV
```

ドメインのパブリックホストゾーン ID がコマンド出力に表示されます。この例では、これは **Z3URY6TWQ91KVV** になります。

5. プライベートゾーンにエイリアスレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
```

```
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

(1) **<private_hosted_zone_id>** については、DNS および負荷分散の CloudFormation テンプレートの出力から値を指定します。

(2) **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。

(3) **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

(4) **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6. パブリックゾーンにレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

(1) **<public_hosted_zone_id>** については、ドメインのパブリックホストゾーンを指定します。

(2) **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します

エイリアスまたはサブドメインを指定します。

3 **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

4 **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

4.12.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行

Amazon Web Service (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後に、デプロイメントを完了するまでモニターします。

前提条件

- OpenShift Container Platform クラスターのブートストラップノードを、ユーザーによってプロビジョニングされた AWS インフラストラクチャーで削除している。
- **oc** CLI をインストールしていること。

手順

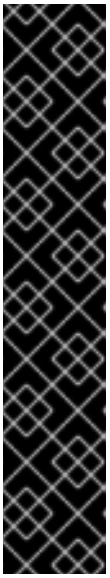
1. インストールプログラムが含まれるディレクトリーから、クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Fe5en-ymBEc-
Wt6NL"
INFO Time elapsed: 1s
```



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Cluster registration ページでクラスターを登録します。

4.12.23. Web コンソールを使用したクラスターへのログイン

kubeadmin ユーザーは、OpenShift Container Platform のインストール後はデフォルトで存在します。OpenShift Container Platform Web コンソールを使用し、**kubeadmin** ユーザーとしてクラスターにログインできます。

前提条件

- インストールホストにアクセスできる。
- クラスターのインストールを完了しており、すべてのクラスター Operator が利用可能である。

手順

1. インストールホストで **kubeadmin-password** ファイルから **kubeadmin** ユーザーのパスワードを取得します。

```
$ cat <installation_directory>/auth/kubeadmin-password
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルから **kubeadmin** パスワードを取得できます。

2. OpenShift Container Platform Web コンソールルートを一覧表示します。

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

注記

または、インストールホストで **<installation_directory>/.openshift_install.log** ログファイルからで OpenShift Container Platform ルートを取得できます。

出力例

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. Web ブラウザーで前述のコマンドの出力で詳細に説明されたルートに移動し、**kubeadmin** ユーザーとしてログインします。

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

4.12.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

4.12.25. 関連情報

- AWS CloudFormation スタックについての詳細は、Working with stacks を参照してください。

4.12.26. 次のステップ

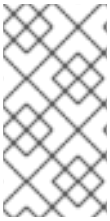
- インストールを検証 します。
- クラスターをカスタマイズします。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

4.13. AWS でのクラスターのアンインストール

Amazon Web Services (AWS) にデプロイしたクラスターは削除することができます。

4.13.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

4.13.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの削除

STS を使用し、手動モードで Cloud Credential Operator (CCO) を使用して OpenShift Container Platform クラスターをアンインストールした後にリソースをクリーンアップするには、CCO ユーティリティー (**ccoctl**) を使用してインストール時に **ccoctl** が作成した AWS リソースを削除します。

前提条件

- **ccoctl** バイナリーを展開して準備します。

- STS を使用し、手動モードで CCO を使用して OpenShift Container Platform クラスターをインストールします。

手順

- **ccoctl** が作成した AWS リソースを削除します。

```
$ ccoctl aws delete --name=<name> --region=<aws_region>
```

ここでは、以下のようになります。

- **<name>** は、クラウドリソースを最初に作成してタグ付けするために使用された名前と一致します。
- **<aws-region>** は、クラウドリソースが削除される AWS リージョンです。

出力例:

```
2021/04/08 17:50:41 Identity Provider object .well-known/openid-configuration deleted from the bucket <name>-oidc
2021/04/08 17:50:42 Identity Provider object keys.json deleted from the bucket <name>-oidc
2021/04/08 17:50:43 Identity Provider bucket <name>-oidc deleted
2021/04/08 17:51:05 Policy <name>-openshift-cloud-credential-operator-cloud-credential-o associated with IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-o deleted
2021/04/08 17:51:05 IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-o deleted
2021/04/08 17:51:07 Policy <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials associated with IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials deleted
2021/04/08 17:51:07 IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials deleted
2021/04/08 17:51:08 Policy <name>-openshift-image-registry-installer-cloud-credentials associated with IAM Role <name>-openshift-image-registry-installer-cloud-credentials deleted
2021/04/08 17:51:08 IAM Role <name>-openshift-image-registry-installer-cloud-credentials deleted
2021/04/08 17:51:09 Policy <name>-openshift-ingress-operator-cloud-credentials associated with IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
2021/04/08 17:51:10 IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
2021/04/08 17:51:11 Policy <name>-openshift-machine-api-aws-cloud-credentials associated with IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
2021/04/08 17:51:11 IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
2021/04/08 17:51:39 Identity Provider with ARN arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com deleted
```

検証

AWS にクエリーを実行すると、リソースが削除されていることを確認できます。詳細は AWS ドキュメントを参照してください。

第5章 AZURE へのインストール

5.1. AZURE へのインストールの準備

5.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

5.1.2. OpenShift Container Platform の Azure へのインストール要件

OpenShift Container Platform を Microsoft Azure にインストールする前に、Azure アカウントを設定する必要があります。アカウントの設定、アカウントの制限、パブリック DNS ゾーン設定、必要なロール、サービスプリンシパルの作成、およびサポートされる Azure リージョンの詳細は、Azure アカウントの設定 を参照してください。

クラウドアイデンティティーおよびアクセス管理 (IAM) API がお使いの環境からアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、他のオプションについて、Azure の IAM の手動作成 を参照してください。

5.1.3. Azure に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

5.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる Azure インフラストラクチャーに、クラスターをインストールできます。

- **クラスターの Azure へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる Azure インフラストラクチャーに OpenShift Container Platform をインストールできます。デフォルトの設定オプションを使用して、クラスターを迅速にインストールできます。
- **カスタマイズされたクラスターの Azure へのインストール**: インストールプログラムがプロビジョニングする Azure インフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の数多くのカスタマイズオプションは、インストール後 に利用できます。

- **ネットワークのカスタマイズを使用したクラスターの Azure へのインストール**: インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **Azure の既存 VNet へのクラスターのインストール**: OpenShift Container Platform を Azure の既存の Azure Virtual Network (VNet) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **プライベートクラスターの Azure へのインストール**: プライベートクラスターを Azure の既存の Azure Virtual Network (VNet) にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

- **Azure の government リージョンへのクラスターのインストール**: OpenShift Container Platform は、機密ワークロードを Azure で実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されている Microsoft Azure Government (MAG) リージョンにデプロイできます。

5.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法を使用して、独自にプロビジョニングする Azure インフラストラクチャーにクラスターをインストールできます。

- **ARM テンプレートを使用したクラスターの Azure へのインストール**: 独自に提供するインフラストラクチャーを使用して、OpenShift Container Platform を Azure にインストールできます。提供される Azure Resource Manager (ARM) テンプレートを使用して、インストールを支援できます。

5.1.4. 次のステップ

- Azure アカウントの設定

5.2. AZURE アカウントの設定

OpenShift Container Platform をインストールする前に、Microsoft Azure アカウントを設定する必要があります。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

5.2.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。

重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	40	リージョンごとに 20	デフォルトのクラスターには 40 の vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップマシンは 4 vCPUS を使用する **Standard_D4s_v3** マシンを使用し、コントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、さらにワーカーマシンは、4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、デフォルトクラスターには 40 の vCPU が必要になります。4 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、または異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。 デフォルトで、インストールプログラムはコントロールプレーンおよびコンピュートマシンを、リージョン 内の すべてのアベイラビリティーゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
OS ディスク	7		仮想マシン OS ディスクは、最小スループットが 5000 IOPS/200MBps を維持できる必要があります。このスループットは、P30 (最低 1 TiB Premium SSD) を使用することで実現できます。Azure では、ディスクのパフォーマンスは SSD のディスクサイズに直接依存するため、利用可能な **Standard_D8s_v3** またはその他の同様のマシンタイプでサポートされるスループット (ターゲット: 5000 IOPS) を実現するには、少なくとも P30 ディスクが必要です。 読み取りのレイテンシーが低く抑え、読み取り IOPS およびスループットが高く保つには、ホストのキャッシュを **ReadOnly** に設定する必要があります。仮想マシンメモリーまたはローカル SSD ディスクにあるキャッシュから実行された読み取りは、Blob ストレージにあるデータディスクからの読み取りよりもはるかに高速です。
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	6	リージョンごとに 65,536	各デフォルトクラスターには、6 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	5000	各デフォルトクラスター。各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。
スポット VM vCPU (オプション)	0 スポット VM を設定する場合には、クラスターのコンピュートノードごとにスポット VM vCPU が 2 つ必要です。	リージョンごとに 20	これはオプションのコンポーネントです。スポット VM を使用するには、Azure の既定の制限を最低でも、クラスター内のコンピュートノード数の 2 倍に増やす必要があります。 **注記** コントロールプレーンノードにスポット VM を使用することはお勧めしません。

5.2.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。

注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、 新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

5.2.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。

注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。

2. **New support request** をクリックしてから必要な値を選択します。

   a. **Issue type** 一覧から、**Service and subscription limits (quotas)** を選択します。

   b. **Subscription** 一覧から、変更するサブスクリプションを選択します。

   c. **Quota type** 一覧から、引き上げるクォータを選択します。たとえば、**Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。

   d. **Next: Solutions** をクリックします。

3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。

a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。

b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。

4. **Next: Review + create** をクリックしてから **Create** をクリックします。

5.2.4. 必要な Azure ロール

OpenShift Container Platform には、Microsoft Azure リソースを管理できるようにサービスプリンシパルが必要です。サービスプリンシパルの作成前に、Azure アカウントサブスクリプションに次のロールが必要です。

- **User Access Administrator**
- **Owner**

Azure ポータルでロールを設定するには、Azure ドキュメントの Manage access to Azure resources using RBAC and the Azure portal を参照します。

5.2.5. サービスプリンシパルの作成

OpenShift Container Platform およびそのインストールプログラムは Azure Resource Manager 経由で Microsoft Azure リソースを作成する必要があるため、これを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- **jq** パッケージをインストールします。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。

手順

1. Azure CLI にログインします。

```
$ az login
```

認証情報を使用して Web コンソールで Azure にログインします。

2. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。

a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": "AzureCloud",
```

```
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** パラメーターの値が適切なサブスクリプションの UUID であることを確認します。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <id> (1)
```

(1) 使用する必要のあるサブスクリプションの **id** の値を **<id>** の代わりに使用します。

d. アクティブなサブスクリプションを変更したら、アカウント情報を再度表示します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
```

```
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 直前の出力の **tenantId** および **id** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

4. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
```

1 **<service_principal>** を、サービスプリンシパルに割り当てる名前に置き換えます。

出力例

```
Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the
required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
Retrying role assignment creation: 4/36
{
  "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
  "displayName": "<service_principal>",
  "name": "http://<service_principal>",
  "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
}
```

5. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

6. サービスプリンシパルに追加パーミッションを付与します。

   - クラスターはそのコンポーネントの認証情報を割り当てできるように、**Contributor** および **User Access Administrator** ロールを常にアプリケーション登録サービスプリンシパルに追加する必要があります。

   - Cloud Credential Operator (CCO) を **mint モード** で操作するには、アプリケーション登録サービスプリンシパルで **Azure Active Directory Graph/Application.ReadWrite.OwnedBy** API パーミッションも必要です。

   - CCO を **passthrough モード** で操作するには、アプリケーション登録サービスプリンシパルで追加の API パーミッションは必要ありません。

   CCO モードの詳細は、**認証および承認**ガイドのクラウドプロバイダークレデンシャルの管理セクションにある Cloud Credential Operator についてを参照してください。

注記

OpenShift Container Platform インストールプログラムのサービスプリンシパルの範囲を既存の Azure リソースグループに制限する場合は、環境内のインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。

a. **User Access Administrator** ロールを割り当てるには、以下のコマンドを実行します。

```
$ az role assignment create --role "User Access Administrator" \
    --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \
      | jq '.[0].id' -r) 1
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

b. **Azure Active Directory Graph** パーミッションを割り当てるには、以下のコマンドを実行します。

```
$ az ad app permission add --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000 \
    --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

出力例

```
Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api
00000002-0000-0000-c000-000000000000" is needed to make the change effective
```

このコマンドで付与する特定のパーミッションについての詳細は、GUID Table for Windows Azure Active Directory Permissions を参照してください。

c. パーミッション要求を承認します。アカウントに Azure Active Directory テナント管理者ロールがない場合は、所属する組織のガイドラインに従い、テナント管理者にパーミッション要求を承認するようにリクエストしてください。

```
$ az ad app permission grant --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

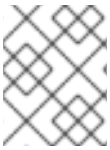
関連情報

- CCO モードの詳細は、Cloud Crednetial Operator について を参照してください。

5.2.6. サポートされている Azure Marketplace リージョン

北米、ヨーロッパ、中東およびアフリカでこのサービスを購入されたお客様は、Azure Marketplace イメージを使用してクラスターをインストールできます。

このオファーは北米または EMEA で購入する必要がありますが、OpenShift Container Platform がサポートする任意の Azure パブリックパーティションにクラスターをデプロイできます。



注記

Azure Marketplace イメージを使用したクラスターのデプロイは、Azure Government リージョンではサポートされません。

5.2.7. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンの一覧を動的に生成します。

サポート対象の Azure パブリックリージョン

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **southafricanorth** (South Africa North)

- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

サポート対象の Azure Government リージョン
以下の Microsoft Azure Government (MAG) リージョンのサポートが OpenShift Container Platform バージョン 4.6 に追加されています。

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

Azure ドキュメント の利用可能なすべての MAG リージョンを参照できます。他の提供される MAG リージョンは OpenShift Container Platform で機能することが予想されますが、まだテストされていません。

5.2.8. 次のステップ

- OpenShift Container Platform クラスターを Azure にインストールします。カスタマイズされたクラスターのインストール、またはデフォルトのオプションで クラスターのクイックインストール を実行できます。

5.3. AZURE の IAM の手動作成

クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境や、管理者がクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存する選択をしない場合に、クラスターのインストール前に Cloud Credential Operator (CCO) を手動モードにすることができます。

5.3.1. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。**credentialsMode** パラメーターの異なる値を **install-config.yaml** ファイルに設定し、組織のセキュリティー要件に応じて CCO を設定できます。

管理者レベルの認証情報シークレットをクラスターの **kube-system** プロジェクトに保存する選択をしない場合、OpenShift Container Platform をインストールし、クラウド認証情報を手動で管理する際に CCO の **credentialsMode** パラメーターを **Manual** に設定できます。

手動モードを使用すると、クラスターに管理者レベルの認証情報を保存する必要なく、各クラスターコンポーネントに必要なパーミッションのみを指定できます。お使いの環境でクラウドプロバイダーのパブリック IAM エンドポイントへの接続がない場合も、このモードを使用できます。ただし、各アップグレードについて、パーミッションを新規リリースイメージを使用して手動で調整する必要があります。また、それらを要求するすべてのコンポーネントについて認証情報を手動で指定する必要があります。

関連情報

- 利用可能なすべての CCO 認証情報モードとそれらのサポートされるプラットフォームの詳細については、Cloud Credential Operator について 参照してください。

5.3.2. IAM の手動作成

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、**install-config.yaml** ファイルを作成します。

```
$ openshift-install create install-config --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

3. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ openshift-install create manifests --dir <installation_directory>
```

4. インストールプログラムが含まれるディレクトリーから、**openshift-install** バイナリーがビルドされる OpenShift Container Platform リリースイメージの詳細を取得します。

```
$ openshift-install version
```

出力例

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. このリリースイメージ内で、デプロイするクラウドをターゲットとする **CredentialsRequest** オブジェクトをすべて特定します。

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 --
credentials-requests --cloud=azure
```

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

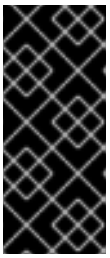
サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-azure
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

7. インストールプログラムが含まれるディレクトリーから、クラスターの作成に進みます。

```
$ openshift-install create cluster --dir <installation_directory>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。詳細は、クラウドプロバイダーのインストールコンテンツの手動でメンテナンスされる認証情報を使用したクラスターのアップグレードについてのセクションを参照してください。

5.3.3. 手動でメンテナンスされた認証情報を使用したクラスターのアップグレード

手動でメンテナンスされる認証情報をを含むクラスターの Cloud Credential Operator (CCO) の **upgradable** ステータスはデフォルトで **false** となります。

- 4.7 から 4.8 などのマイナーリリースの場合には、このステータスを使用することで、パーミッションを更新して **CloudCredential** リソースにアノテーションを付けてパーミッションが次のバージョンの要件に合わせて更新されていることを指定するまで、アップグレードができないようになります。このアノテーションは、**Upgradable** ステータスを **True** に変更します。

- 4.8.9 から 4.8.10 などの z-stream リリースの場合には、パーミッションは追加または変更されないため、アップグレードはブロックされません。

手動でメンテナンスされる認証情報でクラスターをアップグレードする前に、アップグレードするリリースイメージ用に認証情報を新規作成する必要があります。さらに、既存の認証情報に必要なパーミッションを確認し、これらのコンポーネントの新規リリースの新しいパーミッション要件に対応する必要があります。

手順

1. 新規リリースの **CredentialsRequest** カスタムリソースを抽出して検査します。
クラウドプロバイダーのインストールコンテンツの IAM の手動作成についてのセクションでは、クラウドに必要な認証情報を取得し、使用する方法について説明します。

2. クラスターで手動でメンテナンスされる認証情報を更新します。

   - 新規リリースイメージによって追加される **CredentialsRequest** カスタムリソースの新規のシークレットを作成します。

   - シークレットに保存される既存の認証情報の **CredentialsRequest** カスタムリソースにパーミッション要件を変更した場合は、必要に応じてパーミッションを更新します。

3. 新規リリースですべてのシークレットが正しい場合は、クラスターをアップグレードする準備が整っていることを示します。

   a. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

   b. **CloudCredential** リソースを編集して、**metadata** フィールドに **upgradeable-to** アノテーションを追加します。

   ```
   $ oc edit cloudcredential cluster
   ```

   追加するテキスト

   ```
   ...
     metadata:
       annotations:
         cloudcredential.openshift.io/upgradeable-to: <version_number>
   ...
   ```

   ここで、**<version_number>** は、アップグレードするバージョン (**x.y.z** 形式) に置き換えます。例: OpenShift Container Platform **4.8.2** (OpenShift Container Platform 4.8.2 の場合)

   アノテーションを追加してから、upgradeable のステータスが変更されるまで、数分かかる場合があります。

4. CCO がアップグレードできることを確認します。

a. Web コンソールの **Administrator** パースペクティブで、**Administration → Cluster Settings** に移動します。

b. CCO ステータスの詳細を表示するには、**Cluster Operators** 一覧で **cloud-credential** をクリックします。

c. **Conditions** セクションの **Upgradeable** ステータスが **False** の場合に、**upgradeable-to** アノテーションに間違いがないことを確認します。

Conditions セクションの **Upgradeable** ステータスが **True** の場合には、OpenShift Container Platform のアップグレードを開始できます。

5.3.4. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトオプションを使用した クラスターの Azure へのクイックインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのクラウドのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール

5.4. クラスターの AZURE へのクイックインストール

OpenShift Container Platform バージョン 4.8 では、デフォルトの設定オプションを使用する Microsoft Azure にクラスターをインストールできます。

5.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

5.4.2. OpenShift Container Platform のインターネットアクセス

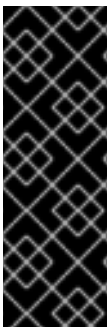
OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブス

クリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.4.3. クラスターノードの SSH アクセス用のキーペアの生成

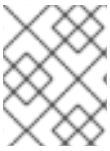
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> (1)
```

(1) 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

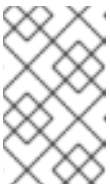
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.4.4. インストールプログラムの取得

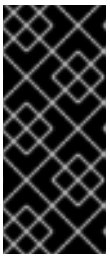
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

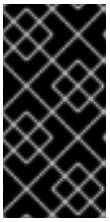
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.4.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

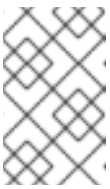
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **azure** を選択します。

c. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

d. クラスターをデプロイするリージョンを選択します。

e. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

f. クラスターの記述名を入力します。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

g. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

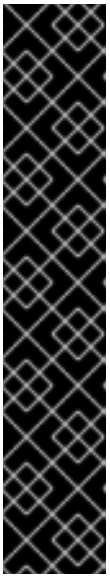
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

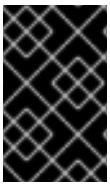
- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラ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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

5.4.6. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.4.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

5.4.8. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

5.4.9. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

5.5. カスタマイズによる AZURE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、インストールプログラムが Microsoft Azure にプロビジョニングするインフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

5.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

5.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.5.3. クラスターノードの SSH アクセス用のキーペアの生成

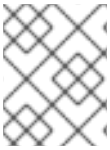
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.5.4. Azure Marketplace イメージの選択

Azure Marketplace オファリングを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に Azure Marketplace イメージを取得する必要があります。インストールプログラムは、このイメージを使用してワーカーノードをデプロイします。イメージを取得するときは、次の点を考慮してください。

- イメージは同じですが、Azure Marketplace のパブリシャーは地域によって異なります。北米にお住まいの場合は、**redhat** をパブリッシャーとして指定してください。EMEA にお住まいの場合は、**redhat-limited** をパブリッシャーとして指定してください。

- このオファーには、**rh-ocp-worker** SKU と **rh-ocp-worker-gen1** SKU が含まれています。**rh-ocp-worker** SKU は、Hyper-V 世代のバージョン 2 VM イメージを表します。OpenShift Container Platform で使用されるデフォルトのインスタンスタイプは、バージョン 2 と互換性があります。バージョン 1 のみと互換性のあるインスタンスタイプを使用する場合は、**rh-ocp-worker-gen1** SKU に関連付けられたイメージを使用します。**rh-ocp-worker-gen1** SKU は、Hyper-V バージョン 1 VM イメージを表します。

前提条件

- Azure CLI クライアント **(az)** をインストールしている。
- お客様の Azure アカウントにはオファーのエンタイトルメントがあり、Azure CLI クライアントを使用してこのアカウントにログインしている。

手順

1. 以下のいずれかのコマンドを実行して、利用可能なすべての OpenShift Container Platform イメージを表示します。
   - 北米:

     ```
     $ az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
     ```

     出力例

     ```
     Offer          Publisher      Sku                Urn                                                                  Version
     -------------  -------------  -----------------  -------------------------------------------------------------------  --------------
     rh-ocp-worker  RedHat         rh-ocp-worker      RedHat:rh-ocp-worker:rh-ocpworker:4.8.2021122100                     4.8.2021122100
     rh-ocp-worker  RedHat         rh-ocp-worker-gen1 RedHat:rh-ocp-worker:rh-ocp-worker-gen1:4.8.2021122100       4.8.2021122100
     ```

   - EMEA:

     ```
     $ az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
     ```

     出力例

     ```
     Offer          Publisher      Sku                Urn                                                                  Version
     -------------  -------------  -----------------  -------------------------------------------------------------------  --------------
     rh-ocp-worker  redhat-limited  rh-ocp-worker      redhat-limited:rh-ocp-worker:rh-ocp-worker:4.8.2021122100      4.8.2021122100
     rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1 redhat-limited:rh-ocp-worker:rh-ocp-worker-gen1:4.8.2021122100 4.8.2021122100
     ```

注記

インストールする OpenShift Container Platform のバージョンに関係なく、使用する Azure Marketplace イメージの正しいバージョンは 4.8.x です。必要に応じて、インストールプロセスの一環として、VM が自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファーのイメージを調べます。
   - 北米:

     ```
     $ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

3. 次のコマンドのいずれかを実行して、オファーの条件を確認します。
   - 北米:

     ```
     $ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

4. 次のコマンドのいずれかを実行して、オファリングの条件に同意します。
   - 北米:

     ```
     $ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

5. オファーのイメージの詳細を記録します。インストールを完了する前に、Updating Manifests for Marketplace Installation というセクションの **99_openshift-cluster-api_worker-machineset-[0-2].yaml** ファイルを更新する必要があります。

5.5.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

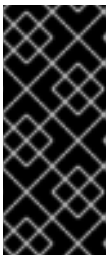
前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。

3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.5.6. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

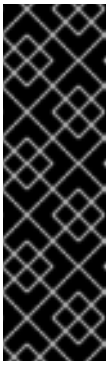
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

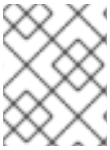
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

5.5.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.5.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.1 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.5.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.2 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.5.6.1.3. オプションの設定パラメーター

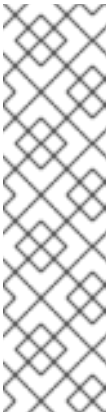
オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.3 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

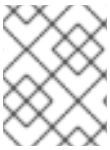
5.5.6.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表5.4 追加の Azure パラメーター

パラメーター	説明	値
compute.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**standard_LRS**、**premium_LRS**、または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.resourceGroupName	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。

パラメーター	説明	値
platform.azure.outboundType	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。	**LoadBalancer** または **UserDefinedRouting**。デフォルトは **LoadBalancer** です。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.cloudName	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。空の場合、デフォルト値の **AzurePublicCloud** が使用されます。	**AzurePublicCloud** または **AzureUSGovernmentCloud** などの有効なクラウド環境。



注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

5.5.6.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group (11)
    region: centralus (12)
    resourceGroupName: existing_resource_group (13)
    outboundType: Loadbalancer
```

```
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
```

1 10 12 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノード (別名マスターノード) の最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

11 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

13 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

5.5.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を

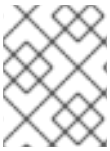
4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.5.7. Marketplace インストールのマニフェストの更新

インストール用に Marketplace イメージを選択している場合は、Marketplace イメージを使用するようにマニフェストを作成および変更する必要があります。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. 以下のコマンドを実行して、インストールプログラムが含まれるディレクトリーに移動して、マニフェストを作成します。

```
$ openshift-install create manifests --dir <installation_dir>
```

2. コンピュートマシンセット定義の **.spec.template.spec.providerSpec.value.image** プロパティーを編集し、**offer**、**publisher**、**sku**、および **version** の値を、Selecting an Azure Marketplace image というセクションで収集された情報に置き換えます。更新する必要がある 3 つのファイルを以下に示します。

   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-0.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-1.yaml**

- **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-2.yaml**

3. 各ファイルで、**.spec.template.spec.providerSpec.value.image.resourceID** プロパティーの値は、空の値 ("") に置き換えます。

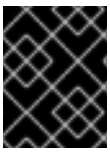
4. 各ファイルで、**type** プロパティーを **MarketplaceWithPlan** に設定します。

5. 最初のマシンセットファイルをサンプルとして使用すると、**.spec.template.spec.providerSpec.value.image** セクションは以下の例のようになります。

```
image:
  offer: rh-ocp-worker
  publisher: redhat
  resourceID: ""
  sku: rh-ocp-worker
  version: 4.8.2021122100
  type: MarketplaceWithPlan
```

5.5.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

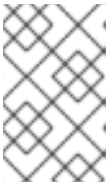
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

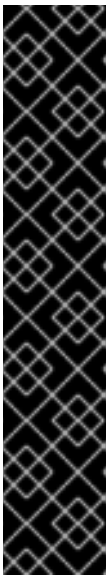
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

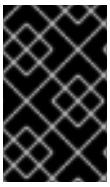
- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラ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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

5.5.9. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.5.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

5.5.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

5.5.12. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

5.6. ネットワークのカスタマイズによる AZURE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、インストールプログラムが Microsoft Azure にプロビジョニングするインフラストラクチャーにカスタマイズされたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

5.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

5.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

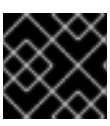
クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.6.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

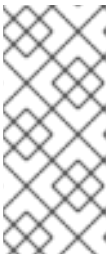
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

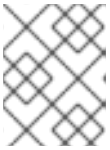
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

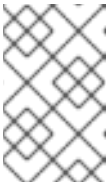
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.6.4. インストールプログラムの取得

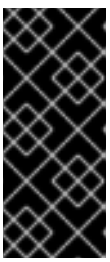
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.6.5. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

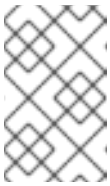
1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

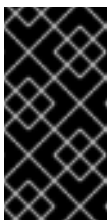
iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

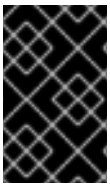
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

5.6.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.6.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.5 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト

パラメーター	説明	値
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.6.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.6 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.6.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.7 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。

パラメーター	説明	値
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

5.6.5.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表5.8 追加の Azure パラメーター

パラメーター	説明	値
compute.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**standard_LRS**、**premium_LRS**、または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。

パラメーター	説明	値
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.resourceGroupName	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。
platform.azure.outboundType	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。	**LoadBalancer** または **UserDefinedRouting**。デフォルトは **LoadBalancer** です。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。

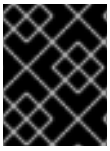
パラメーター	説明	値
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.cloudName	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。空の場合、デフォルト値の **AzurePublicCloud** が使用されます。	**AzurePublicCloud** または **AzureUSGovernmentCloud** などの有効なクラウド環境。

注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

5.6.5.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
```

```
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group 12
    region: centralus 13
    resourceGroupName: existing_resource_group 14
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 15
fips: false 16
sshKey: ssh-ed25519 AAAA... 17
```

1 10 13 15 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 11 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノード (別名マスターノード) の最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

12 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

14 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

16 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

17 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

5.6.5.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

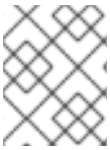
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

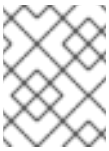
5.6.6. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

5.6.7. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

5.6.8. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

5.6.8.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表5.9 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。
		
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		
		マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表5.10 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定

以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表5.11 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
```

```
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定
以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表5.12 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 暗号化を有効にするために空のオブジェクトを指定します。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表5.13 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。

フィールド	タイプ	説明
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表5.14 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

5.6.9. OVN-Kubernetes を使用したハイブリッドネットワークの設定

OVN-Kubernetes でハイブリッドネットワークを使用するようにクラスターを設定できます。これにより、異なるノードのネットワーク設定をサポートするハイブリッドクラスターが可能になります。たとえば、これはクラスター内の Linux ノードと Windows ノードの両方を実行するために必要です。

重要

クラスターのインストール時に、OVN-Kubernetes を使用してハイブリッドネットワークを設定する必要があります。インストールプロセス後に、ハイブリッドネットワークに切り替えることはできません。

前提条件

- **install-config.yaml** ファイルで **networking.networkType** パラメーターの **OVNKubernetes** を定義していること。詳細は、選択したクラウドプロバイダーでの OpenShift Container Platform ネットワークのカスタマイズの設定についてのインストールドキュメントを参照してください。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

ここでは、以下のようになります。

<installation_directory>

クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
```

```
metadata:
  name: cluster
spec:
EOF
```

ここでは、以下のようになります。

<installation_directory>

クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

3. **cluster-network-03-config.yml** ファイルをエディターで開き、以下の例のようにハイブリッドネットワークで OVN-Kubernetes を設定します。

ハイブリッドネットワーク設定の指定

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
        - cidr: 10.132.0.0/14
          hostPrefix: 23
        hybridOverlayVXLANPort: 9898 2
```

1 追加のオーバーレイネットワーク上のノードに使用される CIDR 設定を指定します。**hybridClusterNetwork** CIDR は **clusterNetwork** CIDR と重複できません。

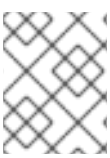
2 追加のオーバーレイネットワークのカスタム VXLAN ポートを指定します。これは、vSphere にインストールされたクラスターで Windows ノードを実行するために必要であり、その他のクラウドプロバイダー用に設定することはできません。カスタムポートには、デフォルトの **4789** ポートを除くいずれかのオープンポートを使用できます。この要件についての詳細は、Microsoft ドキュメントの Pod-to-pod connectivity between hosts is broken を参照してください。

注記

Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019 は、カスタムの VXLAN ポートの選択をサポートしないため、カスタムの **hybridOverlayVXLANPort** 値を持つクラスターではサポートされません。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。



注記

同じクラスターで Linux および Windows ノードを使用する方法についての詳細は、Understanding Windows container workloads を参照してください。

5.6.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

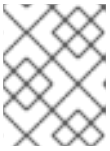
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

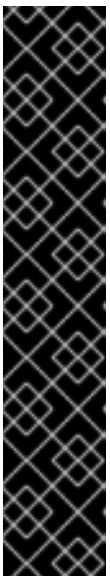
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



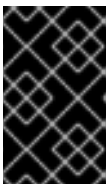
注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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

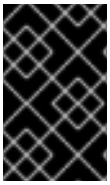


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

5.6.11. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.6.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

5.6.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを

追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

5.6.14. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

5.7. AZURE の既存 VNET へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にインストールできます。インストールプログラムは、さらなるカスタマイズが可能な必要なインフラストラクチャーの残りをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

5.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

5.7.2. OpenShift Container Platform クラスターでの VNet の再利用について

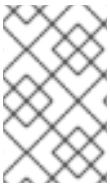
OpenShift Container Platform 4.8 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

5.7.2.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定されたサブネットがすべて存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

5.7.2.1.1. ネットワークセキュリティーグループの要件

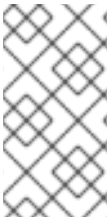
コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。

重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表5.15 必須ポート

ポート	説明	コントロールプレーン	コンピュート
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	



注記

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

5.7.2.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

5.7.2.3. クラスター間の分離

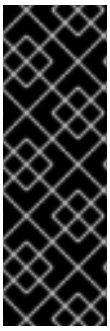
クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

5.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

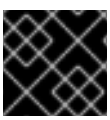
クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.7.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

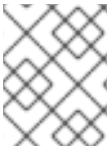
キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

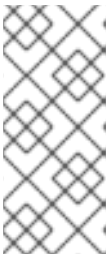
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

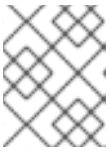
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.7.5. インストールプログラムの取得

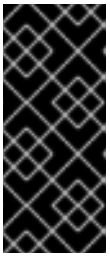
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.7.6. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

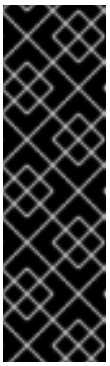
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

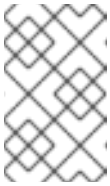
1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

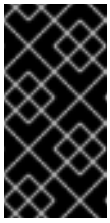
iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。



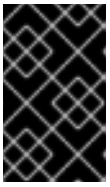
重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



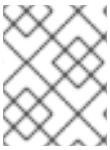
重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

5.7.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズす

るためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.7.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.16 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト

パラメーター	説明	値
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.7.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.17 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.7.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

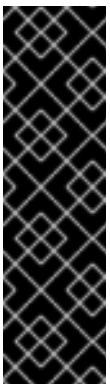
表5.18 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。

パラメーター	説明	値
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

5.7.6.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表5.19 追加の Azure パラメーター

パラメーター	説明	値
compute.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**standard_LRS**、**premium_LRS**、または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。

パラメーター	説明	値
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.resourceGroupName	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。
platform.azure.outboundType	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。	**LoadBalancer** または **UserDefinedRouting**。デフォルトは **LoadBalancer** です。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。

パラメーター	説明	値
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.cloudName	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。空の場合、デフォルト値の **AzurePublicCloud** が使用されます。	**AzurePublicCloud** または **AzureUSGovernmentCloud** などの有効なクラウド環境。

注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

5.7.6.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
```

```
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group 11
    region: centralus 12
    resourceGroupName: existing_resource_group 13
    networkResourceGroupName: vnet_resource_group 14
    virtualNetwork: vnet 15
    controlPlaneSubnet: control_plane_subnet 16
    computeSubnet: compute_subnet 17
    outboundType: Loadbalancer
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 18
fips: false 19
sshKey: ssh-ed25519 AAAA... 20
```

1 10 12 18 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノード (別名マスターノード) の最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

11 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

13 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

14 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

15 既存の VNet を使用する場合は、その名前を指定します。

16 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

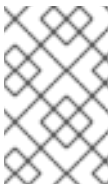
17 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

19 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

20 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

5.7.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限

り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

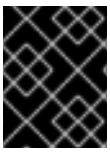
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.7.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

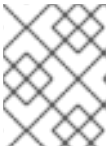
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

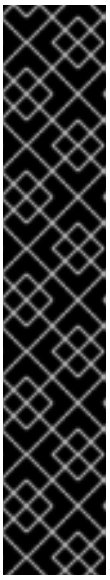
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

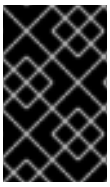
- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラ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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

5.7.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.7.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

5.7.10. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

5.7.11. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

5.8. プライベートクラスターの AZURE へのインストール

OpenShift Container Platform バージョン 4.8 では、プライベートクラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にインストールできます。インストールプログラムは、さらなるカスタマイズが可能な必要なインフラストラクチャーの残りをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

5.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みのリージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

5.8.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

<<<<<<< HEAD デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

5.8.2.1. Azure のプライベートクラスター

Microsoft Azure でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VNet とサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

ネットワークがプライベート VNET に接続される方法によって、クラスターのプライベート DNS レコードを解決するために DNS フォワーダーを使用する必要がある場合があります。クラスターのマシンは、DNS 解決に **168.63.129.16** を内部で使用します。詳細は、Azure ドキュメントの What is Azure Private DNS? および What is IP address 168.63.129.16? を参照してください。

クラスターには、Azure API にアクセスするためにインターネットへのアクセスが依然として必要です。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- **BaseDomainResourceGroup** (クラスターがパブリックレコードを作成しないため)
- パブリック IP アドレス
- パブリック DNS レコード

- パブリックエンドポイント

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

5.8.2.1.1. 制限事項

Azure 上のプライベートクラスターは、既存の VNet の使用に関連する制限のみの制限を受けます。

5.8.2.2. ユーザー定義のアウトバウンドルーティング

OpenShift Container Platform では、クラスターがインターネットに接続するために独自のアウトバウンドルーティングを選択できます。これにより、パブリック IP アドレスおよびパブリックロードバランサーの作成を省略できます。

クラスターをインストールする前に、**install-config.yaml** ファイルのパラメーターを変更してユーザー定義のルーティングを設定できます。クラスターのインストール時にアウトバウンドルーティングを使用するには、既存の VNet が必要です。インストールプログラムはこれを設定しません。

クラスターをユーザー定義のルーティングを使用するように設定する際に、インストールプログラムは以下のリソースを作成しません。

- インターネットにアクセスするためのアウトバウンドルール。
- パブリックロードバランサーのパブリック IP。
- アウトバウンド要求のパブリックロードバランサーにクラスターマシンを追加する Kubernetes Service オブジェクト。

ユーザー定義のルーティングを設定する前に、以下の項目が利用可能であることを確認する必要があります。

- 内部レジストリーミラーを使用しない場合は、コンテナーイメージのプルにインターネットへの Egress を使用できます。
- クラスターは Azure API にアクセスできます。
- 各種の allowlist エンドポイントが設定されます。これらのエンドポイントについては、**ファイアウォールの設定**セクションで参照できます。

ユーザー定義のルーティングを使用したインターネットアクセスでサポートされる既存のネットワーク設定がいくつかあります。

ネットワークアドレス変換のあるプライベートクラスター
Azure VNET NAT (ネットワークアドレス変換) を使用して、クラスター内のサブネットのアウトバウンドインターネットアクセスを提供できます。設定手順については、Azure ドキュメントの Create a NAT gateway using Azure CLI を参照してください。

Azure NAT およびユーザー定義のルーティングが設定された VNet 設定を使用する場合、パブリックエンドポイントのないプライベートクラスターを作成できます。

Azure ファイアウォールのあるプライベートクラスター
Azure ファイアウォールを使用して、クラスターのインストールに使用される VNet のアウトバウンドルーティングを提供できます。Azure ドキュメントで Azure ファイアウォールのあるユーザー定義のルーティングを提供する方法 について確認することができます。

Azure ファイアウォールおよびユーザー定義のルーティングが設定された VNet 設定を使用する場合、パブリックエンドポイントのないプライベートクラスターを作成できます。

プロキシー設定のあるプライベートクラスター
ユーザー定義のルーティングと共にプロキシーを使用し、インターネットへの egress を許可することができます。クラスター Operator がプロキシーを使用して Azure API にアクセスしないようにする必要があります。Operator はプロキシー外から Azure API にアクセスできる必要があります。

0.0.0.0/0 が Azure によって自動的に設定された状態で、サブネットのデフォルトのルートテーブルを使用する場合、すべての Azure API 要求は、IP アドレスがパブリックな場合でも Azure の内部ネットワークでルーティングされます。ネットワークセキュリティーグループのルールが Azure API エンドポイントへの egress を許可している限り、ユーザー定義のルーティングが設定されたプロキシーにより、パブリックエンドポイントのないプライベートクラスターを作成できます。

インターネットアクセスのないプライベートクラスター
Azure API を除く、インターネットへのすべてのアクセスを制限するプライベートネットワークをインストールできます。これは、リリースイメージレジストリーをローカルにミラーリングすることによって実現されます。クラスターは以下にアクセスできる必要があります。

- コンテナーイメージのプルを可能にする内部レジストリーミラー
- Azure API へのアクセス

各種の要件を利用可能な場合、ユーザー定義のルーティングを使用して、パブリックエンドポイントのないプライベートクラスターを作成できます。

5.8.3. OpenShift Container Platform クラスターでの VNet の再利用について

OpenShift Container Platform 4.8 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

5.8.3.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

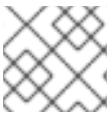
VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定されたサブネットがすべて存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

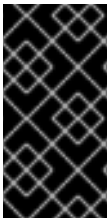


注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

5.8.3.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。



重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表5.20 必須ポート

ポート	説明	コントロールプレーン	コンピュート
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	

注記

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

5.8.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

5.8.3.3. クラスター間の分離

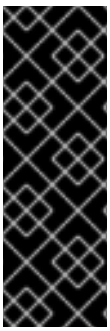
クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

5.8.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.8.5. クラスターノードの SSH アクセス用のキーペアの生成

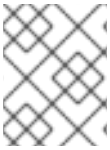
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

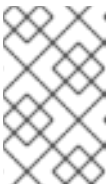
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定しま

9 。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.8.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

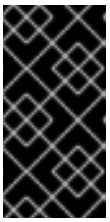
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.8.7. インストール設定ファイルの手動作成

内部ネットワークからのみアクセスでき、インターネット上に表示されないプライベート OpenShift Container Platform クラスターのインストールの場合、インストール設定ファイルを手動で生成する必要があります。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

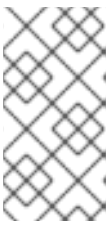
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

5.8.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.8.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.21 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

5.8.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.22 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.8.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.23 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

5.8.7.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表5.24 追加の Azure パラメーター

パラメーター	説明	値
compute.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**standard_LRS**、**premium_LRS**、または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。

パラメーター	説明	値
platform.azure.resourceGroupName	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。
platform.azure.outboundType	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。	**LoadBalancer** または **UserDefinedRouting**。デフォルトは **LoadBalancer** です。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。

パラメーター	説明	値
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.cloudName	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。空の場合、デフォルト値の **AzurePublicCloud** が使用されます。	**AzurePublicCloud** または **AzureUSGovernmentCloud** などの有効なクラウド環境。

注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

5.8.7.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
```

```
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
        diskType: Standard_LRS
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group 11
    region: centralus 12
    resourceGroupName: existing_resource_group 13
    networkResourceGroupName: vnet_resource_group 14
    virtualNetwork: vnet 15
    controlPlaneSubnet: control_plane_subnet 16
    computeSubnet: compute_subnet 17
    outboundType: UserDefinedRouting 18
    cloudName: AzurePublicCloud
pullSecret: '{"auths": ...}' 19
fips: false 20
sshKey: ssh-ed25519 AAAA... 21
publish: Internal 22
```

1 10 12 19 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノード (別名マスターノード) の最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

11 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

13 クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

14 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

15 既存の VNet を使用する場合は、その名前を指定します。

16 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

17 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

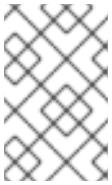
18 独自のアウトバウンドルーティングをカスタマイズすることができます。ユーザー定義のルーティングを設定すると、クラスターに外部エンドポイントが公開されなくなります。エグレスのユーザー定義のルーティングでは、クラスターを既存の VNet にデプロイする必要があります。

20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

22 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

5.8.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存しま

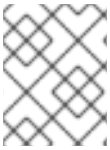
9。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

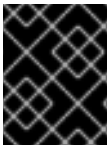


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.8.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

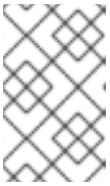
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、以下を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

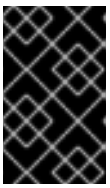
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

5.8.9. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.8.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

5.8.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

5.8.12. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

5.9. AZURE の GOVERNMENT リージョンへのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを Microsoft Azure の government リージョンにインストールできます。government リージョンを設定するには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

5.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 し、クラスターをデプロイするテスト済みおよび検証済みの government リージョンを決定している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

5.9.2. Azure government リージョン

OpenShift Container Platform は、クラスターの Microsoft Azure Government (MAG) リージョンへのデプロイをサポートします。MAG は、機密ワークロードを Azure で実行する必要のある連邦、州、地方の米国の各種の政府機関、請負業者、教育機関、およびその他の米国の顧客向けに設計されています。MAG は、government のみのデータセンターリージョンで設定されており、すべてに 影響レベル 5 の暫定認証 が付与されます。

MAG リージョンにインストールするには、**install-config.yaml** ファイルで Azure Government 専用のクラウドインスタンスおよびリージョンを手動で設定する必要があります。また、適切な government 環境を参照するようにサービスプリンシパルを更新する必要もあります。

注記

Azure government リージョンは、インストールプログラムからガイド付きターミナルプロンプトを使用して選択することはできません。リージョンは **install-config.yaml** ファイルで手動で定義する必要があります。指定されたリージョンに基づいて、**AzureUSGovernmentCloud** などの専用のクラウドインスタンスも設定するようにしてください。

5.9.3. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

<<<<<<< HEAD デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

5.9.3.1. Azure のプライベートクラスター

Microsoft Azure でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VNet とサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

ネットワークがプライベート VNET に接続される方法によって、クラスターのプライベート DNS レコードを解決するために DNS フォワーダーを使用する必要がある場合があります。クラスターのマシンは、DNS 解決に **168.63.129.16** を内部で使用します。詳細は、Azure ドキュメントの What is Azure Private DNS? および What is IP address 168.63.129.16? を参照してください。

クラスターには、Azure API にアクセスするためにインターネットへのアクセスが依然として必要です。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- **BaseDomainResourceGroup** (クラスターがパブリックレコードを作成しないため)
- パブリック IP アドレス
- パブリック DNS レコード
- パブリックエンドポイント

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

5.9.3.1.1. 制限事項

Azure 上のプライベートクラスターは、既存の VNet の使用に関連する制限のみの制限を受けます。

5.9.3.2. ユーザー定義のアウトバウンドルーティング

OpenShift Container Platform では、クラスターがインターネットに接続するために独自のアウトバウンドルーティングを選択できます。これにより、パブリック IP アドレスおよびパブリックロードバランサーの作成を省略できます。

クラスターをインストールする前に、**install-config.yaml** ファイルのパラメーターを変更してユーザー定義のルーティングを設定できます。クラスターのインストール時にアウトバウンドルーティングを使用するには、既存の VNet が必要です。インストールプログラムはこれを設定しません。

クラスターをユーザー定義のルーティングを使用するように設定する際に、インストールプログラムは以下のリソースを作成しません。

- インターネットにアクセスするためのアウトバウンドルール。
- パブリックロードバランサーのパブリック IP。
- アウトバウンド要求のパブリックロードバランサーにクラスターマシンを追加する Kubernetes Service オブジェクト。

ユーザー定義のルーティングを設定する前に、以下の項目が利用可能であることを確認する必要があります。

- 内部レジストリーミラーを使用しない場合は、コンテナーイメージのプルにインターネットへの Egress を使用できます。
- クラスターは Azure API にアクセスできます。
- 各種の allowlist エンドポイントが設定されます。これらのエンドポイントについては、**ファイアウォールの設定**セクションで参照できます。

ユーザー定義のルーティングを使用したインターネットアクセスでサポートされる既存のネットワーク設定がいくつかあります。

ネットワークアドレス変換のあるプライベートクラスター
Azure VNET NAT (ネットワークアドレス変換) を使用して、クラスター内のサブネットのアウトバウンドインターネットアクセスを提供できます。設定手順については、Azure ドキュメントの Create a NAT gateway using Azure CLI を参照してください。

Azure NAT およびユーザー定義のルーティングが設定された VNet 設定を使用する場合、パブリックエンドポイントのないプライベートクラスターを作成できます。

Azure ファイアウォールのあるプライベートクラスター
Azure ファイアウォールを使用して、クラスターのインストールに使用される VNet のアウトバウンドルーティングを提供できます。Azure ドキュメントで Azure ファイアウォールのあるユーザー定義のルーティングを提供する方法 について確認することができます。

Azure ファイアウォールおよびユーザー定義のルーティングが設定された VNet 設定を使用する場合、パブリックエンドポイントのないプライベートクラスターを作成できます。

プロキシー設定のあるプライベートクラスター
ユーザー定義のルーティングと共にプロキシーを使用し、インターネットへの egress を許可することができます。クラスター Operator がプロキシーを使用して Azure API にアクセスしないようにする必要があります。Operator はプロキシー外から Azure API にアクセスできる必要があります。

0.0.0.0/0 が Azure によって自動的に設定された状態で、サブネットのデフォルトのルートテーブルを使用する場合、すべての Azure API 要求は、IP アドレスがパブリックな場合でも Azure の内部ネットワークでルーティングされます。ネットワークセキュリティーグループのルールが Azure API エンドポイントへの egress を許可している限り、ユーザー定義のルーティングが設定されたプロキシーにより、パブリックエンドポイントのないプライベートクラスターを作成できます。

インターネットアクセスのないプライベートクラスター
Azure API を除く、インターネットへのすべてのアクセスを制限するプライベートネットワークをインストールできます。これは、リリースイメージレジストリーをローカルにミラーリングすることによって実現されます。クラスターは以下にアクセスできる必要があります。

- コンテナーイメージのプルを可能にする内部レジストリーミラー
- Azure API へのアクセス

各種の要件を利用可能な場合、ユーザー定義のルーティングを使用して、パブリックエンドポイントのないプライベートクラスターを作成できます。

5.9.4. OpenShift Container Platform クラスターでの VNet の再利用について

OpenShift Container Platform 4.8 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

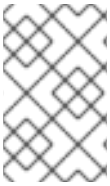
OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

5.9.4.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加の

ネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ



注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定されたサブネットがすべて存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットがあります
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

注記

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

5.9.4.1.1. ネットワークセキュリティーグループの要件

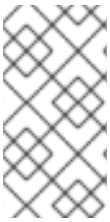
コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。

重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表5.25 必須ポート

ポート	説明	コントロールプレーン	コンピュート
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシンをプロビジョニングするためのマシン設定サーバーへの内部通信を許可します。	x	



注記

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

5.9.4.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリ

ティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

5.9.4.3. クラスター間の分離

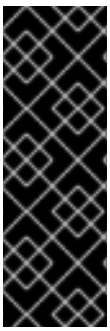
クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

5.9.5. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.9.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

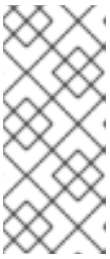
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

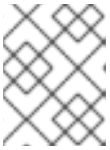
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.9.7. インストールプログラムの取得

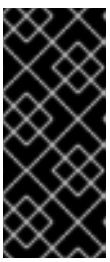
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

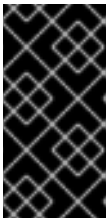
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.9.8. インストール設定ファイルの手動作成

OpenShift Container Platform を Microsoft Azure の government リージョンにインストールする場合、インストール設定ファイルを手動で生成する必要があります。

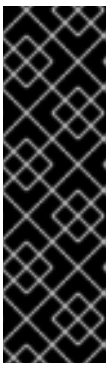
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



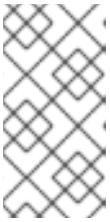
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

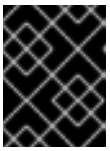
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

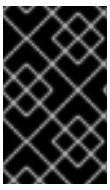
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

5.9.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.9.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.26 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.9.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.27 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.9.8.1.3. オプションの設定パラメーター

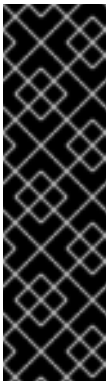
オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.28 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

5.9.8.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表5.29 追加の Azure パラメーター

パラメーター	説明	値
compute.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **128** です。
compute.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**standard_LRS**、**premium_LRS**、または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。デフォルトは **1024** です。
controlPlane.platform.azure.osDisk.diskType	ディスクのタイプを定義します。	**premium_LRS** または **standardSSD_LRS**。デフォルトは **premium_LRS** です。
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.resourceGroupName	クラスターをインストールする既存のリソースグループの名前。このリソースグループは空で、この特定のクラスターにのみ使用する必要があります。クラスターコンポーネントは、リソースグループ内のすべてのリソースの所有権を想定します。インストールプログラムのサービスプリンシパルの範囲をこのリソースグループに制限する場合は、環境内でインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。	文字列 (例: **existing_resource_group**)。

パラメーター	説明	値
platform.azure.outboundType	クラスターをインターネットに接続するために使用されるアウトバウンドルーティングストラテジー。ユーザー定義のルーティングを使用している場合、クラスターをインストールする前にアウトバウンドルーティングがすでに設定されている既存のネットワークが利用可能な状態にする必要があります。インストールプログラムはユーザー定義のルーティングの設定を行いません。	**LoadBalancer** または **UserDefinedRouting**。デフォルトは **LoadBalancer** です。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.cloudName	適切な Azure API エンドポイントで Azure SDK を設定するために使用される Azure クラウド環境の名前。空の場合、デフォルト値の **AzurePublicCloud** が使用されます。	**AzurePublicCloud** または **AzureUSGovernmentCloud** などの有効なクラウド環境。

注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

5.9.8.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
        diskType: Premium_LRS
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
        diskType: Standard_LRS
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: resource_group (11)
    region: usgovvirginia
    resourceGroupName: existing_resource_group (12)
    networkResourceGroupName: vnet_resource_group (13)
```

```
    virtualNetwork: vnet (14)
    controlPlaneSubnet: control_plane_subnet (15)
    computeSubnet: compute_subnet (16)
    outboundType: UserDefinedRouting (17)
    cloudName: AzureUSGovernmentCloud (18)
pullSecret: '{"auths": ...}' (19)
fips: false (20)
sshKey: ssh-ed25519 AAAA... (21)
publish: Internal (22)
```

(1) (10) (19) 必須。

(2) (6) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (7) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

(4) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

(5) (8) 使用するディスクのサイズは、GB 単位で指定できます。コントロールプレーンノード (別名マスターノード) の最小推奨値は 1024 GB です。

(9) マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

(11) ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

(12) クラスターをインストールする既存のリソースグループの名前を指定します。定義されていない場合は、クラスターに新しいリソースグループが作成されます。

(13) 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

(14) 既存の VNet を使用する場合は、その名前を指定します。

(15) 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

(16) 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

(17) 独自のアウトバウンドルーティングをカスタマイズすることができます。ユーザー定義のルーティングを設定すると、クラスターに外部エンドポイントが公開されなくなります。エグレスのユー

ザー定義のルーティングでは、クラスターを既存の VNet にデプロイする必要があります。

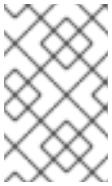
18 クラスターをデプロイする Azure クラウド環境の名前を指定します。**AzureUSGovernmentCloud** を Microsoft Azure Government (MAG) リージョンにデプロイするように設定します。デフォルト値は **AzurePublicCloud** です。

20 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

21 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

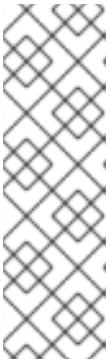
22 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

5.9.8.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

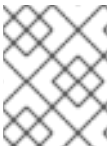
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.9.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

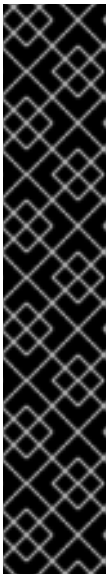
出力例

```
...
INFO Install complete!
```

```
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

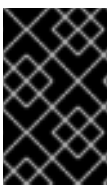
- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラ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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

5.9.10. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.9.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

5.9.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

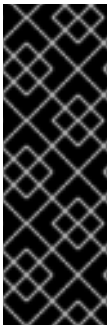
5.9.13. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

5.10. ARM テンプレートを使用したクラスターの AZURE へのインストール

OpenShift Container Platform バージョン 4.8 では、独自に提供するインフラストラクチャーを使用して、クラスターを Microsoft Azure にインストールできます。

これらの手順の実行、または独自の手順の作成を支援する複数の Azure Resource Manager (ARM) テンプレートが提供されます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の ARM テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

5.10.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように Azure アカウントを設定 している。
- Azure CLI をダウンロードし、これをコンピューターにインストールしている。Azure ドキュメントの Install the Azure CLI を参照してください。以下のドキュメントについては、直近で Azure CLI のバージョン **2.2.0** を使用してテストされていますAzure CLI コマンドは、使用するバージョンによって動作が異なる場合があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

5.10.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

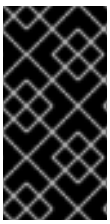
- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.10.3. Azure プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするために Azure プロジェクトを設定する必要があります。

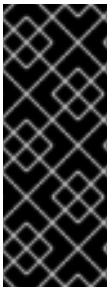


重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

5.10.3.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。



重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	40	リージョンごとに 20	デフォルトのクラスターには 40 の vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップマシンは 4 vCPUS を使用する **Standard_D4s_v3** マシンを使用し、コントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、さらにワーカーマシンは、4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、デフォルトクラスターには 40 の vCPU が必要になります。4 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、または異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。 デフォルトで、インストールプログラムはコントロールプレーンおよびコンピュートマシンを、リージョン 内の すべてのアベイラビリティーゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
OS ディスク	7		仮想マシン OS ディスクは、最小スループットが 5000 IOPS/200MBps を維持できる必要があります。このスループットは、P30 (最低 1 TiB Premium SSD) を使用することで実現できます。Azure では、ディスクのパフォーマンスは SSD のディスクサイズに直接依存するため、利用可能な **Standard_D8s_v3** またはその他の同様のマシンタイプでサポートされるスループット (ターゲット: 5000 IOPS) を実現するには、少なくとも P30 ディスクが必要です。 読み取りのレイテンシーが低く抑え、読み取り IOPS およびスループットが高く保つには、ホストのキャッシュを **ReadOnly** に設定する必要があります。仮想マシンメモリーまたはローカル SSD ディスクにあるキャッシュから実行された読み取りは、Blob ストレージにあるデータディスクからの読み取りよりもはるかに高速です。
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	6	リージョンごとに 65,536	各デフォルトクラスターには、6 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。
ネットワークセキュリティーグループ	2	5000	各デフォルトクラスター。各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。

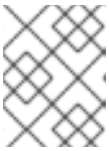
コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。
スポット VM vCPU (オプション)	0 スポット VM を設定する場合には、クラスターのコンピュートノードごとにスポット VM vCPU が 2 つ必要です。	リージョンごとに 20	これはオプションのコンポーネントです。スポット VM を使用するには、Azure の既定の制限を最低でも、クラスター内のコンピュートノード数の 2 倍に増やす必要があります。 **注記** コントロールプレーンノードにスポット VM を使用することはお勧めしません。

5.10.3.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。



注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、 新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

この DNS ゾーンの作成例 を参照し、Azure の DNS ソリューションを確認することができます。

5.10.3.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。

注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。

2. **New support request** をクリックしてから必要な値を選択します。

   a. **Issue type** 一覧から、**Service and subscription limits (quotas)** を選択します。

   b. **Subscription** 一覧から、変更するサブスクリプションを選択します。

   c. **Quota type** 一覧から、引き上げるクォータを選択します。たとえば、**Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。

   d. **Next: Solutions** をクリックします。

3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。
   a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。
   b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。
4. **Next: Review + create** をクリックしてから **Create** をクリックします。

5.10.3.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

5.10.3.5. 必要な Azure ロール

OpenShift Container Platform には、Microsoft Azure リソースを管理できるようにサービスプリンシパルが必要です。サービスプリンシパルの作成前に、Azure アカウントサブスクリプションに次のロールが必要です。

- **User Access Administrator**
- **Owner**

Azure ポータルでロールを設定するには、Azure ドキュメントの Manage access to Azure resources using RBAC and the Azure portal を参照します。

5.10.3.6. サービスプリンシパルの作成

OpenShift Container Platform およびそのインストールプログラムは Azure Resource Manager 経由で Microsoft Azure リソースを作成する必要があるため、これを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- **jq** パッケージをインストールします。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。

手順

1. Azure CLI にログインします。

```
$ az login
```

   認証情報を使用して Web コンソールで Azure にログインします。

2. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。

a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** パラメーターの値が適切なサブスクリプションの UUID であることを確認します。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <id> (1)
```

(1) 使用する必要のあるサブスクリプションの **id** の値を **<id>** の代わりに使用します。

d. アクティブなサブスクリプションを変更したら、アカウント情報を再度表示します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 直前の出力の **tenantId** および **id** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

4. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
```

1 **<service_principal>** を、サービスプリンシパルに割り当てる名前に置き換えます。

出力例

```
Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the
required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
Retrying role assignment creation: 4/36
{
  "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
  "displayName": "<service_principal>",
  "name": "http://<service_principal>",
  "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
}
```

5. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

6. サービスプリンシパルに追加パーミッションを付与します。

- クラスターはそのコンポーネントの認証情報を割り当てできるように、**Contributor** および **User Access Administrator** ロールを常にアプリケーション登録サービスプリンシパルに追加する必要があります。

- Cloud Credential Operator (CCO) を **mint モード** で操作するには、アプリケーション登録サービスプリンシパルで **Azure Active Directory Graph/Application.ReadWrite.OwnedBy** API パーミッションも必要です。
- CCO を **passthrough モード** で操作するには、アプリケーション登録サービスプリンシパルで追加の API パーミッションは必要ありません。

CCO モードの詳細は、**認証および承認**ガイドのクラウドプロバイダークレデンシャルの管理セクションにある Cloud Credential Operator についてを参照してください。

注記

OpenShift Container Platform インストールプログラムのサービスプリンシパルの範囲を既存の Azure リソースグループに制限する場合は、環境内のインストールプログラムが使用する他のすべてのリソースに、パブリック DNS ゾーンや仮想ネットワークなどの必要なパーミッションがあることを確認する必要があります。インストールプログラムを使用してクラスターを破棄すると、このリソースグループが削除されます。

a. **User Access Administrator** ロールを割り当てるには、以下のコマンドを実行します。

```
$ az role assignment create --role "User Access Administrator" \
    --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \
      | jq '.[0].id' -r) 1
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

b. **Azure Active Directory Graph** パーミッションを割り当てるには、以下のコマンドを実行します。

```
$ az ad app permission add --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000 \
    --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

出力例

```
Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api
00000002-0000-0000-c000-000000000000" is needed to make the change effective
```

このコマンドで付与する特定のパーミッションについての詳細は、GUID Table for Windows Azure Active Directory Permissions を参照してください。

c. パーミッション要求を承認します。アカウントに Azure Active Directory テナント管理者ロールがない場合は、所属する組織のガイドラインに従い、テナント管理者にパーミッション要求を承認するようにリクエストしてください。

```
$ az ad app permission grant --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

関連情報

- CCO モードの詳細は、Cloud Crednetial Operator について を参照してください。

5.10.3.7. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンの一覧を動的に生成します。

サポート対象の Azure パブリックリージョン

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)

- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

サポート対象の Azure Government リージョン
以下の Microsoft Azure Government (MAG) リージョンのサポートが OpenShift Container Platform バージョン 4.6 に追加されています。

- **usgovtexas** (US Gov Texas)
- **usgovvirginia** (US Gov Virginia)

Azure ドキュメント の利用可能なすべての MAG リージョンを参照できます。他の提供される MAG リージョンは OpenShift Container Platform で機能することが予想されますが、まだテストされていません。

5.10.4. Azure Marketplace イメージの選択

Azure Marketplace オファリングを使用して OpenShift Container Platform クラスターをデプロイする場合は、最初に Azure Marketplace イメージを取得する必要があります。インストールプログラムは、このイメージを使用してワーカーノードをデプロイします。イメージを取得するときは、次の点を考慮してください。

- イメージは同じですが、Azure Marketplace のパブリシャーは地域によって異なります。北米にお住まいの場合は、**redhat** をパブリッシャーとして指定してください。EMEA にお住まいの場合は、**redhat-limited** をパブリッシャーとして指定してください。
- このオファーには、**rh-ocp-worker** SKU と **rh-ocp-worker-gen1** SKU が含まれています。**rh-ocp-worker** SKU は、Hyper-V 世代のバージョン 2 VM イメージを表します。OpenShift Container Platform で使用されるデフォルトのインスタンスタイプは、バージョン 2 と互換性があります。バージョン 1 のみと互換性のあるインスタンスタイプを使用する場合は、**rh-ocp-worker-gen1** SKU に関連付けられたイメージを使用します。**rh-ocp-worker-gen1** SKU は、Hyper-V バージョン 1 VM イメージを表します。

前提条件

- Azure CLI クライアント **(az)** をインストールしている。
- お客様の Azure アカウントにはオファーのエンタイトルメントがあり、Azure CLI クライアントを使用してこのアカウントにログインしている。

手順

1. 以下のいずれかのコマンドを実行して、利用可能なすべての OpenShift Container Platform イメージを表示します。

   - 北米:

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
     ```

     出力例

     ```
     Offer          Publisher      Sku                Urn                                                                  Version
     -------------  -------------  -----------------  -------------------------------------------------------------------
     ----  --------------
     rh-ocp-worker  RedHat         rh-ocp-worker      RedHat:rh-ocp-worker:rh-
     ocpworker:4.8.2021122100              4.8.2021122100
     rh-ocp-worker  RedHat         rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
     gen1:4.8.2021122100         4.8.2021122100
     ```

   - EMEA:

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
     ```

     出力例

     ```
     Offer          Publisher      Sku                Urn                                                                  Version
     -------------  -------------  -----------------  -------------------------------------------------------------------
     ----  --------------
     rh-ocp-worker  redhat-limited  rh-ocp-worker      redhat-limited:rh-ocp-worker:rh-ocp-
     worker:4.8.2021122100       4.8.2021122100
     rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
     worker-gen1:4.8.2021122100  4.8.2021122100
     ```

     注記

     インストールする OpenShift Container Platform のバージョンに関係なく、使用する Azure Marketplace イメージの正しいバージョンは 4.8.x です。必要に応じて、インストールプロセスの一環として、VM が自動的にアップグレードされます。

2. 次のいずれかのコマンドを実行して、オファーのイメージを調べます。

   - 北米:

     ```
     $ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - EMEA:

     ```
     $ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

3. 次のコマンドのいずれかを実行して、オファーの条件を確認します。

   - 北米:

     ```
     $ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

-

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 次のコマンドのいずれかを実行して、オファリングの条件に同意します。
   - 北米:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

   - EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. オファーのイメージの詳細を記録して使用し、**06_workers.json** Azure Resource Manager (ARM) テンプレートを更新します。**id** パラメーターを削除し、オファーの値を使用して **offer**、**publisher**、**sku**、および **version** パラメーターを追加して、**storageProfile.imageReference** フィールドを更新します。追加のワーカーマシンの作成に関するセクションに、サンプルテンプレートがあります。

5.10.5. インストールプログラムの取得

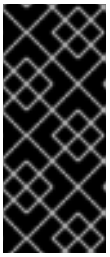
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

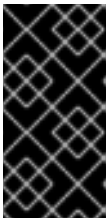
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.10.6. クラスターノードの SSH アクセス用のキーペアの生成

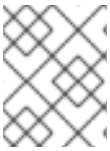
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

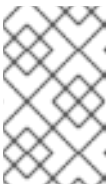
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

5.10.7. Azure のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Microsoft Azure にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

5.10.7.1. オプション: 別個の /var パーティションの作成

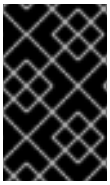
OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。



重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

5.10.7.2. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

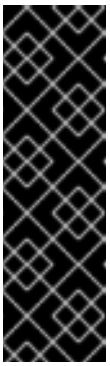
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

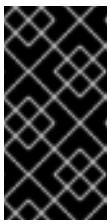
iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

c. オプション: クラスターでコンピュートマシンをプロビジョニングするよう設定する必要がない場合は、 **install-config.yaml** ファイルで **compute** プールの **replicas** を **0** に設定してコンピュートプールを空にします。

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 **0** に設定します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

5.10.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

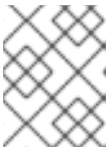
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.10.7.4. ARM テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Microsoft Azure で実行するのに役立つ指定の Azure Resource Manager (ARM) テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の ARM テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. 提供される ARM テンプレートで使用される **install-config.yaml** にある一般的な変数をエクスポートします。

```
$ export CLUSTER_NAME=<cluster_name> (1)
$ export AZURE_REGION=<azure_region> (2)
$ export SSH_KEY=<ssh_key> (3)
$ export BASE_DOMAIN=<base_domain> (4)
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> (5)
```

(1) **install-config.yaml** ファイルからの **.metadata.name** 属性の値。

(2) クラスターをデプロイするリージョン (例: **centralus**)。これは、**install-config.yaml** ファイルからの **.platform.azure.region** 属性の値です。

(3) 文字列としての SSH RSA 公開鍵ファイル。SSH キーは、スペースが含まれているために引用符で囲む必要があります。これは、**install-config.yaml** ファイルからの **.sshKey** 属性の値です。

(4) クラスターをデプロイするベースドメイン。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。これは、**install-config.yaml** からの **.baseDomain** 属性の値です。

(5) パブリック DNS ゾーンが存在するリソースグループ。これは、**install-config.yaml** ファイルからの **.platform.azure.baseDomainResourceGroupName** 属性の値です。

以下に例を示します。

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 認証情報をエクスポートします。

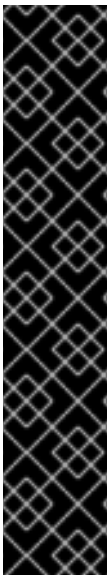
```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

5.10.7.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. ユーザーによってプロビジョニングされるインフラストラクチャーで Azure を設定する場合、Azure Resource Manager (ARM) テンプレートで後に使用するためにマニフェストファイルに定義された一般的な変数の一部をエクスポートする必要があります。

   a. 以下のコマンドを使用してインフラストラクチャー ID をエクスポートします。

```
$ export INFRA_ID=<infra_id> 1
```

   1 OpenShift Container Platform クラスターには、**<cluster_name>-<random_string>** の形式の識別子 (**INFRA_ID**) が割り当てられます。これは、提供される ARM テンプレートを使用して作成されるほとんどのリソースのベース名として使用されます。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.infrastructureName** 属性の値です。

   b. 以下のコマンドを使用してリソースグループをエクスポートします。

```
$ export RESOURCE_GROUP=<resource_group> 1
```

1 この Azure デプロイメントで作成されたすべてのリソースは、リソースグループ の一部として存在します。リソースグループ名は、**<cluster_name>-<random_string>-rg** 形式の **INFRA_ID** をベースとしています。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.platformStatus.azure.resourceGroupName** 属性の値です。

7. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5.10.8. Azure リソースグループおよびアイデンティティーの作成

Microsoft Azure リソースグループ およびリソースグループのアイデンティティーを作成する必要があります。これらはいずれも Azure での OpenShift Container Platform クラスターのインストール時に使用されます。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. サポートされる Azure リージョンにリソースグループを作成します。

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

2. リソースグループの Azure アイデンティティーを作成します。

```
$ az identity create -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity
```

これは、クラスター内の Operator に必要なアクセスを付与するために使用されます。たとえば、これにより Ingress Operator はパブリック IP およびそのロードバランサーを作成できます。Azure アイデンティティーをロールに割り当てる必要があります。

3. Contributor ロールを Azure アイデンティティーに付与します。

   a. Azure ロールの割り当てで必要な以下の変数をエクスポートします。

   ```
   $ export PRINCIPAL_ID=`az identity show -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity --query principalId --out tsv`
   ```

   ```
   $ export RESOURCE_GROUP_ID=`az group show -g ${RESOURCE_GROUP} --query id --out tsv`
   ```

   b. Contributor ロールをアイデンティティーに割り当てます。

   ```
   $ az role assignment create --assignee "${PRINCIPAL_ID}" --role 'Contributor' --scope "${RESOURCE_GROUP_ID}"
   ```

5.10.9. RHCOS クラスターイメージおよびブートストラップ Ignition 設定ファイルのアップロード

Azure クライアントは、ローカルにあるファイルに基づくデプロイメントをサポートしないため、RHCOS 仮想ハードディスク (VHD) クラスターイメージおよびブートストラップ Ignition 設定ファイルをコピーし、それらをストレージコンテナーに保存し、それらをデプロイメント時にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. VHD クラスターイメージを保存するために Azure ストレージアカウントを作成します。

   ```
   $ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
   ```

警告

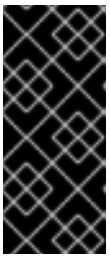
Azure ストレージアカウント名は 3 文字から 24 文字の長さで、数字および小文字のみを使用する必要があります。**CLUSTER_NAME** 変数がこれらの制限に準拠しない場合、Azure ストレージアカウント名を手動で定義する必要があります。Azure ストレージアカウント名の制限についての詳細は、Azure ドキュメントの Resolve errors for storage account names を参照してください。

2. ストレージアカウントキーを環境変数としてエクスポートします。

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. 使用する RHCOS バージョンを選択し、その VHD の URL を環境変数にエクスポートします。

```
$ export VHD_URL=`curl -s https://raw.githubusercontent.com/openshift/installer/release-
4.8/data/data/rhcos.json | jq -r .azure.url`
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージを指定する必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

4. 選択した VHD を blob にコピーします。

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-
key ${ACCOUNT_KEY}
```

```
$ az storage blob copy start --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} --destination-blob "rhcos.vhd" --destination-container vhd --source-uri
"${VHD_URL}"
```

VHD コピータスクの進捗を追跡するには、以下のスクリプトを実行します。

```
status="unknown"
while [ "$status" != "success" ]
do
  status=`az storage blob show --container-name vhd --name "rhcos.vhd" --account-name
${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -o tsv --query
properties.copy.status`
  echo $status
done
```

5. blob ストレージコンテナーを作成し、生成された **bootstrap.ign** ファイルをアップロードします。

```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} --public-access blob
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

5.10.10. DNS ゾーンの作成例

DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターに必要です。シナリオに適した DNS ストラテジーを選択する必要があります。

この例の場合、Azure の DNS ソリューション が使用されるため、外部 (インターネット) の可視性のために新規パブリック DNS ゾーンと、内部クラスターの解決用にプライベート DNS ゾーンが作成されます。

注記

パブリック DNS ゾーンは、クラスターデプロイメントと同じリソースグループに存在している必要はなく、必要なベースドメイン用にすでに組織内に存在している可能性があります。その場合、パブリック DNS ゾーンの作成を省略できます。先に生成したインストール設定がこのシナリオに基づいていることを確認してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. **BASE_DOMAIN_RESOURCE_GROUP** 環境変数でエクスポートされたリソースグループに、新規のパブリック DNS ゾーンを作成します。

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

   すでに存在するパブリック DNS ゾーンを使用している場合は、この手順を省略できます。

2. このデプロイメントの残りの部分と同じリソースグループにプライベート DNS ゾーンを作成します。

```
$ az network private-dns zone create -g ${RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

Azure でのパブリック DNS ゾーンの設定 についてのセクションを参照してください。

5.10.11. Azure での VNet の作成

OpenShift Container Platform クラスター用に Microsoft Azure で使用する仮想ネットワーク (VNet) を作成する必要があります。各種の要件を満たすように VPC をカスタマイズできます。VNet を作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VNet の ARM テンプレート**セクションからテンプレートをコピーし、これを **01_vnet.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要な VNet について記述しています。

2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. VNet テンプレートをプライベート DNS ゾーンにリンクします。

```
$ az network private-dns link vnet create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n ${INFRA_ID}-network-link -v "${INFRA_ID}-vnet"
-e false
```

5.10.11.1. VNet の ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な VNet をデプロイすることができます。

例5.101_vnet.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "addressPrefix" : "10.0.0.0/16",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetPrefix" : "10.0.0.0/24",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetPrefix" : "10.0.1.0/24",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
```

```
    "type" : "Microsoft.Network/virtualNetworks",
    "name" : "[variables('virtualNetworkName')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/networkSecurityGroups/', variables('clusterNsgName'))]"
    ],
    "properties" : {
      "addressSpace" : {
        "addressPrefixes" : [
          "[variables('addressPrefix')]"
        ]
      },
      "subnets" : [
        {
          "name" : "[variables('masterSubnetName')]",
          "properties" : {
            "addressPrefix" : "[variables('masterSubnetPrefix')]",
            "serviceEndpoints": [],
            "networkSecurityGroup" : {
              "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
            }
          }
        },
        {
          "name" : "[variables('nodeSubnetName')]",
          "properties" : {
            "addressPrefix" : "[variables('nodeSubnetPrefix')]",
            "serviceEndpoints": [],
            "networkSecurityGroup" : {
              "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
            }
          }
        }
      ]
    }
  },
  {
    "type" : "Microsoft.Network/networkSecurityGroups",
    "name" : "[variables('clusterNsgName')]",
    "apiVersion" : "2018-10-01",
    "location" : "[variables('location')]",
    "properties" : {
      "securityRules" : [
        {
          "name" : "apiserver_in",
          "properties" : {
            "protocol" : "Tcp",
            "sourcePortRange" : "*",
            "destinationPortRange" : "6443",
            "sourceAddressPrefix" : "*",
            "destinationAddressPrefix" : "*",
            "access" : "Allow",
            "priority" : 101,
            "direction" : "Inbound"
```

```
          }
        }
      ]
    }
  }
 ]
}
```

5.10.12. Azure インフラストラクチャー用の RHCOS クラスターイメージのデプロイ

OpenShift Container Platform ノードに Microsoft Azure 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- RHCOS 仮想ハードディスク (VHD) クラスターイメージを Azure ストレージコンテナーに保存します。
- ブートストラップ Ignition 設定ファイルを Azure ストレージコンテナーに保存します。

手順

1. 本トピックの **イメージストレージの ARM テンプレート**セクションからテンプレートをコピーし、これを **02_storage.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なイメージストレージ について記述しています。
2. RHCOS VHD blob URL を変数としてエクスポートします。

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. クラスターイメージのデプロイ

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1. マスターマシンおよびワーカーマシンを作成するために使用される RHCOS VHD の blob URL。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

5.10.12.1. イメージストレージの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な保存された Red Hat Enterprise Linux CoreOS (RHCOS) をデプロイすることができます。

例5.202_storage.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vhdBlobURL" : {
      "type" : "string",
      "metadata" : {
        "description" : "URL pointing to the blob where the VHD to be used to create master and worker machines is located"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "imageName" : "[concat(parameters('baseName'), '-image')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Compute/images",
      "name": "[variables('imageName')]",
      "location" : "[variables('location')]",
      "properties": {
        "storageProfile": {
          "osDisk": {
            "osType": "Linux",
            "osState": "Generalized",
            "blobUri": "[parameters('vhdBlobURL')]",
            "storageAccountType": "Standard_LRS"
          }
        }
      }
    }
  ]
}
```

5.10.13. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

5.10.13.1. DHCP を使用したクラスターノードのホスト名の設定

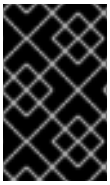
Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

5.10.13.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表5.30 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット

プロトコル	ポート	説明
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表5.31 すべてのマシンからコントロールプレーンへの通信に使用されるポート

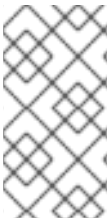
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表5.32 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

5.10.14. Azure でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散を Microsoft Azure で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。

手順

1. 本トピックの **ネットワークおよびロードバランサーの ARM テンプレート**セクションからテンプレートをコピーし、これを **03_infra.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。
2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
```

```
--parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 1
--parameters baseName="${INFRA_ID}" 2
```

1 プライベート DNS ゾーンの名前。

2 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. API パブリックロードバランサーのパブリックゾーンに **api** DNS レコードを作成します。**${BASE_DOMAIN_RESOURCE_GROUP}** 変数は、パブリック DNS ゾーンがあるリソースグループをポイントする必要があります。

   a. 以下の変数をエクスポートします。

   ```
   $ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
   ```

   b. 新しいパブリックゾーンに DNS レコードを作成します。

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
   ```

   c. クラスターを既存のパブリックゾーンに追加する場合は、DNS レコードを代わりに作成できます。

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
   ```

5.10.14.1. ネットワークおよびロードバランサーの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用して、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例5.3 03_infra.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone"
      }
    }
```

```
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterPublicIpAddressName" : "[concat(parameters('baseName'), '-master-pip')]",
    "masterPublicIpAddressID" : "[resourceId('Microsoft.Network/publicIPAddresses',
variables('masterPublicIpAddressName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "masterLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('masterLoadBalancerName'))]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "internalLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('internalLoadBalancerName'))]",
    "skuName": "Standard"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('masterPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('masterPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('masterLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/publicIPAddresses/', variables('masterPublicIpAddressName'))]"
      ],
      "properties" : {
        "frontendIPConfigurations" : [
          {
            "name" : "public-lb-ip",
            "properties" : {
              "publicIPAddress" : {
                "id" : "[variables('masterPublicIpAddressID')]"
              }
            }
```

```
        }
       ],
       "backendAddressPools" : [
        {
         "name" : "public-lb-backend"
        }
       ],
       "loadBalancingRules" : [
        {
         "name" : "api-internal",
         "properties" : {
          "frontendIPConfiguration" : {
           "id" :"[concat(variables('masterLoadBalancerID'), '/frontendIPConfigurations/public-lb-
ip')]"
          },
          "backendAddressPool" : {
           "id" : "[concat(variables('masterLoadBalancerID'), '/backendAddressPools/public-lb-
backend')]"
          },
          "protocol" : "Tcp",
          "loadDistribution" : "Default",
          "idleTimeoutInMinutes" : 30,
          "frontendPort" : 6443,
          "backendPort" : 6443,
          "probe" : {
           "id" : "[concat(variables('masterLoadBalancerID'), '/probes/api-internal-probe')]"
          }
         }
        }
       ],
       "probes" : [
        {
         "name" : "api-internal-probe",
         "properties" : {
          "protocol" : "Https",
          "port" : 6443,
          "requestPath": "/readyz",
          "intervalInSeconds" : 10,
          "numberOfProbes" : 3
         }
        }
       ]
      }
     },
     {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('internalLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
       "name": "[variables('skuName')]"
      },
      "properties" : {
       "frontendIPConfigurations" : [
        {
         "name" : "internal-lb-ip",
```

```
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
              },
              "privateIPAddressVersion" : "IPv4"
            }
          }
        ],
        "backendAddressPools" : [
          {
            "name" : "internal-lb-backend"
          }
        ],
        "loadBalancingRules" : [
          {
            "name" : "api-internal",
            "properties" : {
              "frontendIPConfiguration" : {
                "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
              },
              "frontendPort" : 6443,
              "backendPort" : 6443,
              "enableFloatingIP" : false,
              "idleTimeoutInMinutes" : 30,
              "protocol" : "Tcp",
              "enableTcpReset" : false,
              "loadDistribution" : "Default",
              "backendAddressPool" : {
                "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
              },
              "probe" : {
                "id" : "[concat(variables('internalLoadBalancerID'), '/probes/api-internal-probe')]"
              }
            }
          },
          {
            "name" : "sint",
            "properties" : {
              "frontendIPConfiguration" : {
                "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
              },
              "frontendPort" : 22623,
              "backendPort" : 22623,
              "enableFloatingIP" : false,
              "idleTimeoutInMinutes" : 30,
              "protocol" : "Tcp",
              "enableTcpReset" : false,
              "loadDistribution" : "Default",
              "backendAddressPool" : {
                "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
              },
```

```
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/sint-probe')]"
            }
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 6443,
            "requestPath": "/readyz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        },
        {
          "name" : "sint-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 22623,
            "requestPath": "/healthz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "name": "[concat(parameters('privateDNSZoneName'), '/api')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
    ],
    "properties": {
      "ttl": 60,
      "aRecords": [
        {
          "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "name": "[concat(parameters('privateDNSZoneName'), '/api-int')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
```

```
      ],
      "properties": {
        "ttl": 60,
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
        ]
      }
    }
  ]
}
```

5.10.15. Azure でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Microsoft Azure で作成する必要があります。このマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供されている ARM テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

手順

1. 本トピックの **ブートストラップマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. ブートストラップマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export BOOTSTRAP_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c "files" -n "bootstrap.ign" -o tsv`
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "3.2.0" --arg url ${BOOTSTRAP_URL}
'{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
```

1. ブートストラップクラスターのブートストラップ Ignition コンテンツ。
2. 文字列としての SSH RSA 公開鍵ファイル。
3. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

5.10.15.1. ブートストラップマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例5.4 04_bootstrap.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "bootstrapIgnition" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Bootstrap ignition content for the bootstrap cluster"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string."
      }
    },
    "bootstrapVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "allowedValues" : [
        "Standard_A2",
        "Standard_A3",
        "Standard_A4",
```

```
        "Standard_A5",
        "Standard_A6",
        "Standard_A7",
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",
        "Standard_D11_v2",
        "Standard_D12_v2",
        "Standard_D13_v2",
        "Standard_D14_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
```

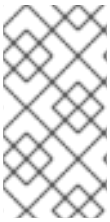
```
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the Bootstrap Virtual Machine"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "vmName" : "[concat(parameters('baseName'), '-bootstrap')]",
    "nicName" : "[concat(variables('vmName'), '-nic')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]",
    "sshPublicIpAddressName" : "[concat(variables('vmName'), '-ssh-pip')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('sshPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('sshPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "name" : "[variables('nicName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Network/publicIPAddresses', variables('sshPublicIpAddressName'))]"
      ],
      "properties" : {
```

```
        "ipConfigurations" : [
         {
           "name" : "pipConfig",
           "properties" : {
            "privateIPAllocationMethod" : "Dynamic",
            "publicIPAddress": {
              "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('sshPublicIpAddressName'))]"
            },
            "subnet" : {
              "id" : "[variables('masterSubnetRef')]"
            },
            "loadBalancerBackendAddressPools" : [
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
              },
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
              }
            ]
           }
         }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "name" : "[variables('vmName')]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('bootstrapVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmName')]",
          "adminUsername" : "core",
          "customData" : "[parameters('bootstrapIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : true,
            "ssh" : {
              "publicKeys" : [
```

```
                {
                  "path" : "[variables('sshKeyPath')]",
                  "keyData" : "[parameters('sshKeyData')]"
                }
              ]
            }
          }
        },
        "storageProfile" : {
          "imageReference": {
            "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmName'),'_OSDisk')]",
            "osType" : "Linux",
            "createOption" : "FromImage",
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : 100
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
            }
          ]
        }
      },
      {
        "apiVersion" : "2018-06-01",
        "type": "Microsoft.Network/networkSecurityGroups/securityRules",
        "name" : "[concat(variables('clusterNsgName'), '/bootstrap_ssh_in')]",
        "location" : "[variables('location')]",
        "dependsOn" : [
          "[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
        ],
        "properties": {
          "protocol" : "Tcp",
          "sourcePortRange" : "*",
          "destinationPortRange" : "22",
          "sourceAddressPrefix" : "*",
          "destinationAddressPrefix" : "*",
          "access" : "Allow",
          "priority" : 100,
          "direction" : "Inbound"
        }
      }
    ]
  }
```

5.10.16. Azure でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Microsoft Azure で作成する必要があります。これらのマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの **コントロールプレーンマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **05_masters.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. コントロールプレーンマシンのデプロイメントに必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 3
  --parameters baseName="${INFRA_ID}" 4
```

1. コントロールプレーンノード (別名マスターノード) の Ignition コンテンツ。
2. 文字列としての SSH RSA 公開鍵ファイル。
3. コントロールプレーンノードが割り当てられているプライベート DNS ゾーンの名前。
4. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

5.10.16.1. コントロールプレーンマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例5.5 05_masters.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "masterIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the master nodes"
      }
    },
    "numberOfMasters" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift masters to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone the master nodes are going to be attached to"
      }
    },
    "masterVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D8s_v3",
      "allowedValues" : [
        "Standard_A2",
        "Standard_A3",
        "Standard_A4",
        "Standard_A5",
        "Standard_A6",
        "Standard_A7",
```

```
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",
        "Standard_D11_v2",
        "Standard_D12_v2",
        "Standard_D13_v2",
        "Standard_D14_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
```

```
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the Master Virtual Machines"
      }
    },
    "diskSizeGB" : {
      "type" : "int",
      "defaultValue" : 1024,
      "metadata" : {
        "description" : "Size of the Master VM OS disk, in GB"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfMasters')]",
        "input" : "[concat(parameters('baseName'), '-master-', copyIndex('vmNames'))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "copy" : {
        "name" : "nicCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
      "location" : "[variables('location')]",
      "properties" : {
        "ipConfigurations" : [
          {
            "name" : "pipConfig",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
```

```
            },
            "loadBalancerBackendAddressPools" : [
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
              },
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
              }
            ]
          }
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/SRV",
    "name": "[concat(parameters('privateDNSZoneName'), '/_etcd-server-ssl._tcp')]",
    "location" : "[variables('location')]",
    "properties": {
      "ttl": 60,
      "copy": [{
        "name": "srvRecords",
        "count": "[length(variables('vmNames'))]",
        "input": {
          "priority": 0,
          "weight" : 10,
          "port" : 2380,
          "target" : "[concat('etcd-', copyIndex('srvRecords'), '.',
parameters('privateDNSZoneName'))]"
        }
      }]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "copy" : {
      "name" : "dnsCopy",
      "count" : "[length(variables('vmNames'))]"
    },
    "name": "[concat(parameters('privateDNSZoneName'), '/etcd-', copyIndex())]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]"
    ],
    "properties": {
      "ttl": 60,
      "aRecords": [
        {
          "ipv4Address": "[reference(concat(variables('vmNames')[copyIndex()], '-
```

```
nic')).ipConfigurations[0].properties.privateIPAddress]"
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "copy" : {
        "name" : "vmCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[variables('vmNames')[copyIndex()]]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]",
        "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'),
'/A/etcd-', copyIndex())]",
        "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'),
'/SRV/_etcd-server-ssl._tcp')]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('masterVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmNames')[copyIndex()]]",
          "adminUsername" : "core",
          "customData" : "[parameters('masterIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : true,
            "ssh" : {
              "publicKeys" : [
                {
                  "path" : "[variables('sshKeyPath')]",
                  "keyData" : "[parameters('sshKeyData')]"
                }
              ]
            }
          }
        },
        "storageProfile" : {
          "imageReference": {
            "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
            "osType" : "Linux",
```

```
            "createOption" : "FromImage",
            "caching": "ReadOnly",
            "writeAcceleratorEnabled": false,
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : "[parameters('diskSizeGB')]"
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')
[copyIndex()], '-nic'))]",
              "properties": {
                "primary": false
              }
            }
          ]
        }
      }
    }
  ]
}
```

5.10.17. ブートストラップの完了を待機し、Azure のブートストラップリソースを削除する

Microsoft Azure ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

注記

ブートストラップサーバーを削除しないと、API トラフィックがブートストラップサーバーにルーティングされるため、インストールが成功しない場合があります。

5.10.18. Azure での追加のワーカーマシンの作成

Microsoft Azure でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラススケーリングメカニズムやマシン API を利用できます。

この例では、Azure Resource Manager (ARM) テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_workers.json** というタイプのリソースを追加して起動することができます。

注記

提供される ARM テンプレートを使用してワーカーマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの **ワーカーマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **06_workers.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. ワーカーマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64 | tr -d '\n'`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
```

1. ワーカーノードの Ignition コンテンツ。
2. 文字列としての SSH RSA 公開鍵ファイル。
3. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

5.10.18.1. ワーカーマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例5.6 06_workers.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
```

```
    "metadata" : {
      "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
    }
  },
  "workerIgnition" : {
    "type" : "string",
    "metadata" : {
      "description" : "Ignition content for the worker nodes"
    }
  },
  "numberOfNodes" : {
    "type" : "int",
    "defaultValue" : 3,
    "minValue" : 2,
    "maxValue" : 30,
    "metadata" : {
      "description" : "Number of OpenShift compute nodes to deploy"
    }
  },
  "sshKeyData" : {
    "type" : "securestring",
    "metadata" : {
      "description" : "SSH RSA public key file as a string"
    }
  },
  "nodeVMSize" : {
    "type" : "string",
    "defaultValue" : "Standard_D4s_v3",
    "allowedValues" : [
      "Standard_A2",
      "Standard_A3",
      "Standard_A4",
      "Standard_A5",
      "Standard_A6",
      "Standard_A7",
      "Standard_A8",
      "Standard_A9",
      "Standard_A10",
      "Standard_A11",
      "Standard_D2",
      "Standard_D3",
      "Standard_D4",
      "Standard_D11",
      "Standard_D12",
      "Standard_D13",
      "Standard_D14",
      "Standard_D2_v2",
      "Standard_D3_v2",
      "Standard_D4_v2",
      "Standard_D5_v2",
      "Standard_D8_v3",
      "Standard_D11_v2",
      "Standard_D12_v2",
      "Standard_D13_v2",
      "Standard_D14_v2",
      "Standard_E2_v3",
```

```
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the each Node Virtual Machine"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('nodeSubnetName'))]",
    "infraLoadBalancerName" : "[parameters('baseName')]",
    "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
```

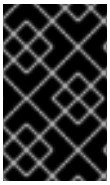
```
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfNodes')]",
        "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-',
copyIndex('vmNames', 1))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2019-05-01",
      "name" : "[concat('node', copyIndex())]",
      "type" : "Microsoft.Resources/deployments",
      "copy" : {
        "name" : "nodeCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
          "contentVersion" : "1.0.0.0",
          "resources" : [
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Network/networkInterfaces",
              "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
              "location" : "[variables('location')]",
              "properties" : {
                "ipConfigurations" : [
                  {
                    "name" : "pipConfig",
                    "properties" : {
                      "privateIPAllocationMethod" : "Dynamic",
                      "subnet" : {
                        "id" : "[variables('nodeSubnetRef')]"
                      }
                    }
                  }
                ]
              }
            },
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Compute/virtualMachines",
              "name" : "[variables('vmNames')[copyIndex()]]",
              "location" : "[variables('location')]",
              "tags" : {
                "kubernetes.io-cluster-ffranzupi": "owned"
              },
              "identity" : {
                "type" : "userAssigned",
                "userAssignedIdentities" : {
```

```
            "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
          }
        },
        "dependsOn" : [
          "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
        ],
        "properties" : {
          "hardwareProfile" : {
            "vmSize" : "[parameters('nodeVMSize')]"
          },
          "osProfile" : {
            "computerName" : "[variables('vmNames')[copyIndex()]]",
            "adminUsername" : "capi",
            "customData" : "[parameters('workerIgnition')]",
            "linuxConfiguration" : {
              "disablePasswordAuthentication" : true,
              "ssh" : {
                "publicKeys" : [
                  {
                    "path" : "[variables('sshKeyPath')]",
                    "keyData" : "[parameters('sshKeyData')]"
                  }
                ]
              }
            }
          },
          "storageProfile" : {
            "imageReference": {
              "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
            },
            "osDisk" : {
              "name": "[concat(variables('vmNames')[copyIndex()],'_OSDisk')]",
              "osType" : "Linux",
              "createOption" : "FromImage",
              "managedDisk": {
                "storageAccountType": "Premium_LRS"
              },
              "diskSizeGB": 128
            }
          },
          "networkProfile" : {
            "networkInterfaces" : [
              {
                "id" : "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('vmNames')[copyIndex()], '-nic'))]",
                "properties": {
                  "primary": true
                }
              }
            ]
          }
        }
      }
    ]
```

```
        }
      }
    }
  ]
}
```

5.10.19. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。

2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

5.10.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

5.10.21. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

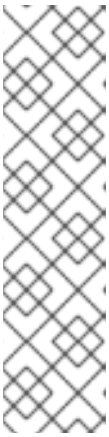
```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

5.10.22. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用して、OpenShift Container Platform クラスターを Microsoft Azure にデプロイしています。
- OpenShift CLI (**oc**) をインストールします。
- **jq** パッケージをインストールします。
- Azure CLI のインストールまたは更新を実行します。

手順

1. Ingress ルーターがロードバランサーを作成し、**EXTERNAL-IP** フィールドにデータを設定していることを確認します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.20.10  35.130.120.110
80:32288/TCP,443:31215/TCP  20
```

2. Ingress ルーター IP を変数としてエクスポートします。

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. パブリック DNS ゾーンに ***.apps** レコードを追加します。

   a. このクラスターを新しいパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

   b. このクラスターを既存のパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

4. ***.apps** レコードをプライベート DNS ゾーンに追加します。

a. 以下のコマンドを使用して ***.apps** レコードを作成します。

```
$ az network private-dns record-set a create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps --ttl 300
```

b. 以下のコマンドを使用して ***.apps** レコードをプライベート DNS ゾーンに追加します。

```
$ az network private-dns record-set a add-record -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER}
```

ワイルドカードを使用する代わりに明示的なドメインを追加する場合は、クラスターのそれぞれの現行ルートのエントリーを作成できます。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

出力例

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
grafana-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

5.10.23. ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure インストールの実行

Microsoft Azure のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる Azure インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

- クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

5.10.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

5.11. AZURE でのクラスターのアンインストール

Microsoft Azure にデプロイしたクラスターは削除することができます。

5.11.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。

- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第6章 GCP へのインストール

6.1. GCP へのインストールの準備

6.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

6.1.2. OpenShift Container Platform の GCP へのインストール要件

OpenShift Container Platform を Google Cloud Platform (GCP) にインストールする前に、サービスアカウントを作成し、GCP プロジェクトを設定する必要があります。プロジェクトの作成、API サービスの有効化、DNS の設定、GCP アカウントの制限、およびサポート対象の GCP リージョンに関する詳細は、GCP プロジェクトの設定 を参照してください。

お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、他のオプションについて、GCP の IAM の手動作成 を参照してください。

6.1.3. GCP に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

6.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる GCP インフラストラクチャーに、クラスターをインストールできます。

- **クラスターの GCP へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる GCP インフラストラクチャーに OpenShift Container Platform をインストールできます。デフォルトの設定オプションを使用して、クラスターを迅速にインストールできます。
- **カスタマイズされたクラスターの GCP へのインストール**: インストールプログラムがプロビジョニングする GCP インフラストラクチャーに、カスタマイズされたクラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の数多くのカスタマイズオプションは、インストール後 に利用できます。

- **ネットワークのカスタマイズを使用したクラスターの GCP へのインストール**: インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **ネットワークが制限された環境での GCP へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、インストーラーでプロビジョニングされる GCP インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。ミラーリングされたコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットへのアクセスが必要です。

- **クラスターの既存の Virtual Private Cloud へのインストール**: OpenShift Container Platform を既存の GCP Virtual Private Cloud (VPC) にインストールできます。このインストール方法は、新規アカウントまたはインフラストラクチャーを作成する際の制限など、会社のガイドラインによる制約がある場合に使用できます。

- **プライベートクラスターの既存の VPC へのインストール**: プライベートクラスターを既存の GCP VPC にインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

6.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングする GCP インフラストラクチャーにクラスターをインストールできます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP へのクラスターのインストール**: 独自に提供する GCP インフラストラクチャーに OpenShift Container Platform をインストールできます。提供される Deployment Manager テンプレートを使用して、インストールを支援できます。

- **GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへの共有 VPC を設定したクラスターのインストール**: 提供される Deployment Manager テンプレートを使用して、共有 VPC インフラストラクチャーに GCP リソースを作成できます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したネットワークが制限された環境での GCP へのクラスターのインストール**: ユーザーによってプロビジョニングされるインフラストラクチャーを使用して、ネットワークが制限された環境で GCP に OpenShift Container Platform をインストールできます。インストールリリースコンテンツの内部ミラーを作成することにより、ソフトウェアコンポーネントを取得するためのアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

6.1.4. 次のステップ

- GCP プロジェクトの設定

6.2. GCP プロジェクトの設定

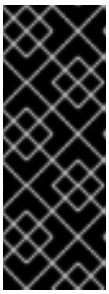
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

6.2.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

6.2.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表6.1 必要な API サービス

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**

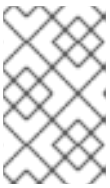
API サービス	コンソールサービス名
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

6.2.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。



注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

6.2.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表6.2 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	コンピュート	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
使用中のグローバル IP アドレス	コンピュート	グローバル	4	1
ヘルスチェック	コンピュート	グローバル	3	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	コンピュート	グローバル	2	0
静的 IP アドレス	コンピュート	リージョン	4	1
ルーター	コンピュート	グローバル	1	0
ルート	コンピュート	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	コンピュート	グローバル	3	0
CPU	コンピュート	リージョン	28	4
永続ディスク SSD (GB)	コンピュート	リージョン	896	128

注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

6.2.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

6.2.5.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表6.3 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser

アカウント	ロール
コンピュート	**roles/compute.viewer**
	roles/storage.admin

6.2.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)

- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)

6.2.7. 次のステップ

- GCP に OpenShift Container Platform クラスターをインストールします。カスタマイズされたクラスターのインストール、またはデフォルトのオプションで クラスターのクイックインストール を実行できます。

6.3. GCP の IAM の手動作成

クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境や、管理者がクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存する選択をしない場合に、クラスターのインストール前に Cloud Credential Operator (CCO) を手動モードにすることができます。

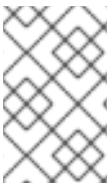
6.3.1. 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報を Kubernetes カスタムリソース定義 (CRD) として管理します。**credentialsMode** パラメーターの異なる値を **install-config.yaml** ファイルに設定し、組織のセキュリティー要件に応じて CCO を設定できます。

管理者レベルの認証情報シークレットをクラスターの **kube-system** プロジェクトに保存する選択をしない場合、OpenShift Container Platform をインストールする際に以下のいずれかのオプションを選択できます。

- **クラウド認証情報を手動で管理** します。
  CCO の **credentialsMode** パラメーターを **Manual** に設定し、クラウド認証情報を手動で管理できます。手動モードを使用すると、クラスターに管理者レベルの認証情報を保存する必要なく、各クラスターコンポーネントに必要なパーミッションのみを指定できます。お使いの環境でクラウドプロバイダーのパブリック IAM エンドポイントへの接続がない場合も、このモードを使用できます。ただし、各アップグレードについて、パーミッションを新規リリースイメージを使用して手動で調整する必要があります。また、それらを要求するすべてのコンポーネントについて認証情報を手動で指定する必要があります。

- **OpenShift Container Platform を mint モードでインストールした後に、管理者レベルの認証情報シークレットを削除** します。
  credentialsMode パラメーターが **Mint** に設定された状態で CCO を使用している場合、OpenShift Container Platform のインストール後に管理者レベルの認証情報を削除したり、ローテーションしたりできます。Mint モードは、CCO のデフォルト設定です。このオプションには、インストール時に管理者レベルの認証情報が必要になります。管理者レベルの認証情報はインストール時に、付与された一部のパーミッションと共に他の認証情報を生成するために使用されます。元の認証情報シークレットはクラスターに永続的に保存されません。



注記

z-stream 以外のアップグレードの前に、認証情報のシークレットを管理者レベルの認証情報と共に元に戻す必要があります。認証情報が存在しない場合は、アップグレードがブロックされる可能性があります。

関連情報

- クラウドプロバイダーの認証情報のローテーションまたは削除

利用可能なすべての CCO 認証情報モードとそれらのサポートされるプラットフォームの詳細については、Cloud Credential Operator について 参照してください。

6.3.2. IAM の手動作成

Cloud Credential Operator (CCO) は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、**install-config.yaml** ファイルを作成します。

```
$ openshift-install create install-config --dir <installation_directory>
```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

3. マニフェストを生成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ openshift-install create manifests --dir <installation_directory>
```

4. インストールプログラムが含まれるディレクトリーから、**openshift-install** バイナリーがビルドされる OpenShift Container Platform リリースイメージの詳細を取得します。

```
$ openshift-install version
```

   出力例

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. このリリースイメージ内で、デプロイするクラウドをターゲットとする **CredentialsRequest** オブジェクトをすべて特定します。

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 --
credentials-requests --cloud=gcp
```

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

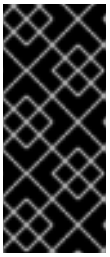
サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-gcs
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
```

6. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

7. インストールプログラムが含まれるディレクトリーから、クラスターの作成に進みます。

```
$ openshift-install create cluster --dir <installation_directory>
```



重要

手動でメンテナンスされる認証情報を使用するクラスターをアップグレードする前に、CCO がアップグレード可能な状態であることを確認します。詳細は、クラウドプロバイダーのインストールコンテンツの手動でメンテナンスされる認証情報を使用したクラスターのアップグレードについてのセクションを参照してください。

6.3.3. 手動でメンテナンスされた認証情報を使用したクラスターのアップグレード

手動でメンテナンスされる認証情報をを含むクラスターの Cloud Credential Operator (CCO) の **upgradable** ステータスはデフォルトで **false** となります。

- 4.7 から 4.8 などのマイナーリリースの場合には、このステータスを使用することで、パーミッションを更新して **CloudCredential** リソースにアノテーションを付けてパーミッションが次のバージョンの要件に合わせて更新されていることを指定するまで、アップグレードができないようになります。このアノテーションは、**Upgradable** ステータスを **True** に変更します。

- 4.8.9 から 4.8.10 などの z-stream リリースの場合には、パーミッションは追加または変更されないため、アップグレードはブロックされません。

手動でメンテナンスされる認証情報でクラスターをアップグレードする前に、アップグレードするリリースイメージ用に認証情報を新規作成する必要があります。さらに、既存の認証情報に必要なパーミッションを確認し、これらのコンポーネントの新規リリースの新しいパーミッション要件に対応する必要があります。

手順

1. 新規リリースの **CredentialsRequest** カスタムリソースを抽出して検査します。
クラウドプロバイダーのインストールコンテンツの IAM の手動作成についてのセクションでは、クラウドに必要な認証情報を取得し、使用する方法について説明します。

2. クラスターで手動でメンテナンスされる認証情報を更新します。

   - 新規リリースイメージによって追加される **CredentialsRequest** カスタムリソースの新規のシークレットを作成します。

   - シークレットに保存される既存の認証情報の **CredentialsRequest** カスタムリソースにパーミッション要件を変更した場合は、必要に応じてパーミッションを更新します。

3. 新規リリースですべてのシークレットが正しい場合は、クラスターをアップグレードする準備が整っていることを示します。

   a. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

   b. **CloudCredential** リソースを編集して、**metadata** フィールドに **upgradeable-to** アノテーションを追加します。

   ```
   $ oc edit cloudcredential cluster
   ```

   追加するテキスト

   ```
   ...
     metadata:
       annotations:
         cloudcredential.openshift.io/upgradeable-to: <version_number>
   ...
   ```

   ここで、**<version_number>** は、アップグレードするバージョン (**x.y.z** 形式) に置き換えます。例: OpenShift Container Platform **4.8.2** (OpenShift Container Platform 4.8.2 の場合)

   アノテーションを追加してから、upgradeable のステータスが変更されるまで、数分かかる場合があります。

4. CCO がアップグレードできることを確認します。

   a. Web コンソールの **Administrator** パースペクティブで、**Administration → Cluster Settings** に移動します。

   b. CCO ステータスの詳細を表示するには、**Cluster Operators** 一覧で **cloud-credential** をクリックします。

c. **Conditions** セクションの **Upgradeable** ステータスが **False** の場合に、**upgradeable-to** アノテーションに間違いがないことを確認します。

Conditions セクションの **Upgradeable** ステータスが **True** の場合には、OpenShift Container Platform のアップグレードを開始できます。

6.3.4. mint モード

mint モードは、OpenShift Container Platform をサポートするプラットフォーム上の OpenShift Container Platform のデフォルトの Cloud Credential Operator (CCO) クレデンシャルモードです。このモードでは、CCO は提供される管理者レベルのクラウド認証情報を使用してクラスターを実行します。Mint モードは AWS と GCP でサポートされています。

mint モードでは、**admin** 認証情報は **kube-system** namespace に保存され、次に CCO によってクラスターの **CredentialsRequest** オブジェクトを処理し、特定のパーミッションでそれぞれのユーザーを作成するために使用されます。

mint モードには以下の利点があります。

- 各クラスターコンポーネントにはそれぞれが必要なパーミッションのみがあります。
- クラウド認証情報の自動の継続的な調整が行われます。これには、アップグレードに必要になる可能性のある追加の認証情報またはパーミッションが含まれます。

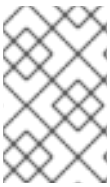
1 つの不利な点として、mint モードでは、**admin** 認証情報がクラスターの **kube-system** シークレットに保存される必要があります。

6.3.5. 管理者レベルの認証情報の削除またはローテーション機能を持つ mint モード

現時点で、このモードは AWS および GCP でのみサポートされます。

このモードでは、ユーザーは通常の mint モードと同様に管理者レベルの認証情報を使用して OpenShift Container Platform をインストールします。ただし、このプロセスはクラスターのインストール後の管理者レベルの認証情報シークレットを削除します。

管理者は、Cloud Credential Operator に読み取り専用の認証情報について独自の要求を行わせることができます。これにより、すべての **CredentialsRequest** オブジェクトに必要なパーミッションがあることの確認が可能になります。そのため、いずれかの変更が必要にならない限り、管理者レベルの認証情報は必要になりません。関連付けられた認証情報が削除された後に、必要な場合は、これは基礎となるクラウドで破棄するか、または非アクティブにできます。



注記

z-stream 以外のアップグレードの前に、認証情報のシークレットを管理者レベルの認証情報と共に元に戻す必要があります。認証情報が存在しない場合は、アップグレードがブロックされる可能性があります。

管理者レベルの認証情報はクラスターに永続的に保存されません。

これらの手順を実行するには、短い期間にクラスターでの管理者レベルの認証情報が必要になります。また、アップグレードごとに管理者レベルの認証情報を使用してシークレットを手動で再インストールする必要があります。

6.3.6. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトオプションを使用した クラスターの GCP へのクイックインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのクラウドのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール

6.4. GCP へのクラスターのクイックインストール

OpenShift Container Platform バージョン 4.8 では、デフォルトの設定オプションを使用する Google Cloud Platform (GCP) にクラスターをインストールできます。

6.4.1. 前提条件

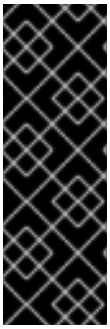
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.4.3. クラスターノードの SSH アクセス用のキーペアの生成

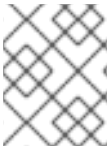
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.4.4. インストールプログラムの取得

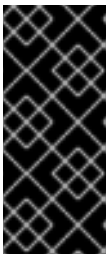
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

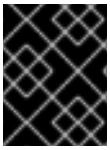
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.4.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

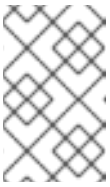
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **gcp** を選択します。

c. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

d. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

e. クラスターをデプロイするリージョンを選択します。

f. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

g. クラスターの記述名を入力します。7 文字以上の名前を指定すると、クラスター名から生成されるインフラストラクチャー ID で最初の 6 文字のみが使用されます。

h. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

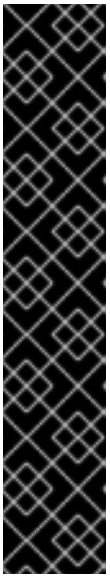
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

6.4.6. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.4.7. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

6.4.8. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.4.9. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.5. カスタマイズによる GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、インストールプログラムが Google Cloud Platform (GCP) にプロビジョニングするインフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

6.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.5.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

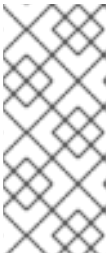
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

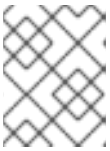
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.5.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

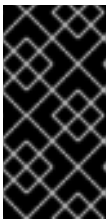
- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.5.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

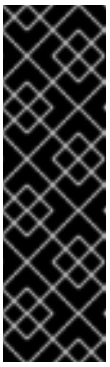
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.5.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

6.5.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表6.4 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

6.5.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表6.5 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

6.5.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表6.6 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

6.5.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表6.7 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.region	クラスターをホストする GCP リージョンの名前。	有効なリージョン名 (例: **us-central1**)。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.licenses	コンピューティングイメージに適用する必要があるライセンス URL のリスト。  **重要** **license** パラメーターは非推奨のフィールドであり、ネストされた仮想化はデフォルトで有効になっています。このフィールドの使用は推奨されません。	ネストされた仮想化 を有効にするライセンスなど、ライセンス API で使用可能なすべてのライセンス。このパラメーターは、ビルド済みのイメージを生成するメカニズムでは使用できません。ライセンス URL を使用すると、インストーラーは使用前にソースイメージをコピーする必要があります。

パラメーター	説明	値
platform.gcp.osDisk.diskSizeGB	ディスクのサイズ (GB 単位)。	16 GB から 65536 GB の間のサイズ
platform.gcp.osDisk.diskType	ディスクのタイプ。	デフォルトの **pd-ssd** または **pd-standard** ディスクタイプ。コントロールプレーンノードは **pd-ssd** ディスクタイプである必要があります。ワーカーノードはいずれのタイプでも構いません。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールプレーンマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コントロールプレーンマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	コントロールプレーンマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コントロールプレーンマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コンピュートマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	コンピュートマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コンピュートマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

6.5.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
```

```
        encryptionKey: 5
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: 6 7
- hyperthreading: Enabled 8
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 9
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
pullSecret: '{"auths": ...}' 13
fips: false 14
sshKey: ssh-ed25519 AAAA... 15
```

1 10 11 12 13 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

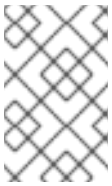
5 9 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントの適切なパーミッションを付与する方法についての詳細は、マシン管理 → マシンセットの作成 → GCP でのマシンセットの作成を参照してください。

14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

15 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- マシンセットの顧客管理の暗号鍵の有効化

6.5.5.3. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。

custom-<number_of_cpus>-<amount_of_memory_in_mb>

たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプを含む install-config.yaml ファイルのサンプル

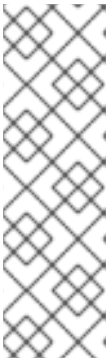
```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

6.5.5.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

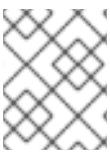
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.5.6. GCP Marketplace イメージの使用

GCP Marketplace イメージを使用して OpenShift Container Platform クラスターをデプロイする場合は、マニフェストを作成し、コンピュートマシンセットの定義を編集して GCP Marketplace イメージを指定する必要があります。

前提条件

- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットがある。

手順

1. 以下のコマンドを実行して、インストールマニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_dir>
```

2. 以下のファイルを見つけます。
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-0.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-1.yaml**
   - **<installation_dir>/openshift/99_openshift-cluster-api_worker-machineset-2.yaml**
3. 各ファイルで、**.spec.template.spec.providerSpec.value.disks[0].image** プロパティーを編集して、使用するサービスを参照します。

   OpenShift Container Platform

   projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145

   OpenShift Platform Plus

   projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145

   OpenShift Kubernetes Engine

   projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145

GCP Marketplace イメージを使用したコンピュートマシンセットの例

```
deletionProtection: false
disks:
- autoDelete: true
  boot: true
  image: projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-
202210040145
  labels: null
  sizeGb: 128
  type: pd-ssd
kind: GCPMachineProviderSpec
machineType: n2-standard-4
```

6.5.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

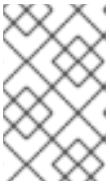
- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

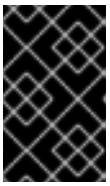
```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

6.5.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.5.9. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

6.5.10. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.5.11. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.6. ネットワークのカスタマイズによる GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、インストールプログラムが Google Cloud Platform (GCP) にプロビジョニングするインフラストラクチャーにカスタマイズされたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

6.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.6.3. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.6.4. インストールプログラムの取得

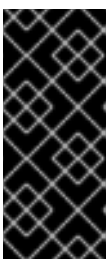
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

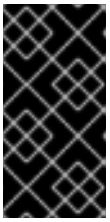
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.6.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

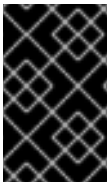
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.6.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

6.6.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表6.8 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

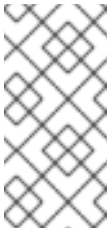
pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

6.6.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表6.9 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

6.6.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表6.10 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

6.6.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表6.11 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。

パラメーター	説明	値
platform.gcp.region	クラスターをホストする GCP リージョンの名前。	有効なリージョン名 (例: **us-central1**)。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.licenses	コンピューティングイメージに適用する必要があるライセンス URL のリスト。 **重要** **license** パラメーターは非推奨のフィールドであり、ネストされた仮想化はデフォルトで有効になっています。このフィールドの使用は推奨されません。	ネストされた仮想化 を有効にするライセンスなど、ライセンス API で使用可能なすべてのライセンス。このパラメーターは、ビルド済みのイメージを生成するメカニズムでは使用できません。ライセンス URL を使用すると、インストーラーは使用前にソースイメージをコピーする必要があります。
platform.gcp.osDisk.diskSizeGB	ディスクのサイズ (GB 単位)。	16 GB から 65536 GB の間のサイズ

パラメーター	説明	値
platform.gcp.osDisk.diskType	ディスクのタイプ。	デフォルトの **pd-ssd** または **pd-standard** ディスクタイプ。コントロールプレーンノードは **pd-ssd** ディスクタイプである必要があります。ワーカーノードはいずれのタイプでも構いません。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールプレーンマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コントロールプレーンマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	コントロールプレーンマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。

パラメーター	説明	値
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コントロールプレーンマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コンピュートマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	コンピュートマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。

パラメーター	説明	値
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コンピュートマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

6.6.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
```

```
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 9
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 12
    region: us-central1 13
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
```

1 10 12 13 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 11 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

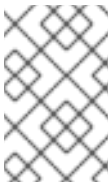
5 9 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

6.6.6. 関連情報

- マシンセットの顧客管理の暗号鍵の有効化

6.6.6.1. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプを含む install-config.yaml ファイルのサンプル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
```

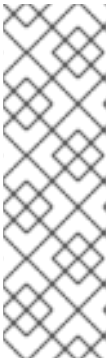
```
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

6.6.6.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

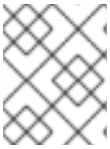
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.6.7. ネットワーク設定フェーズ

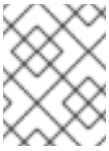
OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**

これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

6.6.8. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

6.6.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

6.6.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表6.12 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

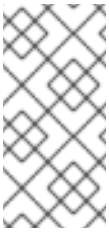
フィールド	タイプ	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表6.13 **defaultNetwork** オブジェクト

フィールド	タイプ	説明

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表6.14 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	タイプ	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表6.15 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 暗号化を有効にするために空のオブジェクトを指定します。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表6.16 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

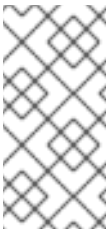
フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表6.17 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

6.6.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、以下を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

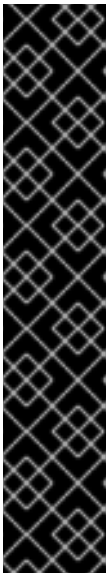
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

6.6.11. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.6.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

6.6.13. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

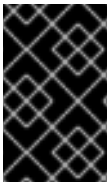
- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.6.14. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.7. ネットワークが制限された環境での GCP へのクラスターのインストール

OpenShift Container Platform 4.8 では、既存の Google Virtual Private Cloud (VPC) でインストールリリースコンテンツの内部ミラーを作成して、クラスターをネットワークが制限された環境で Google Cloud Platform (GCP) にインストールできます。



重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットアクセスが必要になります。

6.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得しています。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- GCP に既存の VPC がある。インストーラーでプロビジョニングされるインフラストラクチャーを使用するネットワークが制限された環境にクラスターをインストールする場合は、インストーラーでプロビジョニングされる VPC を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - ミラーレジストリーが含まれる。
  - 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。他のサイトへのアクセスを付与する必要がある場合もありますが、***.googleapis.com** および **accounts.google.com** へのアクセスを付与する必要があります。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.7.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

6.7.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

6.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を

無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.7.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

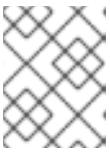
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

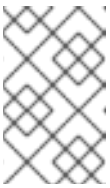
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.7.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既存のサブネットを指定します。

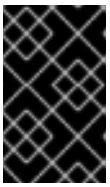
d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.7.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

6.7.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表6.18 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

6.7.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表6.19 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

6.7.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

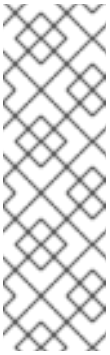
表6.20 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

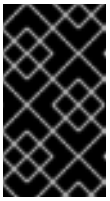
パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

6.7.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表6.21 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.region	クラスターをホストする GCP リージョンの名前。	有効なリージョン名 (例: **us-central1**)。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。

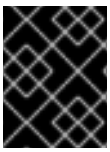
パラメーター	説明	値
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.licenses	コンピューティングイメージに適用する必要があるライセンス URL のリスト。  **重要** **license** パラメーターは非推奨のフィールドであり、ネストされた仮想化はデフォルトで有効になっています。このフィールドの使用は推奨されません。	ネストされた仮想化 を有効にするライセンスなど、ライセンス API で使用可能なすべてのライセンス。このパラメーターは、ビルド済みのイメージを生成するメカニズムでは使用できません。ライセンス URL を使用すると、インストーラーは使用前にソースイメージをコピーする必要があります。
platform.gcp.osDisk.diskSizeGB	ディスクのサイズ (GB 単位)。	16 GB から 65536 GB の間のサイズ
platform.gcp.osDisk.diskType	ディスクのタイプ。	デフォルトの **pd-ssd** または **pd-standard** ディスクタイプ。コントロールプレーンノードは **pd-ssd** ディスクタイプである必要があります。ワーカーノードはいずれのタイプでも構いません。

パラメーター	説明	値
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールプレーンマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コントロールプレーンマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	コントロールプレーンマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コントロールプレーンマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

パラメーター	説明	値
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コンピュートマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	コンピュートマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コンピュートマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

6.7.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
        encryptionKey: (5)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: (6) (7)
- hyperthreading: Enabled (8)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: (9)
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
```

```
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
    network: existing_vpc 13
    controlPlaneSubnet: control_plane_subnet 14
    computeSubnet: compute_subnet 15
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
additionalTrustBundle: | 19
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 20
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 10 11 12 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

5 9 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカ

ウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントの適切なパーミッションを付与する方法についての詳細は、マシン管理 → マシンセットの作成 → GCP でのマシンセットの作成を参照してください。

13 既存 VPC の名前を指定します。

14 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

15 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

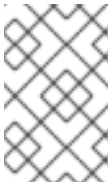
16 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

17 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

18 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

19 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

20 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

6.7.5.3. GCP にグローバルにアクセスできる Ingress コントローラーの作成

Google Cloud Platform (GCP) クラスターにグローバルにアクセスできる Ingress コントローラーを作成できます。グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

前提条件

- **install-config.yaml** を作成し、これに対する変更を完了している。

手順

グローバルアクセスが設定された Ingress コントローラーの新規の GCP クラスターへの作成

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

2. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

サンプル clientAccess 設定を Global に設定します。

```
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1 **gcp.clientAccess** を **Global** に設定します。

2 グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

6.7.5.4. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
custom-<number_of_cpus>-<amount_of_memory_in_mb>

たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプを含む install-config.yaml ファイルのサンプル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

6.7.5.5. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.7.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

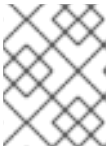
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

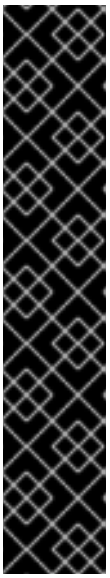
```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



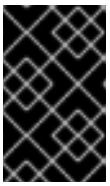
注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

6.7.7. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.7.8. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.7.9. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェ

クトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

6.7.10. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.7.11. 次のステップ

- インストールを検証 します。
- クラスターをカスタマイズします。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.8. クラスターの GCP の既存 VPC へのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを Google Cloud Platform (GCP) の既存の Virtual Private Cloud (VPC) にインストールできます。インストールプログラムは、さらなるカス

タマイズが可能な必要なインフラストラクチャーの残りをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

6.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.8.2. カスタム VPC の使用について

OpenShift Container Platform 4.8 では、Google Cloud Platform (GCP) の既存の Virtual Private Cloud (VPC) 内の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の GCP VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。サブネットのネットワークを設定する必要があります。

6.8.2.1. VPC を使用するための要件

VPC CIDR ブロックとマシンネットワーク CIDR の組み合わせは、空であってはなりません。サブネットはマシンネットワーク内にある必要があります。

インストールプログラムでは、次のコンポーネントは作成されません。

- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC ネットワーク

注記

インストールプログラムでは、クラウド提供の DNS サーバーを使用する必要があります。カスタム DNS サーバーの使用はサポートされていないため、インストールが失敗します。

6.8.2.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- コントロールプレーンマシン用に 1 つのサブネットを提供し、コンピューティングマシン用に 1 つのサブネットを提供します。
- サブネットの CIDR は指定されたマシン CIDR に属します。

6.8.2.3. パーミッションの区分

一部の個人は、クラウド内に他のリソースとは異なるリソースを作成できます。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

6.8.2.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体で許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

6.8.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.8.4. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して **~/.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、**~/.ssh/id_rsa** および **~/.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

   a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.8.5. インストールプログラムの取得

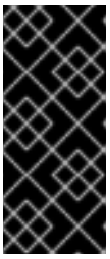
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.8.6. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

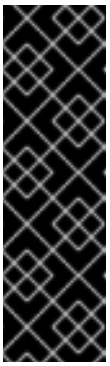
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

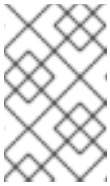
   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



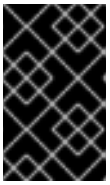
注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。
      vii. クラスターの記述名を入力します。
      viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

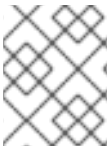


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

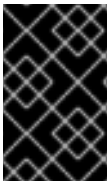
6.8.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

6.8.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表6.22 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

6.8.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表6.23 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

6.8.6.1.3. オプションの設定パラメーター

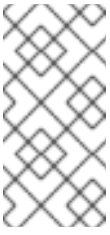
オプションのインストール設定パラメーターは、以下の表で説明されています。

表6.24 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

6.8.6.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表6.25 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.region	クラスターをホストする GCP リージョンの名前。	有効なリージョン名 (例: **us-central1**)。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.licenses	コンピューティングイメージに適用する必要があるライセンス URL のリスト。 **重要** **license** パラメーターは非推奨のフィールドであり、ネストされた仮想化はデフォルトで有効になっています。このフィールドの使用は推奨されません。	ネストされた仮想化 を有効にするライセンスなど、ライセンス API で使用可能なすべてのライセンス。このパラメーターは、ビルド済みのイメージを生成するメカニズムでは使用できません。ライセンス URL を使用すると、インストーラーは使用前にソースイメージをコピーする必要があります。

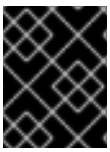
パラメーター	説明	値
platform.gcp.osDisk.diskSizeGB	ディスクのサイズ (GB 単位)。	16 GB から 65536 GB の間のサイズ
platform.gcp.osDisk.diskType	ディスクのタイプ。	デフォルトの **pd-ssd** または **pd-standard** ディスクタイプ。コントロールプレーンノードは **pd-ssd** ディスクタイプである必要があります。ワーカーノードはいずれのタイプでも構いません。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールプレーンマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コントロールプレーンマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	コントロールプレーンマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コントロールプレーンマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コンピュートマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	コンピュートマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コンピュートマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

6.8.6.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
```

```
        encryptionKey: 5
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: 6 7
- hyperthreading: Enabled 8
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 9
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
    network: existing_vpc 13
    controlPlaneSubnet: control_plane_subnet 14
    computeSubnet: compute_subnet 15
pullSecret: '{"auths": ...}' 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
```

1 10 11 12 16 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行は

ハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

4 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

5 9 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントの適切なパーミッションを付与する方法についての詳細は、マシン管理 → マシンセットの作成 → GCP でのマシンセットの作成を参照してください。

13 既存 VPC の名前を指定します。

14 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

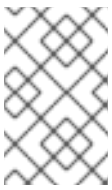
15 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

17 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

18 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

6.8.6.3. GCP にグローバルにアクセスできる Ingress コントローラーの作成

Google Cloud Platform (GCP) クラスターにグローバルにアクセスできる Ingress コントローラーを作成できます。グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

前提条件

- **install-config.yaml** を作成し、これに対する変更を完了している。

手順

グローバルアクセスが設定された Ingress コントローラーの新規の GCP クラスターへの作成

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

2. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

サンプル clientAccess 設定を Global に設定します。

```
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1 **gcp.clientAccess** を **Global** に設定します。

2 グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

6.8.7. 関連情報

- マシンセットの顧客管理の暗号鍵の有効化

6.8.7.1. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプを含む install-config.yaml ファイルのサンプル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

6.8.7.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

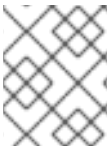
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.8.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

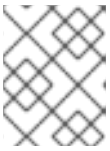
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

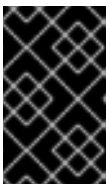


注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

重要

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



重要

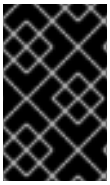
インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。

- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

6.8.9. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
   PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.8.10. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

6.8.11. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.8.12. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.9. GCP へのプライベートクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、プライベートクラスターを Google Cloud Platform (GCP) の既存の VPC にインストールできます。インストールプログラムは、さらなるカスタマイズが可能な必要なインフラストラクチャーの残りをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

6.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

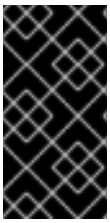
- クラスターをホストするように GCP プロジェクトを設定 している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.9.2. プライベートクラスター

外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

<<<<<<< HEAD デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。



重要

クラスターにパブリックサブネットがある場合、管理者により作成されたロードバランサーサービスはパブリックにアクセスできる可能性があります。クラスターのセキュリティーを確保するには、これらのサービスに明示的にプライベートアノテーションが付けられていることを確認してください。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

6.9.2.1. GCP のプライベートクラスター

Google Cloud Platform (GCP) にプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

クラスターには、GCP API にアクセスするためにインターネットへのアクセスが依然として必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックネットワークロードバランサー
- クラスターの **baseDomain** に一致するパブリック DNS ゾーン

インストールプログラムは、プライベート DNS ゾーンおよびクラスターに必要なレコードを作成するために指定する **baseDomain** を使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

ソースタグに基づいて外部ロードバランサーへのアクセスを制限できないため、プライベートクラスターは内部ロードバランサーのみを使用して内部インスタンスへのアクセスを許可します。

内部ロードバランサーは、ネットワークロードバランサーが使用するターゲットプールではなく、インスタンスグループに依存します。インストールプログラムは、グループにインスタンスがない場合でも、各ゾーンのインスタンスグループを作成します。

- クラスター IP アドレスは内部のみで使用されます。
- 1 つの転送ルールが Kubernetes API およびマシン設定サーバーポートの両方を管理します。
- バックエンドサービスは各ゾーンのインスタンスグループ、および存在する場合はブートストラップインスタンスグループで設定されます。
- ファイアウォールは、内部のソース範囲のみに基づく単一ルールを使用します。

6.9.2.1.1. 制限事項

ロードバランサーの機能の違いにより、マシン設定サーバー **/healthz** のヘルスチェックは実行されません。2 つの内部ロードバランサーが 1 つの IP アドレスを共有できませんが、2 つのネットワークロードバランサーは 1 つの外部 IP アドレスを共有できます。インスタンスが健全であるかどうかについては、ポート 6443 の **/readyz** チェックで完全に判別されます。

6.9.3. カスタム VPC の使用について

OpenShift Container Platform 4.8 では、クラスターを Google Cloud Platform (GCP) の既存の VPC にデプロイできます。これを実行する場合、VPC 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の GCP VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

6.9.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなります。

- VPC
- サブネット
- Cloud Router
- Cloud NAT
- NAT IP アドレス

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP など

の VPC オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

VPC およびサブネットは以下の要件を満たす必要があります。

- VPC は、OpenShift Container Platform クラスターをデプロイする同じ GCP プロジェクトに存在する必要があります。
- コントロールプレーンおよびコンピュートマシンからインターネットにアクセスできるようにするには、サブネットで Cloud NAT を設定してこれに対する egress を許可する必要があります。これらのマシンにパブリックアドレスがありません。インターネットへのアクセスが必要ない場合でも、インストールプログラムおよびイメージを取得できるように VPC ネットワークに対して egress を許可する必要があります。複数の Cloud NAT を共有サブネットで設定できないため、インストールプログラムはこれを設定できません。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定するすべてのサブネットが存在し、指定した VPC に属します。
- サブネットの CIDR はマシン CIDR に属します。
- クラスターのコントロールプレーンおよびコンピュートマシンをデプロイするためにサブネットを指定する必要があります。両方のマシンタイプに同じサブネットを使用できます。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。

6.9.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する GCP の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

6.9.3.3. クラスター間の分離

OpenShift Container Platform を既存ネットワークにデプロイする場合、クラスターサービスの分離は、クラスターのインフラストラクチャー ID によるクラスター内のマシンを参照するファイアウォールルールによって保持されます。クラスター内のトラフィックのみが許可されます。

複数のクラスターを同じ VPC にデプロイする場合、以下のコンポーネントはクラスター間のアクセスを共有する可能性があります。

- API: 外部公開ストラテジーでグローバルに利用可能か、または内部公開ストラテジーのネットワーク全体で利用できる。
- デバッグツール: SSH および ICMP アクセス用にマシン CIDR に対して開かれている仮想マシンインスタンス上のポートなど。

6.9.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.9.5. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

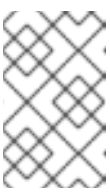
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

5. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

6. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.9.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.9.7. インストール設定ファイルの手動作成

内部ネットワークからのみアクセスでき、インターネット上に表示されないプライベート OpenShift Container Platform クラスターのインストールの場合、インストール設定ファイルを手動で生成する必要があります。

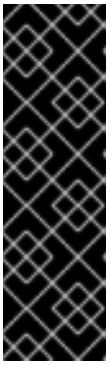
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

6.9.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

6.9.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表6.26 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

6.9.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表6.27 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

6.9.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表6.28 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

6.9.7.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表6.29 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.region	クラスターをホストする GCP リージョンの名前。	有効なリージョン名 (例: **us-central1**)。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.licenses	コンピューティングイメージに適用する必要があるライセンス URL のリスト。  **重要** **license** パラメーターは非推奨のフィールドであり、ネストされた仮想化はデフォルトで有効になっています。このフィールドの使用は推奨されません。	ネストされた仮想化 を有効にするライセンスなど、ライセンス API で使用可能なすべてのライセンス。このパラメーターは、ビルド済みのイメージを生成するメカニズムでは使用できません。ライセンス URL を使用すると、インストーラーは使用前にソースイメージをコピーする必要があります。

パラメーター	説明	値
platform.gcp.osDisk.diskSizeGB	ディスクのサイズ (GB 単位)。	16 GB から 65536 GB の間のサイズ
platform.gcp.osDisk.diskType	ディスクのタイプ。	デフォルトの **pd-ssd** または **pd-standard** ディスクタイプ。コントロールプレーンノードは **pd-ssd** ディスクタイプである必要があります。ワーカーノードはいずれのタイプでも構いません。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールプレーンマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コントロールプレーンマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.location	コントロールプレーンマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
controlPlane.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コントロールプレーンマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID
compute.platform.gcp.osDisk.encryptionKey.kmsKey.name	コントロールマシンのディスク暗号化に使用されるお客様が管理する暗号化キーの名前。	暗号化キー名。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.keyRing	コンピュートマシンの場合、KMS キーが属する KMS キーリングの名前。	KMS キーリング名。

パラメーター	説明	値
compute.platform.gcp.osDisk.encryptionKey.kmsKey.location	コンピュートマシンの場合、キーリングが存在する GCP の場所。KMS の場所についての詳細は、Google ドキュメントの Cloud KMS locations を参照してください。	キーリングの GCP の場所。
compute.platform.gcp.osDisk.encryptionKey.kmsKey.projectID	コンピュートマシンの場合、KMS キーリングが存在するプロジェクトの ID。設定されていない場合、この値は VM プロジェクト ID にデフォルト設定されます。	GCP プロジェクト ID

6.9.7.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
```

```
        encryptionKey: 5
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
compute: 6 7
- hyperthreading: Enabled 8
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 9
          kmsKey:
            name: worker-key
            keyRing: test-machine-keys
            location: global
            projectID: project-id
  replicas: 3
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 11
    region: us-central1 12
    network: existing_vpc 13
    controlPlaneSubnet: control_plane_subnet 14
    computeSubnet: compute_subnet 15
pullSecret: '{"auths": ...}' 16
fips: false 17
sshKey: ssh-ed25519 AAAA... 18
publish: Internal 19
```

1 10 11 12 16 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行は

4 8 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

5 9 オプション: 仮想マシンと永続ボリュームの両方を暗号化するカスタム暗号化キーセクション。デフォルトのコンピュートサービスアカウントには、KMS キーを使用するためのパーミッションが付与され、適切な IAM ロールが割り当てられている必要があります。デフォルトのサービスアカウント名は、**service-<project_number>@compute-system.iam.gserviceaccount.com** パターンをベースにしています。サービスアカウントの適切なパーミッションを付与する方法についての詳細は、マシン管理 → マシンセットの作成 → GCP でのマシンセットの作成を参照してください。

13 既存 VPC の名前を指定します。

14 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

15 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

17 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

18 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

19 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

6.9.7.3. GCP にグローバルにアクセスできる Ingress コントローラーの作成

Google Cloud Platform (GCP) クラスターにグローバルにアクセスできる Ingress コントローラーを作成できます。グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

前提条件

- **install-config.yaml** を作成し、これに対する変更を完了している。

手順

グローバルアクセスが設定された Ingress コントローラーの新規の GCP クラスターへの作成

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

出力例

```
cluster-ingress-default-ingresscontroller.yaml
```

2. エディターで **cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、必要な Operator 設定を記述するカスタムリソース (CR) を入力します。

サンプル clientAccess 設定を Global に設定します。

```
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
        type: GCP
      scope: Internal 2
    type: LoadBalancerService
```

1 **gcp.clientAccess** を **Global** に設定します。

2 グローバルアクセスは、内部ロードバランサーを使用する Ingress コントローラーでのみ利用できます。

6.9.8. 関連情報

- マシンセットの顧客管理の暗号鍵の有効化

6.9.8.1. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

インストールプロセスの一環として、カスタムマシンタイプを **install-config.yaml** ファイルで指定します。

カスタムマシンタイプを含む install-config.yaml ファイルのサンプル

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

6.9.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

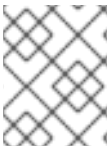
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.9.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、以下を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

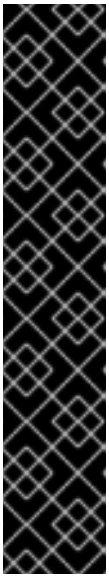
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
```

```
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

6.9.10. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.9.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

6.9.12. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.9.13. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.10. DEPLOYMENT MANAGER テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、独自に提供するインフラストラクチャーを使用する Google Cloud Platform (GCP) にクラスターをインストールできます。

以下に、ユーザーによって提供されるインフラストラクチャーのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

6.10.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

6.10.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

6.10.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.10.4. GCP プロジェクトの設定

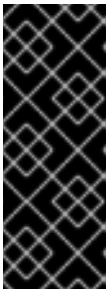
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

6.10.4.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

6.10.4.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表6.30 必要な API サービス

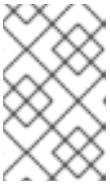
API サービス	コンソールサービス名
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

6.10.4.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。



注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

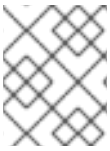
6.10.4.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表6.31 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
ヘルスチェック	コンピュート	グローバル	2	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0



注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**

- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

6.10.4.5. GCP でのサービスアカウントの作成

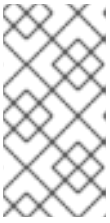
OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。



注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

6.10.4.5.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor
- サービスアカウントキー管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表6.32 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

6.10.4.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)

6.10.4.7. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
   GCP ドキュメントで、サービスアカウントでの認証 について参照してください。

6.10.5. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

6.10.5.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
```

```
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の /**var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

6.10.5.2. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。
      vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

c. オプション: クラスターでコンピュートマシンをプロビジョニングするよう設定する必要がない場合は、 **install-config.yaml** ファイルで **compute** プールの **replicas** を **0** に設定してコンピュートプールを空にします。

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 **0** に設定します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.10.5.3. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

6.10.5.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対す

る呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

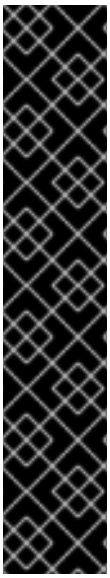
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.10.5.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- オプション: Ingress DNS レコードの追加

6.10.6. 一般的な変数のエクスポート

6.10.6.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な GCP リソースを見つけるためにも使用されます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

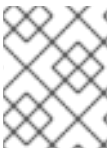
出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

6.10.6.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

6.10.7. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/17**)。
4. **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.128.0/17**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

6.10.7.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.101_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
```

```
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

6.10.8. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

6.10.8.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

6.10.8.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表6.33 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット

プロトコル	ポート	説明
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表6.34 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表6.35 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

6.10.9. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** として

コンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。

3. デプロイメントテンプレートが使用する変数をエクスポートします。

   a. クラスターネットワークの場所をエクスポートします。

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
   ```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

3. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
4. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
5. **control_subnet** は、コントロールサブセットの URL です。
6. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

6.10.9.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例6.2 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
```

```
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

6.10.9.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例6.3 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
```

```
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

6.10.10. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。
   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

6.10.10.1. プライベート DNS の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例6.4 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

6.10.11. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

6.10.11.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例6.5 03_firewall.py Deployment Manager テンプレート

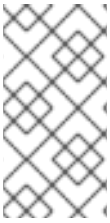
```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
```

```
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

6.10.12. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
```

```
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' 1
EOF
```

1 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コンピュートマシンをホストするサブネットの変数をエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

-

6.10.12.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例6.6 03_iam.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

6.10.13. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE="gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz"
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

6.10.14. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。
4. **cluster_network** はクラスターネットワークの **selfLink** URL です。
5. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
6. **image** は RHCOS イメージの **selfLink** URL です。
7. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
8. **root_volume_size** はブートストラップマシンのブートディスクサイズです。
9. **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager の制限によりテンプレートではロードバランサーのメンバーシップを管理しないため、ブートストラップマシンは手動で追加する必要があります。

   a. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

   ```
   $ gcloud compute instance-groups unmanaged add-instances \
     ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
   bootstrap
   ```

   b. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

   ```
   $ gcloud compute backend-services add-backend \
     ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
   group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
   ```

6.10.14.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例6.7 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
```

```
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet'],
            'accessConfigs': [{
                'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
            }]
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-bootstrap'
            ]
        },
        'zone': context.properties['zone']
    }
}, {
    'name': context.properties['infra_id'] + '-bootstrap-instance-group',
    'type': 'compute.v1.instanceGroup',
    'properties': {
        'namedPorts': [
            {
                'name': 'ignition',
                'port': 22623
            }, {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

6.10.15. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)

    ignition: '${MASTER_IGNITION}' (7)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。

(3) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(4) **image** は RHCOS イメージの **selfLink** URL です。

(5) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

6 **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。

7 **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

- 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

6.10.15.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例6.8 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
```

```
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
```

```
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

6.10.16. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

6.10.17. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。

注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. サービスアカウントのメールアドレスをエクスポートします。

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

   ```
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
```

```
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF
```

1 **name** はワーカーマシンの名前です (例: **worker-0**)。

2 9 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3 10 **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

4 11 **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

5 12 **image** は RHCOS イメージの **selfLink** URL です。[1]

6 13 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

7 14 **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

8 15 **ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace イメージを使用するには、使用するオファーを指定します。

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145**

6.10.17.1. ワーカーマシンの Deployment Manager テンプレート

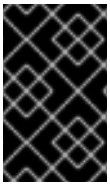
以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例6.9 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

6.10.18. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

   ```
   $ echo $PATH
   ```

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

```
$ oc <command>
```

6.10.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

   ```
   $ oc whoami
   ```

出力例

```
system:admin
```

6.10.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

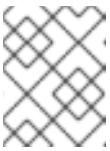
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                    CONDITION
```

```
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

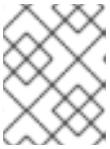
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.10.21. オプション: Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.*** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP  98
```

2. A レコードをゾーンに追加します。
   - A レコードを使用するには、以下を実行します。
     i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

     iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
```

```
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

6.10.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

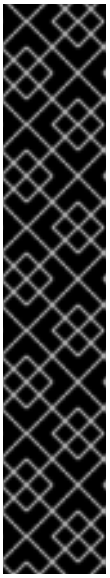
1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. クラスターの稼働状態を確認します。

   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

   出力例

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

   b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

   出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
```

```
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                          NAME
READY     STATUS      RESTARTS   AGE
kube-system                                        etcd-member-ip-10-0-3-111.us-east-
2.compute.internal              1/1       Running     0          35m
kube-system                                        etcd-member-ip-10-0-3-239.us-east-
2.compute.internal              1/1       Running     0          37m
kube-system                                        etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1       Running     0          35m
openshift-apiserver-operator                       openshift-apiserver-operator-6d6674f4f4-
h7t2t                           1/1       Running     1          37m
openshift-apiserver                                apiserver-fm48r
1/1       Running     0          30m
openshift-apiserver                                apiserver-fxkvv
1/1       Running     0          29m
openshift-apiserver                                apiserver-q85nm
1/1       Running     0          29m
...
openshift-service-ca-operator                      openshift-service-ca-operator-66ff6dc6cd-
9r257                           1/1       Running     0          37m
openshift-service-ca                               apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1       Running     0          35m
openshift-service-ca                               configmap-cabundle-injector-8498544d7-
25qn6                           1/1       Running     0          35m
openshift-service-ca                               service-serving-cert-signer-6445fc9c6-wqdqn
1/1       Running     0          35m
openshift-service-catalog-apiserver-operator       openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1       Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1       Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

6.10.23. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.10.24. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- GCP にグローバルにアクセスできる Ingress コントローラーを設定 します。

6.11. DEPLOYMENT MANAGER テンプレートを使用した GCP の共有 VPC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、独自に提供するインフラストラクチャーを使用する Google Cloud Platform (GCP) の共有 VPC (Virtual Private Cloud) にクラスターをインストールできます。この場合、共有 VPC にインストールされたクラスターは、クラスターがデプロイされる場所とは異なるプロジェクトから VPC を使用するように設定されるクラスターです。

共有 VPC により、組織は複数のプロジェクトから共通の VPC ネットワークにリソースを接続できるようになります。対象のネットワークの内部 IP を使用して、組織内の通信を安全かつ効率的に実行できます。共有 VPC の詳細は、GCP ドキュメントの Shared VPC overview を参照してください。

以下に、ユーザーによって提供されるインフラストラクチャーの共有 VPC へのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

6.11.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

6.11.2. 証明書署名要求の管理

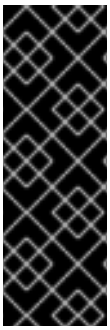
ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

6.11.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.11.4. クラスターをホストする GCP プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

6.11.4.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。

重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

6.11.4.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表6.36 必要な API サービス

API サービス	コンソールサービス名
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**

API サービス	コンソールサービス名
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

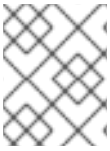
6.11.4.3. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表6.37 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
ヘルスチェック	コンピュート	グローバル	2	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0



注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

6.11.4.4. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

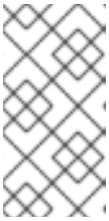
前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。

2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。



注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

6.11.4.4.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor
- サービスアカウントキー管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表6.38 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

6.11.4.5. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)

- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)

6.11.4.6. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

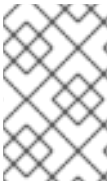
手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
   GCP ドキュメントで、サービスアカウントでの認証 について参照してください。

6.11.5. 共有 VPC ネットワークをホストする GCP プロジェクトの設定

共有 VPC (Virtual Private Cloud) を使用して Google Cloud Platform (GCP) で OpenShift Container Platform クラスターをホストする場合、これをホストするプロジェクトを設定する必要があります。



注記

共有 VPC ネットワークをホストするプロジェクトがすでにある場合は、本セクションを参照して、プロジェクトが OpenShift Container Platform クラスターのインストールに必要なすべての要件を満たすことを確認します。

手順

1. OpenShift Container Platform クラスターの共有 VPC をホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。
2. 共有 VPC をホストするプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
3. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

共有 VPC ネットワークをホストするプロジェクトのサービスアカウントには以下のロールが必要です。

- コンピュートネットワークユーザー
- コンピュートセキュリティー管理者
- Deployment Manager Editor
- DNS 管理者
- セキュリティー管理者
- ネットワーク管理者

6.11.5.1. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、クラスターをインストールする共有 VPC をホストするプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。

openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

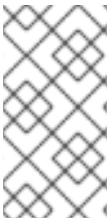
4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

6.11.5.2. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

2. リソース定義で必要な以下の変数をエクスポートします。

   a. コントロールプレーンの CIDR をエクスポートします。

   ```
   $ export MASTER_SUBNET_CIDR='10.0.0.0/17'
   ```

   b. コンピュート CIDR をエクスポートします。

   ```
   $ export WORKER_SUBNET_CIDR='10.0.128.0/17'
   ```

   c. VPC ネットワークおよびクラスターをデプロイするリージョンを以下にエクスポートします。

   ```
   $ export REGION='<region>'
   ```

3. 共有 VPC をホストするプロジェクトの ID の変数をエクスポートします。

```
$ export HOST_PROJECT=<host_project>
```

4. ホストプロジェクトに属するサービスアカウントのメールの変数をエクスポートします。

```
$ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
```

5. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '<prefix>' (1)
    region: '${REGION}' (2)
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' (3)
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' (4)
EOF
```

(1) **infra_id** は、ネットワーク名の接頭辞です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/17**)。

(4) **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.128.0/17**)。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config
01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} (1)
```

(1) **<vpc_deployment_name>** には、デプロイする VPC の名前を指定します。

7. 他のコンポーネントが必要とする VPC 変数をエクスポートします。
   a. ホストプロジェクトネットワークの名前をエクスポートします。

```
$ export HOST_PROJECT_NETWORK=<vpc_network>
```

   b. ホストプロジェクトのコントロールプレーンのサブネットの名前をエクスポートします。

```
$ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
```

   c. ホストプロジェクトのコンピュートサブネットの名前をエクスポートします。

```
$ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
```

8. 共有 VPC を設定します。GCP ドキュメントの 共有 VPC の設定 を参照してください。

6.11.5.2.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.10 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
```

```
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }]
    }
}]

return {'resources': resources}
```

6.11.6. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

6.11.6.1. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

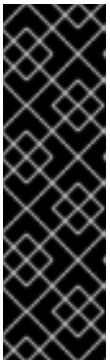
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



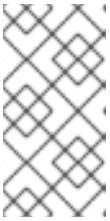
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

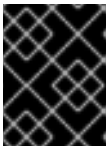
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

6.11.6.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5)
- hyperthreading: Enabled (6)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 0
metadata:
  name: test-cluster
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 7
    region: us-central1 8
pullSecret: '{"auths": ...}'
fips: false 9
sshKey: ssh-ed25519 AAAA... 10
publish: Internal 11
```

1 ホストプロジェクトでパブリック DNS を指定します。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

7 仮想マシンインスタンスが存在するメインのプロジェクトを指定します。

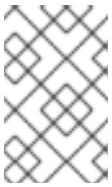
8 VPC ネットワークが置かれているリージョンを指定します。

9 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

10 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

11 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。独自にプロビジョニングするインフラストラクチャーを使用するクラスターで共有 VPC を使用するには、**publish** を **Internal** に設定する必要があります。インストールプログラムは、ホストプロジェクトのベースドメインのパブリック DNS ゾーンにアクセスできなくなります。

6.11.6.3. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

6.11.6.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.11.6.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

5. **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
status: {}
```

1 このセクションを完全に削除します。

6. VPC のクラウドプロバイダーを設定します。

   a. **<installation_directory>/manifests/cloud-provider-config.yaml** ファイルを開きます。

   b. **network-project-id** パラメーターを追加し、その値を共有 VPC ネットワークをホストするプロジェクトの ID に設定します。

   c. **network-name** パラメーターを追加し、その値を OpenShift Container Platform クラスターをホストする共有 VPC ネットワークの名前に設定します。

   d. **subnetwork-name** パラメーターの値を、コンピュートマシンをホストする共有 VPC サブネットの値に置き換えます。

<installation_directory>/manifests/cloud-provider-config.yaml の内容は以下の例のようになります。

```
config: |+
  [global]
  project-id      = example-project
  regional        = true
  multizone       = true
  node-tags       = opensh-ptzzx-master
  node-tags       = opensh-ptzzx-worker
  node-instance-prefix = opensh-ptzzx
  external-instance-groups-prefix = opensh-ptzzx
  network-project-id = example-shared-vpc
  network-name    = example-network
  subnetwork-name = example-worker-subnet
```

7. プライベートネットワーク上にないクラスターをデプロイする場合は、**<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、**scope** パラメーターの値を **External** に置き換えます。ファイルの内容は以下の例のようになります。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
    type: LoadBalancerService
status:
  availableReplicas: 0
  domain: ''
  selector: ''
```

8. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

6.11.7. 一般的な変数のエクスポート

6.11.7.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な GCP リソースを見つけるためにも使用されます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

6.11.7.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- クラスターの Ignition 設定ファイルを生成します。

- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>' 1
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>' 2
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 3
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
```

1 2 ホストプロジェクトの値を指定します。

3 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

6.11.8. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

6.11.8.1. DHCP を使用したクラスターノードのホスト名の設定

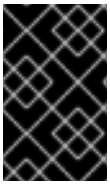
Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

6.11.8.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表6.39 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表6.40 すべてのマシンからコントロールプレーンへの通信に使用されるポート

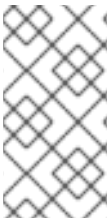
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表6.41 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

6.11.9. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。
3. デプロイメントテンプレートが使用する変数をエクスポートします。
   a. クラスターネットワークの場所をエクスポートします。

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe
${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`)
```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`)
```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(4) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(5) **control_subnet** は、コントロールサブセットの URL です。

(6) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

6.11.9.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

-

例6.11 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

6.11.9.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例6.12 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
```

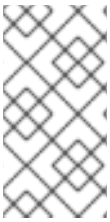
```
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
```

```
                'port': 22623
            }, {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': zone
    }
})

return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

6.11.10. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
```

```
  cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
  cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。

   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

   b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

6.11.10.1. プライベート DNS の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例6.13 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):
```

```
resources = [{
    'name': context.properties['infra_id'] + '-private-zone',
    'type': 'dns.v1.managedZone',
    'properties': {
        'description': '',
        'dnsName': context.properties['cluster_domain'] + '.',
        'visibility': 'private',
        'privateVisibilityConfig': {
            'networks': [{
                'networkUrl': context.properties['cluster_network']
            }]
        }
    }
}]

return {'resources': resources}
```

6.11.11. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
```

```
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

6.11.11.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例6.14 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
```

```
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
```

```
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

6.11.12. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コントロールプレーンおよびコンピュートサブネットをホストするサブネットのサービスアカウントに、インストールプログラムが必要とするパーミッションを割り当てます。
   a. 共有 VPC をホストするプロジェクトの **networkViewer** ロールをマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
projects add-iam-policy-binding ${HOST_PROJECT} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
"roles/compute.networkViewer"
```

b. **networkUser** ロールをコントロールプレーンサブネットのマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

c. **networkUser** ロールをコントロールプレーンサブネットのワーカーサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

d. **networkUser** ロールをコンピュートサブネットのマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

e. **networkUser** ロールをコンピュートサブネットのワーカーサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

6.11.12.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例6.15 03_iam.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

6.11.13. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE="gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz"
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

6.11.14. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。

- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

3. **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。
4. **cluster_network** はクラスターネットワークの **selfLink** URL です。
5. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
6. **image** は RHCOS イメージの **selfLink** URL です。
7. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
8. **root_volume_size** はブートストラップマシンのブートディスクサイズです。
9. **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-
instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal-backend-service
--region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --instance-
group-zone=${ZONE_0}
```

6.11.14.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

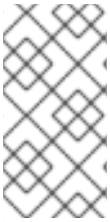
例6.16 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-bootstrap-public-ip',
    'type': 'compute.v1.address',
    'properties': {
      'region': context.properties['region']
    }
  }, {
    'name': context.properties['infra_id'] + '-bootstrap',
    'type': 'compute.v1.instance',
    'properties': {
      'disks': [{
        'autoDelete': True,
        'boot': True,
```

```
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

6.11.15. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)
```

```
    ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。
3. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
4. **image** は RHCOS イメージの **selfLink** URL です。
5. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
6. **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。
7. **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

- 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

6.11.15.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例6.17 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
```

```
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][1]
    }
}, {
    'name': context.properties['infra_id'] + '-master-2',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][2]
    }
}]

return {'resources': resources}
```

6.11.16. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

6.11.17. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラススケーリングメカニズム

やマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。

注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

   b. サービスアカウントのメールアドレスをエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

   c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name** はワーカーマシンの名前です (例: **worker-0**)。

(2) (9) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) (10) **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

(4) (11) **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

(5) (12) **image** は RHCOS イメージの **selfLink** URL です。[1]

(6) (13) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(7) (14) **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

(8) (15) **ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace イメージを使用するには、使用するオファーを指定します。

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145**

6.11.17.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

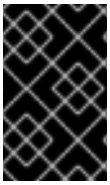
例6.18 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
```

```
    }]

    return {'resources': resources}
```

6.11.18. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

6.11.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.11.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

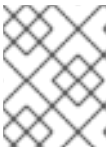
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
master-2 Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

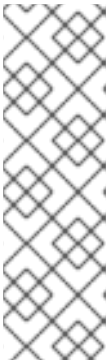
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.11.21. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定が削除されます。Ingress ロードバランサーを参照する DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、**EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP   98
```

2. A レコードをゾーンに追加します。
   - A レコードを使用するには、以下を実行します。
     i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
```

```
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME} -
-project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

6.11.22. Ingress ファイアウォールルールの追加

クラスターには複数のファイアウォールルールが必要です。共有 VPC を使用しない場合、これらのルールは GCP クラウドプロバイダーを介して Ingress コントローラーによって作成されます。共有 VPC を使用する場合は、現在すべてのサービスのクラスター全体のファイアウォールルールを作成するか、またはクラスターがアクセスを要求する際にイベントに基づいて各ルールを作成できます。クラスターがアクセスを要求する際に各ルールを作成すると、どのファイアウォールルールが必要であるかを正確に把握できます。クラスター全体のファイアウォールルールを作成する場合、同じルールセットを複数のクラスターに適用できます。

イベントに基づいて各ルールを作成する選択をする場合、クラスターをプロビジョニングした後、またはクラスターの有効期間中にコンソールがルールが見つからないことを通知する場合にファイアウォールルールを作成する必要があります。以下のイベントと同様のイベントが表示され、必要なファイアウォールルールを追加する必要があります。

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

出力例

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-a26e631036a3f46cba28f8df67266d55 --network example-network --description "{\"kubernetes.io/service-name\":\"openshift-ingress/router-default\", \"kubernetes.io/service-ip\":\"35.237.236.234\"}\" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exampl-fqzq7-master,exampl-fqzq7-worker --project example-project`
```

これらのルールベースのイベントの作成時に問題が発生した場合には、クラスターの実行中にクラスター全体のファイアウォールルールを設定できます。

6.11.22.1. GCP での共有 VPC のクラスター全体のファイアウォールルールの作成

クラスター全体のファイアウォールルールを作成して、OpenShift Container Platform クラスターに必要なアクセスを許可します。

警告

クラスターイベントに基づいてファイアウォールルールを作成しない場合、クラスター全体のファイアウォールルールを作成する必要があります。

前提条件

- Deployment Manager テンプレートがクラスターをデプロイするために必要な変数をエクスポートしています。
- GCP にクラスターに必要なネットワークおよび負荷分散コンポーネントを作成しています。

手順

1. 単一のファイアウォールルールを追加して、Google Cloud Engine のヘルスチェックがすべてのサービスにアクセスできるようにします。このルールにより、Ingress ロードバランサーはインスタンスのヘルスステータスを判別できます。

```
$ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --network="${CLUSTER_NETWORK}" --source-ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress-hc --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

2. 単一のファイアウォールルールを追加して、すべてのクラスターサービスへのアクセスを許可します。
   - 外部クラスターの場合:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --network="${CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

- プライベートクラスターの場合:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="${CLUSTER_NETWORK}" --source-ranges=${NETWORK_CIDR} --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

このルールでは、TCP ポート **80** および **443** でのトラフィックのみを許可するため、サービスが使用するすべてのポートを追加するようにしてください。

6.11.23. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

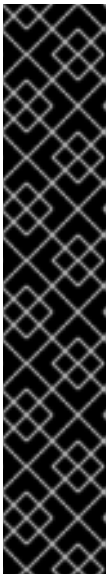
1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. クラスターの稼働状態を確認します。

   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

   出力例

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

   b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

   出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
```

```
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                          NAME
READY     STATUS      RESTARTS   AGE
kube-system                                        etcd-member-ip-10-0-3-111.us-east-
2.compute.internal              1/1       Running     0          35m
kube-system                                        etcd-member-ip-10-0-3-239.us-east-
2.compute.internal              1/1       Running     0          37m
kube-system                                        etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1       Running     0          35m
openshift-apiserver-operator                       openshift-apiserver-operator-6d6674f4f4-
h7t2t                           1/1       Running     1          37m
openshift-apiserver                                apiserver-fm48r
1/1       Running     0          30m
openshift-apiserver                                apiserver-fxkvv
1/1       Running     0          29m
openshift-apiserver                                apiserver-q85nm
1/1       Running     0          29m
...
openshift-service-ca-operator                      openshift-service-ca-operator-66ff6dc6cd-
9r257                           1/1       Running     0          37m
openshift-service-ca                               apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1       Running     0          35m
openshift-service-ca                               configmap-cabundle-injector-8498544d7-
25qn6                           1/1       Running     0          35m
openshift-service-ca                               service-serving-cert-signer-6445fc9c6-wqdqn
1/1       Running     0          35m
openshift-service-catalog-apiserver-operator       openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1       Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1       Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

6.11.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.11.25. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.12. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したネットワークが制限された環境での GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、独自に提供するインフラストラクチャーおよびインストールリリースコンテンツの内部ミラーを使用して、クラスターを Google Cloud Platform (GCP) にインストールできます

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットへのアクセスが必要になります。

以下に、ユーザーによって提供されるインフラストラクチャーのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

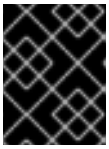
重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

6.12.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。他のサイトへのアクセスを付与する必要がある場合もありますが、***.googleapis.com** および **accounts.google.com** へのアクセスを付与する必要があります。
- お使いの環境でクラウドアイデンティティーおよびアクセス管理 (IAM) API にアクセスできない場合や、管理者レベルの認証情報シークレットを **kube-system** namespace に保存することを望まない場合は、IAM 認証情報を手動で作成および維持 することができます。

6.12.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

6.12.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。

- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

6.12.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

6.12.4. GCP プロジェクトの設定

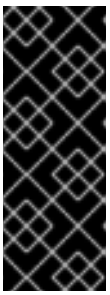
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

6.12.4.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

6.12.4.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表6.42 必要な API サービス

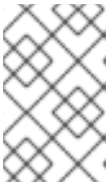
API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

6.12.4.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。



注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

6.12.4.4. GCP アカウントの制限

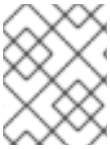
OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表6.43 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
ヘルスチェック	コンピュート	グローバル	2	0
イメージ	コンピュート	グローバル	1	0

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0



注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

6.12.4.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

6.12.4.5.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor
- サービスアカウントキー管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表6.44 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

6.12.4.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)

- **australia-southeast1** (Sydney, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)

6.12.4.7. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。

GCP ドキュメントで、サービスアカウントでの認証 について参照してください。

6.12.5. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。また、インストールの準備フェーズ時にまず別の **var** パーティションを設定するオプションもあります。

6.12.5.1. オプション: 別個の /var パーティションの作成

OpenShift Container Platform のディスクパーティション設定はインストーラー側で行う必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

重要

この手順で個別の **/var** パーティションを作成する手順を実行する場合、このセクションで後に説明されるように、Kubernetes マニフェストおよび Ignition 設定ファイルを再び作成する必要はありません。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

出力例

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. オプション: インストールプログラムで **clusterconfig/openshift** ディレクトリーにマニフェストが作成されたことを確認します。

```
$ ls $HOME/clusterconfig/openshift/
```

出力例

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 MiB (メビバイト) の最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

5. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールするためにインストール手順への入力として使用できます。

6.12.5.2. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

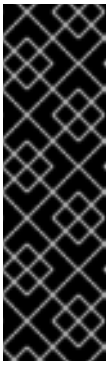
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既存のサブネットを指定します。

d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

6.12.5.3. カスタムマシンタイプの使用

カスタムマシンタイプを使用した OpenShift Container Platform クラスターのインストールがサポートされます。

カスタムマシンタイプを使用する場合は、以下を考慮してください。

- 事前定義されたインスタンスタイプと同様に、カスタムマシンタイプはコントロールプレーンおよびコンピュートマシンの最小リソース要件を満たしている必要があります。
- カスタムマシンタイプの名前は、以下の構文に準拠する必要があります。
  custom-<number_of_cpus>-<amount_of_memory_in_mb>

  たとえば、**custom-6-20480** です。

6.12.5.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。

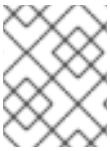
4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



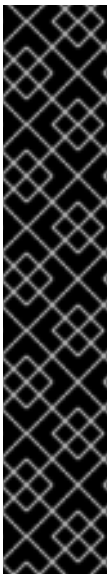
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.12.5.5. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- オプション: Ingress DNS レコードの追加

6.12.6. 一般的な変数のエクスポート

6.12.6.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。インフラストラクチャー名は、OpenShift Container Platform のインストール時に適切な GCP リソースを見つけるためにも使用されます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

6.12.6.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- クラスターの Ignition 設定ファイルを生成します。

- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

6.12.7. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py
```

```
resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/17**)。
4. **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.128.0/17**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

6.12.7.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例6.19 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
```

```
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

6.12.8. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

6.12.8.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

6.12.8.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表6.45 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表6.46 すべてのマシンからコントロールプレーンへの通信に使用されるポート

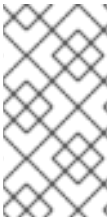
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表6.47 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

6.12.9. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。
3. デプロイメントテンプレートが使用する変数をエクスポートします。
   a. クラスターネットワークの場所をエクスポートします。

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
   ```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(4) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(5) **control_subnet** は、コントロールサブセットの URL です。

(6) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

6.12.9.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例6.20 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

6.12.9.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例6.21 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
```

```
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
```

```
                'port': 22623
            }, {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': zone
    }
})

return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

6.12.10. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
```

```
  cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
  cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。

   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

   b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

6.12.10.1. プライベート DNS の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例6.22 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-private-zone',
    'type': 'dns.v1.managedZone',
    'properties': {
      'description': '',
      'dnsName': context.properties['cluster_domain'] + '.',
```

```
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

6.12.11. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

1. **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。
2. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
3. **cluster_network** はクラスターネットワークの **selfLink** URL です。
4. **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

6.12.11.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例6.23 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
```

```
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
```

```
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

6.12.12. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コンピュートマシンをホストするサブネットの変数をエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
```

```
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

6.12.12.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例6.24 03_iam.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

6.12.13. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE="gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz"
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

6.12.14. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。
4. **cluster_network** はクラスターネットワークの **selfLink** URL です。
5. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
6. **image** は RHCOS イメージの **selfLink** URL です。
7. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
8. **root_volume_size** はブートストラップマシンのブートディスクサイズです。
9. **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager の制限によりテンプレートではロードバランサーのメンバーシップを管理しないため、ブートストラップマシンは手動で追加する必要があります。
   a. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances \
    ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
bootstrap
```

   b. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

```
$ gcloud compute backend-services add-backend \
    ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
```

6.12.14.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例6.25 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

6.12.15. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
```

```
  image: '${CLUSTER_IMAGE}' (4)
  machine_type: 'n1-standard-4' (5)
  root_volume_size: '128'
  service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)

  ignition: '${MASTER_IGNITION}' (7)
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。
3. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
4. **image** は RHCOS イメージの **selfLink** URL です。
5. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
6. **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。
7. **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

- 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

6.12.15.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例6.26 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
```

```
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
```

```
    }]

    return {'resources': resources}
```

6.12.16. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

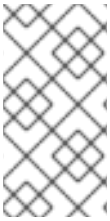
2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

6.12.17. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。



注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

   b. サービスアカウントのメールアドレスをエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

   c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name** はワーカーマシンの名前です (例: **worker-0**)。

(2) (9) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) (10) **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

(4) (11) **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

(5) (12) **image** は RHCOS イメージの **selfLink** URL です。[1]

(6) (13) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(7) (14) **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

(8) (15) **ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1. GCP Marketplace イメージを使用するには、使用するオファーを指定します。

   - OpenShift Container Platform:
     https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-48-x86-64-202210040145
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
   - OpenShift Kubernetes Engine:
     https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145

6.12.17.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例6.27 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
```

```
            ]
        },
        'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

6.12.18. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.12.19. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

6.12.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

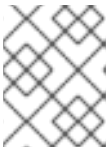
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

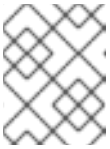
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.12.21. オプション: Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP  98
```

2. A レコードをゾーンに追加します。
   - A レコードを使用するには、以下を実行します。
     i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
```

```
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

6.12.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

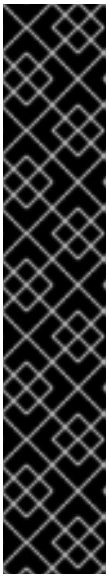
1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

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

2. クラスターの稼働状態を確認します。

a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

出力例

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

出力例

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                       4.5.4     True        False         False      7m56s
cloud-credential                     4.5.4     True        False         False      31m
cluster-autoscaler                   4.5.4     True        False         False      16m
console                              4.5.4     True        False         False      10m
csi-snapshot-controller              4.5.4     True        False         False      16m
dns                                  4.5.4     True        False         False      22m
etcd                                 4.5.4     False       False         False      25s
image-registry                       4.5.4     True        False         False      16m
ingress                              4.5.4     True        False         False      16m
insights                             4.5.4     True        False         False      17m
kube-apiserver                       4.5.4     True        False         False      19m
kube-controller-manager              4.5.4     True        False         False      20m
kube-scheduler                       4.5.4     True        False         False      20m
```

```
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                       NAME
READY   STATUS    RESTARTS   AGE
kube-system                                     etcd-member-ip-10-0-3-111.us-east-
2.compute.internal              1/1     Running   0          35m
kube-system                                     etcd-member-ip-10-0-3-239.us-east-
2.compute.internal              1/1     Running   0          37m
kube-system                                     etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1     Running   0          35m
openshift-apiserver-operator                    openshift-apiserver-operator-6d6674f4f4-
h7t2t                           1/1     Running   1          37m
openshift-apiserver                             apiserver-fm48r
1/1     Running   0          30m
openshift-apiserver                             apiserver-fxkvv
1/1     Running   0          29m
openshift-apiserver                             apiserver-q85nm
1/1     Running   0          29m
...
openshift-service-ca-operator                   openshift-service-ca-operator-66ff6dc6cd-
9r257                           1/1     Running   0          37m
openshift-service-ca                            apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running   0          35m
openshift-service-ca                            configmap-cabundle-injector-8498544d7-
25qn6                           1/1     Running   0          35m
openshift-service-ca                            service-serving-cert-signer-6445fc9c6-wqdqn
1/1     Running   0          35m
openshift-service-catalog-apiserver-operator    openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1     Running   0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1     Running   0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

6.12.23. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

6.12.24. 次のステップ

- クラスターをカスタマイズします。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.13. GCP でのクラスターのアンインストール

Google Cloud Platform (GCP) にデプロイしたクラスターを削除できます。

6.13.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。たとえば、一部の Google Cloud リソースには共有 VPC ホストプロジェクトで IAM パーミッション が必要になるか、または 削除する必要のあるヘルスチェック が使用されていない可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。

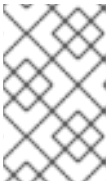
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第7章 ベアメタルへのインストール

7.1. ベアメタルクラスターのインストールの準備

7.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターンストール方法の選択およびそのユーザー向けの準備 を確認している。

7.1.2. OpenShift 仮想化のためのベアメタルクラスターの計画

OpenShift 仮想化を使用する場合は、ベアメタルクラスターをインストールする前に、いくつかの要件を認識することが重要です。

- ライブマイグレーション機能を使用する場合は、**クラスターのインストール時に** 複数のワーカーノードが必要です。これは、ライブマイグレーションではクラスターレベルの高可用性 (HA) フラグを true に設定する必要があるためです。HA フラグは、クラスターのインストール時に設定され、後で変更することはできません。クラスターのインストール時に定義されたワーカーノードが 2 つ未満の場合、クラスターの存続期間中、HA フラグは false に設定されます。

  注記

  単一ノードのクラスターに OpenShift Virtualization をインストールできますが、単一ノードの OpenShift は高可用性をサポートしていません。

- ライブマイグレーションには共有ストレージが必要です。OpenShift Virtualization のストレージは、ReadWriteMany (RWX) アクセスモードをサポートし、使用する必要があります。
- Single Root I/O Virtualization (SR-IOV) を使用する予定の場合は、ネットワークインターフェイスコントローラー (NIC) が OpenShift Container Platform でサポートされていることを確認してください。

関連情報

- OpenShift Virtualization のクラスターの準備
- Single Root I/O Virtualization (SR-IOV) ハードウェアネットワークについて
- 仮想マシンの SR-IOV ネットワークデバイスの設定

7.1.3. ベアメタルに OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

7.1.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法を使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされるベアメタルインフラストラクチャーに、クラスターをインストールできます。

- **インストーラーでプロビジョニングされるクラスターのベアメタルへのインストール**: インストーラーのプロビジョニングを使用して、OpenShift Container Platform をベアメタルにインストールできます。

7.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングするベアメタルインフラストラクチャーに、クラスターをインストールできます。

- **ユーザーによってプロビジョニングされるクラスターのベアメタルへのインストール**: OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールできます。ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

- **ネットワークのカスタマイズを使用したユーザーによってプロビジョニングされるベアメタルクラスターのインストール**: ネットワークのカスタマイズを使用して、ユーザーによってプロビジョニングされるインフラストラクチャーにベアメタルクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。ネットワークのカスタマイズのほとんどは、インストールステージで適用する必要があります。

- **ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール**: ミラーレジストリーを使用して、ユーザーによってプロビジョニングされるベアメタルクラスターを制限されたネットワークまたは非接続ネットワークにインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

7.2. ユーザーによってプロビジョニングされるクラスターのベアメタルへのインストール

OpenShift Container Platform version 4.8 では、独自にプロビジョニングするベアメタルのインフラストラクチャーにクラスターをインストールできます。

重要

以下の手順に従って仮想化環境またはクラウド環境にクラスターをデプロイすることができますが、ベアメタルプラットフォーム以外の場合は追加の考慮事項に注意してください。このような環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

7.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

注記

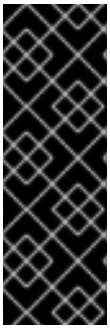
プロキシーを設定する場合は、このサイト一覧も確認してください。

7.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

関連情報

- ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーへのネットワークが制限された環境でのインストールの実行についての詳細は、ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール を参照してください。

7.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

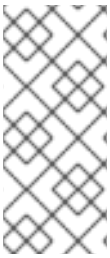
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

7.2.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表7.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



注記

例外として、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターで実行できます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。1 つのコンピュートマシンの実行はサポートされていません。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.2.3.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.2 最小リソース要件

マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	IOPS [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS または RHEL 7.9 [3]	2	8 GB	100 GB	300

1. CPU 1 つ分は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = CPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL 7 コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

7.2.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- ベアメタル環境での 3 ノードクラスターのデプロイに関する詳細は、3 ノードクラスターの設定 を参照してください。
- インストール後のクラスター証明書署名要求の承認についての詳細は、マシンの証明書署名要求の承認 を参照してください。

7.2.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起

動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

7.2.3.4.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

7.2.3.4.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表7.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表7.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイム

サーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

7.2.3.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

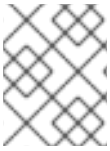
各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表7.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7.2.3.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例7.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例7.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

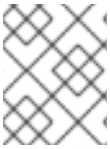
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。



注記

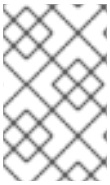
PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

7.2.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。

- ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。

   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

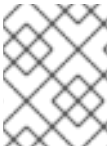
ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

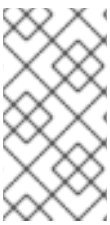


注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

7.2.3.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例7.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
```

```
bind *:80
mode tcp
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1. この例では、クラスター名は **ocp4** です。
2. ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。
3. 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。
4. ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。
6. ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。
7. ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

7.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効に

し、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

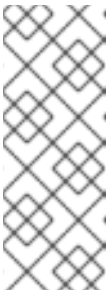
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。



注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. クラスターに必要な DNS インフラストラクチャーを設定します。
   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コント

ロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

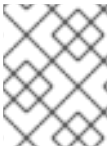
b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件
- RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始
- DHCP を使用したクラスターノードのホスト名の設定
- 高度な RHCOS インストール設定
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

7.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

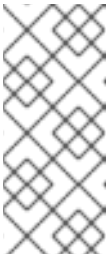
   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

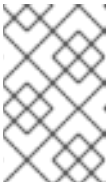
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

関連情報

- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

7.2.6. クラスターノードの SSH アクセス用のキーペアの生成

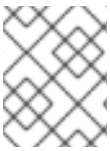
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

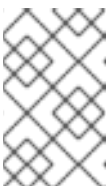
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

関連情報

- ノードの正常性の確認

7.2.7. インストールプログラムの取得

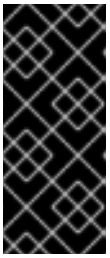
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

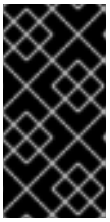
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.2.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

7.2.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

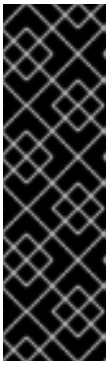
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



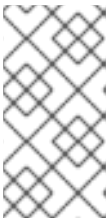
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

7.2.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

7.2.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表7.9 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.2.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表7.10 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

7.2.9.1.3. オプションの設定パラメーター

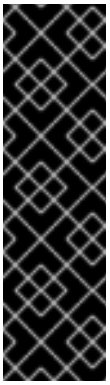
オプションのインストール設定パラメーターは、以下の表で説明されています。

表7.11 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

7.2.9.2. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
```

```
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

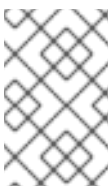
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



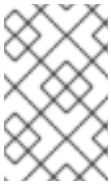
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- API およびアプリケーションの Ingress 負荷分散要件の詳細は、ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件 を参照してください。

7.2.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

注記

ベアメタルインストールでは、**install-config.yaml** ファイルの **networking.machineNetwork[].cidr** フィールドで指定される範囲にあるノード IP アドレスを割り当てない場合、それらを **proxy.noProxy** フィールドに含める必要があります。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

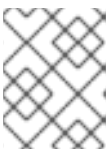
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

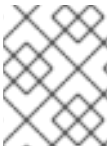


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.2.9.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

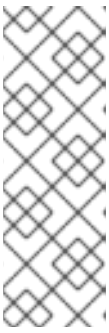
前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの

mastersSchedulable パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

7.2.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- kubelet 証明書のリカバリーに関する詳細は、コントロールプレーン証明書の期限切れの状態からのリカバリー を参照してください。

7.2.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。

注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。

注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

7.2.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

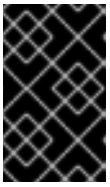
手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

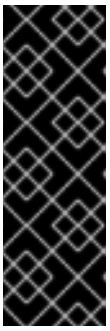
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-
live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

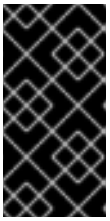
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. 継続してクラスターの他のマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

7.2.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

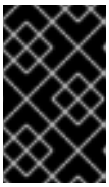
PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```

重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーと **rootfs** ファイルを HTTP サーバーにアップロードします。
   - iPXE の場合、**kernel**、**initramfs**、および **rootfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。
   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

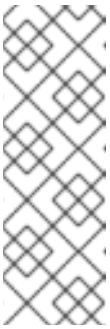
注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

- iPXE の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

1. HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。
2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。
3. HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. PXE UEFI を使用する場合は、以下の操作を実行します。
   a. システムの起動に必要な **shimx64.efi** and **grubx64.efi** EFI バイナリーと **grub.cfg** ファイルを指定します。
      - ホストに RHCOS ISO をマウントしてから、**images/efiboot.img** ファイルをマウントし、必要な EFI バイナリーを展開します。

```
$ mkdir -p /mnt/iso
```

```
$ mkdir -p /mnt/efiboot
```

```
$ mount -o loop rhcos-installer.x86_64.iso /mnt/iso
```

```
$ mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
```

- **efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
$ cp /mnt/efiboot/EFI/redhat/shimx64.efi .
```

```
$ cp /mnt/efiboot/EFI/redhat/grubx64.efi .
```

```
$ umount /mnt/efiboot
```

```
$ umount /mnt/iso
```

- RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

b. **grub.cfg** ファイルを編集し、以下のような引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linuxefi rhcos-<version>-live-kernel-<architecture> coreos.inst.install_dev=/dev/sda
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
 initrdefi rhcos-<version>-live-initramfs.<architecture>.img
}
```

詳細は以下のようになります。

rhcos-<version>-live-kernel-<architecture>

TFTP サーバーにアップロードした **kernel** ファイルを指定します。

http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img

HTTP サーバーにアップロードしたライブ rootfs イメージの場所を指定します。

http://<HTTP_server>/bootstrap.ign

HTTP サーバーにアップロードしたブートストラップ Ignition 設定ファイルの場所を指定します。

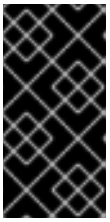
rhcos-<version>-live-initramfs.<architecture>.img

TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

UEFI ブート用に PXE サーバーを設定する方法は、Red Hat ナレッジベースの記事 How to configure/setup a PXE server for Red Hat Enterprise Linux? を参照してください。

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

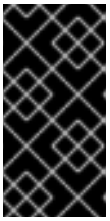


重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

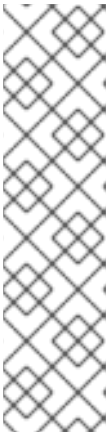
10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

7.2.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ISO への Ignition 設定の埋め込み

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

7.2.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

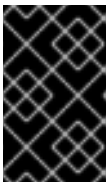
- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

手順

1. ISO インストーラーを起動します。
2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。
3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

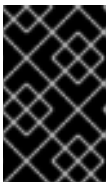
- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

7.2.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。特定のアーキテクチャーの各 RHCOS ノードは、デフォルトのパーティション設定が上書きされない限り、同じパーティションレイアウトを使用します。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

以下は、OpenShift Container Platform クラスターノードへの RHCOS のインストール時に、デフォルトのパーティション設定の上書きが必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションにマウントする場合にのみ正式にサポートされます。



重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

警告

カスタムパーティションを使用すると、これらのパーティションが OpenShift Container Platform によって監視されないか、またはアラートが通知される可能性があります。デフォルトのパーティションを上書きする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

7.2.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

7.2.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、 **coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。

注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

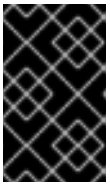
この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

7.2.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。



重要

これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは手動で作成され、Red Hat でサポートされていないため、可能な場合は使用しないようにする必要があります。この方法では、Ignition 設定はライブインストールメディアに渡され、起動時にすぐに実行され、RHCOS システムのディスクへのインストール前後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

7.2.11.3.3.1. RHCOS ISO へのライブインストール Ignition 設定の埋め込み

ライブインストール Ignition 設定を RHCOS ISO イメージに直接埋め込むことができます。ISO イメージが起動すると、埋め込み設定が自動的に適用されます。

手順

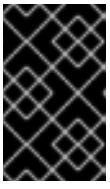
1. 以下のイメージミラーページから **coreos-installer** バイナリーをダウンロードします:
   https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/
2. RHCOS ISO イメージおよび Ignition 設定ファイルを取得し、それらを **/mnt** などのアクセス可能なディレクトリーにコピーします。

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 以下のコマンドを実行して Ignition 設定を ISO に埋め込みます。

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

   その ISO を使用して、指定されたライブインストール Ignition 設定を使用して RHCOS をインストールできるようになります。



重要

coreos-installer iso ignition embed を使用して、**openshift-installer** によって生成されるファイル (例: **bootstrap.ign**、**master.ign** および **worker.ign**) を埋め込むことはサポートされておらず、かつ推奨されていません。

4. 埋め込み Ignition 設定の内容を表示し、これをファイルに転送するには、以下を実行します。

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

出力例

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 設定を削除し、再利用できるように ISO をその初期状態に戻すには、以下を実行します。

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

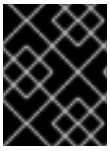
別の Ignition 設定を ISO に埋め込むか、または ISO を初期状態で使用することができるようになりました。

7.2.11.3.4. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

7.2.11.3.4.1. ISO インストールのネットワークおよびボンディングのオプション

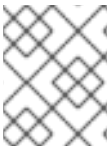
ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定

IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

7.2.11.3.4.2. ISO インストールの coreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表7.12 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**
$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。
-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--append-karg <arg>...	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>...	インストール済みシステムからデフォルトのカーネル引数を削除します。

-n、--copy-network	インストール環境からネットワーク設定をコピーします。  **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。
--save-partindex <id>…	この数または範囲でパーティションを保存します。
--insecure	署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。デフォルトは **x86_64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。
coreos-install install サブコマンド引数	
引数	**説明**
<device>	宛先デバイス。
coreos-installer ISO Ignition サブコマンド	
サブコマンド	**説明**

$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer iso ignition show <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を表示します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。
coreos-installer ISO Ignition サブコマンドオプション	
オプション	**説明**
-f、--force	既存の Ignition 設定を上書きします。
-i、--ignition-file <path>	使用される Ignition 設定。デフォルトは **stdin** です。
-o、--output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。
coreos-installer PXE Ignition サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。
coreos-installer PXE Ignition サブコマンドオプション	
オプション	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
-i、--ignition-file <path>	使用される Ignition 設定。デフォルトは **stdin** です。
-o, --output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。

7.2.11.3.4.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表7.13 coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。

引数	説明
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

7.2.11.4. RHCOS のカーネル引数でのマルチパスの有効化

RHCOS はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。

OpenShift Container Platform 4.8 以降でプロビジョニングされたノードのマルチパスを有効にできます。インストール後のサポートはマシン設定を使用してマルチパスをアクティベートすることで利用できますが、インストール中にマルチパスを有効にすることをお勧めします。

非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。

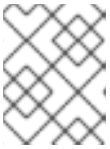
重要

IBM Z および LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z および LinuxONE への z/VM を使用したクラスターのインストール**の RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始を参照してください。

以下の手順では、インストール時にマルチパスを有効にし、**coreos-installer install** コマンドにカーネル引数を追加して、インストール済みシステム自体が初回起動からマルチパスを使用できるようにします。

前提条件

- バージョン 4.8 以降を使用する OpenShift Container Platform クラスターが実行中である。



注記

OpenShift Container Platform は、4.6 以前からアップグレードされたノードでの day-2 アクティビティーとしてのマルチパスの有効化をサポートしません。

- 管理者権限を持つユーザーとしてクラスターにログインしている。

手順

1. マルチパスを有効にして **multipathd** デーモンを起動するには、以下のコマンドを実行します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

   - 必要に応じて、PXE または ISO を起動する場合は、カーネルコマンドラインから **rd.multipath=default** を追加することで、マルチパスを有効にできます。

2. **coreos-installer** プログラムを呼び出してカーネル引数を追加します。

   - マシンに接続されているマルチパスデバイスが 1 つしかない場合は、このデバイスは **/dev/mapper/mpatha** のパスで利用できます。以下に例を示します。

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

   1 1 つのマルチパスデバイスのパスを指定します。

   - 複数のマルチパスデバイスがマシンに接続している場合には、より明示的に **/dev/mapper/mpatha** を使用する代わりに、**/dev/disk/by-id** で利用可能な World Wide Name (WWN) シンボリックリンクを使用することが推奨されます。以下に例を示します。

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

   1 マルチパス化されたデバイスの WWN ID を指定します。例: **0xx194e957fcedb4841**

特別な **coreos.inst.*** 引数を使用してライブインストーラーを指定する場合に、このシンボリックリンクを **coreos.inst.install_dev** カーネル引数として使用することもできます。詳細は、Installing RHCOS and starting the OpenShift Container Platform bootstrap process を参照してください。

3. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) を一覧表示してカーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

出力例

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

関連情報

- 特別な **coreos.inst.*** 引数を使用してライブインストーラーを指示する方法は、RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始 を参照してください。

7.2.11.5. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。

注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

7.2.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をイ

ンストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

関連情報

- インストールログの監視と、インストールの問題が発生した場合の診断データの取得の詳細については、インストールの進捗の監視 を参照してください。

7.2.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.2.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
```

```
master-2  Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

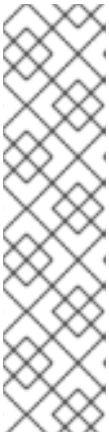
```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

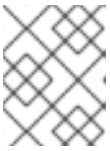
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.2.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
```

```
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

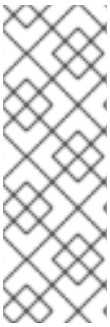
関連情報

- OpenShift Container Platform インストールの失敗時のデータの収集についての詳細は、失敗したインストールのログの収集 を参照してください。
- クラスター全体で Operator Pod の正常性を確認し、診断用に Operator ログを収集する手順の詳細は、Operator 関連の問題のトラブルシューティング を参照してください。

7.2.15.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

7.2.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

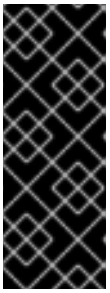
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

7.2.15.2.1. ベアメタルおよび他の手動インストールの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージがある。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

7.2.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

7.2.15.2.3. ブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

7.2.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

–

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。

注記

マルチパスを使用してインストールする場合は、後で有効にすると問題が発生する可能性があるので、後ではなく、インストール時に有効にすることを強く推奨します。

詳細は、**ベアメタルへのインストール** の RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

7.2.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

7.2.18. 次のステップ

- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。

7.3. ネットワークのカスタマイズを使用したユーザーによってプロビジョニングされるベアメタルクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、カスタマイズされたネットワーク設定オプションでプロビジョニングするベアメタルインフラストラクチャーにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

OpenShift Container Platform ネットワークをカスタマイズする場合、インストール時にほとんどのネットワーク設定パラメーターを設定する必要があります。実行中のクラスターで変更できるのは **kubeProxy** ネットワーク設定パラメーターのみです。

7.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

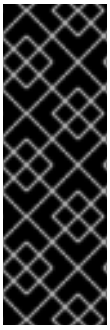
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

7.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

関連情報

- ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーへのネットワークが制限された環境でのインストールの実行についての詳細は、ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール を参照してください。

7.3.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

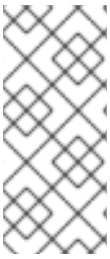
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

7.3.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

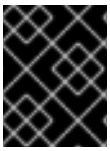
表7.14 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



注記

例外として、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターで実行できます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。1 つのコンピュートマシンの実行はサポートされていません。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.3.3.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.15 最小リソース要件

マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	IOPS [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300

マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	IOPS [2]
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS または RHEL 7.9 [3]	2	8 GB	100 GB	300

1. CPU 1 つ分は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = CPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL 7 コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

7.3.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- ベアメタル環境での 3 ノードクラスターのデプロイに関する詳細は、3 ノードクラスターの設定 を参照してください。
- インストール後のクラスター証明書署名要求の承認についての詳細は、マシンの証明書署名要求の承認 を参照してください。

7.3.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その

後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

7.3.3.4.1. DHCP を使用したクラスターノードのホスト名の設定

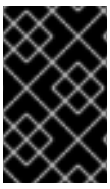
Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

7.3.3.4.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表7.16 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表7.17 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.18 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

7.3.3.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

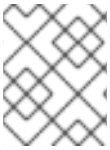
各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表7.19 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7.3.3.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

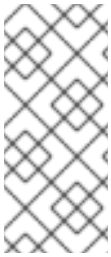
ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例7.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例7.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

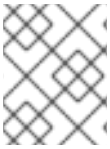
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。



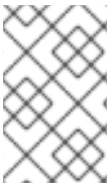
注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

7.3.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



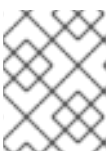
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.20 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

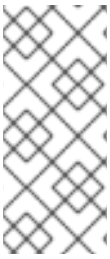
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.21 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

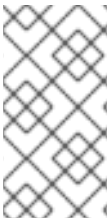
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

7.3.3.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例7.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

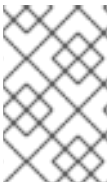
7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

7.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。

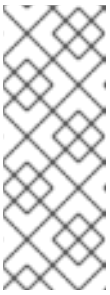
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

   注記

   DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

   注記

   DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

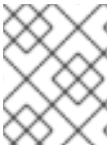
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

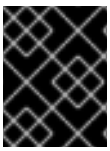
一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件
- RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始
- DHCP を使用したクラスターノードのホスト名の設定
- 高度な RHCOS インストール設定
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

7.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

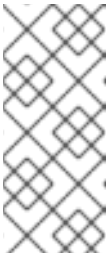
   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

   ```
   $ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
   <cluster_name>.<base_domain>
   ```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

関連情報

- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

7.3.6. クラスターノードの SSH アクセス用のキーペアの生成

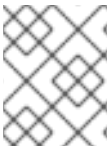
OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。



注記

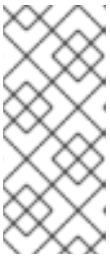
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

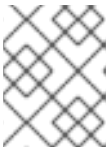
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

関連情報

- ノードの正常性の確認

7.3.7. インストールプログラムの取得

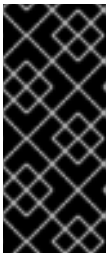
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- 500 MB のローカルディスク領域がある Linux または macOS を実行するコンピューターが必要です。

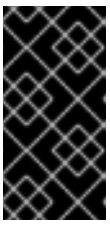
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

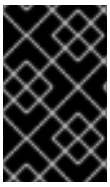
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.3.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

7.3.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

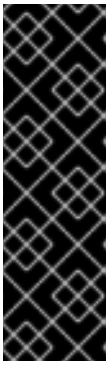
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



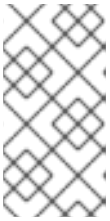
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

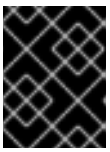
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

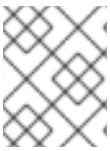


重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

7.3.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

7.3.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表7.22 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.3.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表7.23 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

7.3.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表7.24 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

7.3.9.2. ベアメタルのサンプル install-config.yaml ファイル

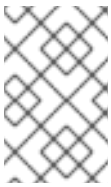
install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
  - 172.30.0.0/16
platform:
  none: {} (12)
fips: false (13)
pullSecret: '{"auths": ...}' (14)
sshKey: 'ssh-ed25519 AAAA...' (15)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行は

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



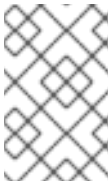
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するよう

に設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

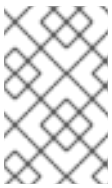
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

関連情報

- API およびアプリケーションの Ingress 負荷分散要件の詳細は、ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件 を参照してください。

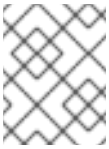
7.3.10. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

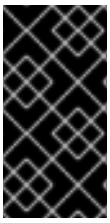
フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

7.3.11. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。



重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

7.3.12. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

7.3.12.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表7.25 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表7.26 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表7.27 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表7.28 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 暗号化を有効にするために空のオブジェクトを指定します。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表7.29 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig: {}
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表7.30 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

7.3.13. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。

重要

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

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

7.3.14. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。

注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

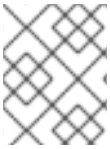
- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル

引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。



注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

7.3.14.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-
live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

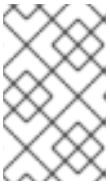
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。



注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、**sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

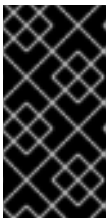
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

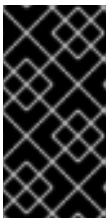


重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

7.3.14.2. PXE または iPXE ブートを使用した RHCOS のインストール

PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

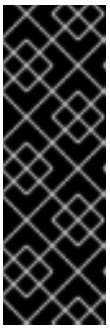
3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
```

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



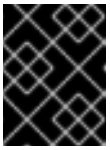
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーと **rootfs** ファイルを HTTP サーバーにアップロードします。
   - iPXE の場合、**kernel**、**initramfs**、および **rootfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

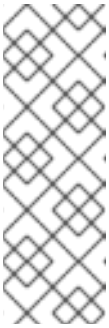
   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

     (1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

     (2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

     (3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

   - iPXE の場合:

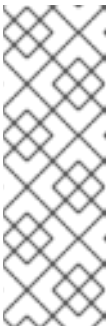
```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

     (1) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場

所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

3 HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. PXE UEFI を使用する場合は、以下の操作を実行します。

   a. システムの起動に必要な **shimx64.efi** and **grubx64.efi** EFI バイナリーと **grub.cfg** ファイルを指定します。

   - ホストに RHCOS ISO をマウントしてから、**images/efiboot.img** ファイルをマウントし、必要な EFI バイナリーを展開します。

```
$ mkdir -p /mnt/iso
```

```
$ mkdir -p /mnt/efiboot
```

```
$ mount -o loop rhcos-installer.x86_64.iso /mnt/iso
```

```
$ mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
```

   - **efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
$ cp /mnt/efiboot/EFI/redhat/shimx64.efi .
```

```
$ cp /mnt/efiboot/EFI/redhat/grubx64.efi .
```

```
$ umount /mnt/efiboot
```

```
$ umount /mnt/iso
```

   - RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

   b. **grub.cfg** ファイルを編集し、以下のような引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linuxefi rhcos-<version>-live-kernel-<architecture> coreos.inst.install_dev=/dev/sda
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
 initrdefi rhcos-<version>-live-initramfs.<architecture>.img
}
```

詳細は以下のようになります。

rhcos-<version>-live-kernel-<architecture>

TFTP サーバーにアップロードした **kernel** ファイルを指定します。

http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img

HTTP サーバーにアップロードしたライブ rootfs イメージの場所を指定します。

http://<HTTP_server>/bootstrap.ign

HTTP サーバーにアップロードしたブートストラップ Ignition 設定ファイルの場所を指定します。

rhcos-<version>-live-initramfs.<architecture>.img

TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

UEFI ブート用に PXE サーバーを設定する方法は、Red Hat ナレッジベースの記事 How to configure/setup a PXE server for Red Hat Enterprise Linux? を参照してください。

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

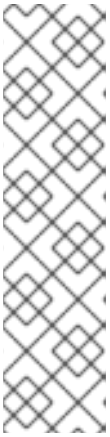
9. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

7.3.14.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ISO への Ignition 設定の埋め込み

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

7.3.14.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

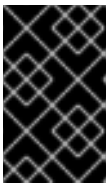
手順

1. ISO インストーラーを起動します。

2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。

3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

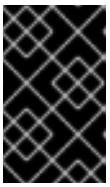
- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

7.3.14.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。特定のアーキテクチャーの各 RHCOS ノードは、デフォルトのパーティション設定が上書きされない限り、同じパーティションレイアウトを使用します。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

以下は、OpenShift Container Platform クラスターノードへの RHCOS のインストール時に、デフォルトのパーティション設定の上書きが必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションにマウントする場合にのみ正式にサポートされます。



重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

警告

カスタムパーティションを使用すると、これらのパーティションが OpenShift Container Platform によって監視されないか、またはアラートが通知される可能性があります。デフォルトのパーティションを上書きする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

7.3.14.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
```

```
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のフートストラップ、コントロールフレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

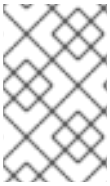
次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

7.3.14.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、**coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。



注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

7.3.14.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

  重要

  これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

  PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは手動で作成され、Red Hat でサポートされていないため、可能な場合は使用しないようにする必要があります。この方法では、Ignition 設定はライブインストールメディアに渡され、起動時にすぐに実行され、RHCOS システムのディスクへのインストール前後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

7.3.14.3.3.1. RHCOS ISO へのライブインストール Ignition 設定の埋め込み

ライブインストール Ignition 設定を RHCOS ISO イメージに直接埋め込むことができます。ISO イメージが起動すると、埋め込み設定が自動的に適用されます。

手順

1. 以下のイメージミラーページから **coreos-installer** バイナリーをダウンロードします: https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/

2. RHCOS ISO イメージおよび Ignition 設定ファイルを取得し、それらを **/mnt** などのアクセス可能なディレクトリーにコピーします。

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 以下のコマンドを実行して Ignition 設定を ISO に埋め込みます。

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

その ISO を使用して、指定されたライブインストール Ignition 設定を使用して RHCOS をインストールできるようになります。

重要

coreos-installer iso ignition embed を使用して、**openshift-installer** によって生成されるファイル (例: **bootstrap.ign**、**master.ign** および **worker.ign**) を埋め込むことはサポートされておらず、かつ推奨されていません。

4. 埋め込み Ignition 設定の内容を表示し、これをファイルに転送するには、以下を実行します。

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

出力例

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 設定を削除し、再利用できるように ISO をその初期状態に戻すには、以下を実行します。

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

別の Ignition 設定を ISO に埋め込むか、または ISO を初期状態で使用することができるようになりました。

7.3.14.3.4. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

7.3.14.3.4.1. ISO インストールのネットワークおよびボンディングのオプション

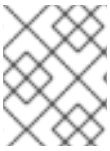
ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを

設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定

静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定され

静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、network_interfacesは物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

7.3.14.3.4.2. ISO インストールの coreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表7.31 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**
$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。

-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--append-karg <arg>…	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>…	インストール済みシステムからデフォルトのカーネル引数を削除します。
-n、--copy-network	インストール環境からネットワーク設定をコピーします。  **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。
--save-partindex <id>…	この数または範囲でパーティションを保存します。
--insecure	署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。デフォルトは **x86_64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。

coreos-install install サブコマンド引数	
引数	**説明**
<device>	宛先デバイス。
coreos-installer ISO Ignition サブコマンド	
サブコマンド	**説明**
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer iso ignition show <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を表示します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。
coreos-installer ISO Ignition サブコマンドオプション	
オプション	**説明**
-f、--force	既存の Ignition 設定を上書きします。
-i、--ignition-file <path>	使用される Ignition 設定。デフォルトは **stdin** です。
-o、--output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。
coreos-installer PXE Ignition サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。
coreos-installer PXE Ignition サブコマンドオプション	
オプション	**説明**

これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
-i、--ignition-file <path>	使用される Ignition 設定。デフォルトは **stdin** です。
-o, --output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。

7.3.14.3.4.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表7.32 coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。

引数	説明
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

7.3.14.4. RHCOS のカーネル引数でのマルチパスの有効化

RHCOS はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。

OpenShift Container Platform 4.8 以降でプロビジョニングされたノードのマルチパスを有効にできます。インストール後のサポートはマシン設定を使用してマルチパスをアクティベートすることで利用できますが、インストール中にマルチパスを有効にすることをお勧めします。

非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。

重要

IBM Z および LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z および LinuxONE への z/VM を使用したクラスターのインストール**の RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始を参照してください。

以下の手順では、インストール時にマルチパスを有効にし、**coreos-installer install** コマンドにカーネル引数を追加して、インストール済みシステム自体が初回起動からマルチパスを使用できるようにします。

前提条件

- バージョン 4.8 以降を使用する OpenShift Container Platform クラスターが実行中である。

注記

OpenShift Container Platform は、4.6 以前からアップグレードされたノードでの day-2 アクティビティーとしてのマルチパスの有効化をサポートしません。

- 管理者権限を持つユーザーとしてクラスターにログインしている。

手順

1. マルチパスを有効にして **multipathd** デーモンを起動するには、以下のコマンドを実行します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

   - 必要に応じて、PXE または ISO を起動する場合は、カーネルコマンドラインから **rd.multipath=default** を追加することで、マルチパスを有効にできます。

2. **coreos-installer** プログラムを呼び出してカーネル引数を追加します。

   - マシンに接続されているマルチパスデバイスが 1 つしかない場合は、このデバイスは **/dev/mapper/mpatha** のパスで利用できます。以下に例を示します。

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

   1 1 つのマルチパスデバイスのパスを指定します。

   - 複数のマルチパスデバイスがマシンに接続している場合には、より明示的に **/dev/mapper/mpatha** を使用する代わりに、**/dev/disk/by-id** で利用可能な World Wide Name (WWN) シンボリックリンクを使用することが推奨されます。以下に例を示します。

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
```

```
--append-karg rw
```

1 マルチパス化されたデバイスの WWN ID を指定します。例: **0xx194e957fcedb4841**

特別な **coreos.inst.*** 引数を使用してライブインストーラーを指定する場合に、このシンボリックリンクを **coreos.inst.install_dev** カーネル引数として使用することもできます。詳細は、Installing RHCOS and starting the OpenShift Container Platform bootstrap process を参照してください。

3. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) を一覧表示してカーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

出力例

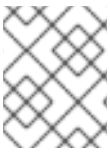
```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

7.3.14.5. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。



注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

7.3.15. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をイ

ンストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

関連情報

- インストールログの監視と、インストールの問題が発生した場合の診断データの取得の詳細については、インストールの進捗の監視 を参照してください。

7.3.16. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.3.17. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
```

```
master-2  Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

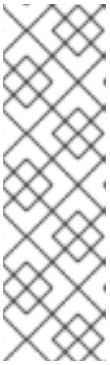
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.3.18. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
```

```
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

関連情報

- OpenShift Container Platform インストールの失敗時のデータの収集についての詳細は、失敗したインストールのログの収集 を参照してください。
- クラスター全体で Operator Pod の正常性を確認し、診断用に Operator ログを収集する手順の詳細は、Operator 関連の問題のトラブルシューティング を参照してください。

7.3.18.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

7.3.18.2. イメージレジストリーストレージの設定

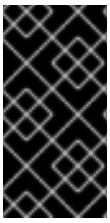
イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

7.3.18.3. ブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

7.3.19. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False     19m
```

```
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

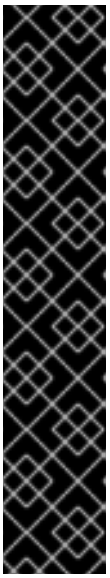
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

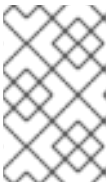
   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace> 1
   ```

   1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。



注記

マルチパスを使用してインストールする場合は、後で有効にすると問題が発生する可能性があるので、後ではなく、インストール時に有効にすることを強く推奨します。

詳細は、**ベアメタルへのインストール** の RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

7.3.20. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

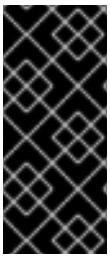
- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

7.3.21. 次のステップ

- インストールを検証します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。

7.4. ネットワークが制限された環境でのユーザーによってプロビジョニングされるベアメタルクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターをネットワークが制限された環境でプロビジョニングするベアメタルインフラストラクチャーにインストールできます。



重要

以下の手順に従って仮想化環境またはクラウド環境にクラスターをデプロイすることができますが、ベアメタルプラットフォーム以外の場合は追加の考慮事項に注意してください。このような環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

7.4.1. 前提条件

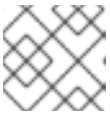
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

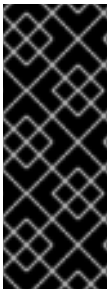
プロキシーを設定する場合は、このサイト一覧も確認してください。

7.4.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

7.4.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。

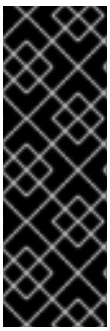
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

7.4.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

7.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

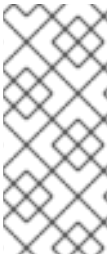
7.4.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表7.33 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

ホスト	説明
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



注記

例外として、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターで実行できます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。1 つのコンピュートマシンの実行はサポートされていません。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.4.4.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表7.34 最小リソース要件

マシン	オペレーティングシステム	CPU [1]	RAM	ストレージ	IOPS [2]
ブートストラップ	RHCOS	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS または RHEL 7.9 [3]	2	8 GB	100 GB	300

1. CPU 1 つ分は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = CPU

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd については、高速ストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

3. ユーザーによってプロビジョニングされるすべてのインストールと同様に、クラスターで RHEL 7 コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

7.4.4.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

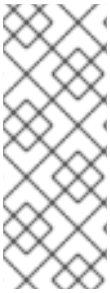
- ベアメタル環境での 3 ノードクラスターのデプロイに関する詳細は、3 ノードクラスターの設定 を参照してください。
- インストール後のクラスター証明書署名要求の承認についての詳細は、マシンの証明書署名要求の承認 を参照してください。

7.4.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。



注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

7.4.4.4.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

7.4.4.4.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表7.35 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve

プロトコル	ポート	説明
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表7.36 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.37 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

7.4.4.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード

- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーに関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表7.38 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7.4.4.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

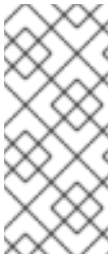
ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例7.7 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例7.8 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

7.4.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

   表7.39 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

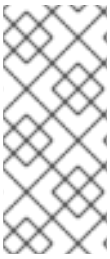
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.40 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

7.4.4.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例7.9 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

7.4.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。

- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

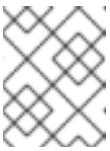
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

   注記

   DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

   注記

   DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

関連情報

- ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件
- RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始
- DHCP を使用したクラスターノードのホスト名の設定
- 高度な RHCOS インストール設定
- ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件
- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

7.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

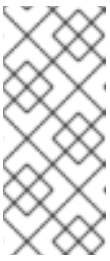
   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   注記

   出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

   random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

   ```
   $ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
   <cluster_name>.<base_domain>
   ```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

関連情報

- ユーザーによってプロビジョニングされる DNS 要件
- ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

7.4.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

注記

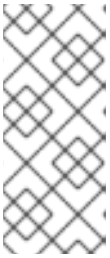
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

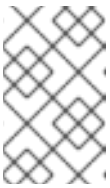
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

関連情報

- ノードの正常性の確認

7.4.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得する。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

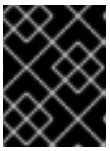
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

7.4.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

7.4.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表7.41 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

7.4.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表7.42 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

7.4.8.1.3. オプションの設定パラメーター

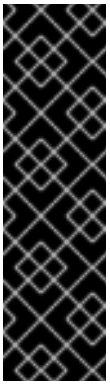
オプションのインストール設定パラメーターは、以下の表で説明されています。

表7.43 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

7.4.8.2. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

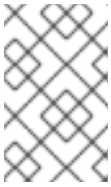
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
```

```
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



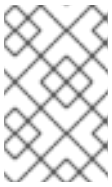
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

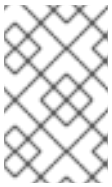
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

関連情報

- API およびアプリケーションの Ingress 負荷分散要件の詳細は、ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件 を参照してください。

7.4.8.3. インストール時のクラスター全体のプロキシーの設定

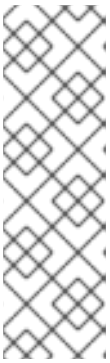
実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

注記

ベアメタルインストールでは、**install-config.yaml** ファイルの **networking.machineNetwork[].cidr** フィールドで指定される範囲にあるノード IP アドレスを割り当てない場合、それらを **proxy.noProxy** フィールドに含める必要があります。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に . を

4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.4.8.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

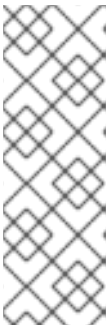
前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

7.4.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

- OpenShift Container Platform のインストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- kubelet 証明書のリカバリーに関する詳細は、コントロールプレーン証明書の期限切れの状態からのリカバリー を参照してください。

7.4.10. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定する必要があります。

手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

(1) (2) コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

(3) マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc** <mc-name> -o yaml で YAML ファイルを確認できます。

<mc-name> -o yaml で YAML ファイルを確認できます。

4 DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

- クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
- クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-chrony.yaml
```

7.4.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。

注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

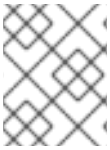
以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。
- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールす

るノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。



注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

7.4.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-
live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

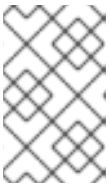
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



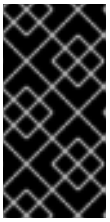
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

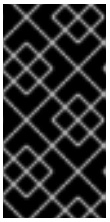
8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。
10. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

7.4.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
```

```
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```

重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーと **rootfs** ファイルを HTTP サーバーにアップロードします。
   - iPXE の場合、**kernel**、**initramfs**、および **rootfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。
   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
```

```
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2 3
```

1 1 HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

3 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

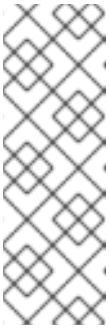
- iPXE の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

1 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

3 HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. PXE UEFI を使用する場合は、以下の操作を実行します。

   a. システムの起動に必要な **shimx64.efi** and **grubx64.efi** EFI バイナリーと **grub.cfg** ファイルを指定します。

      - ホストに RHCOS ISO をマウントしてから、**images/efiboot.img** ファイルをマウントし、必要な EFI バイナリーを展開します。

        ```
        $ mkdir -p /mnt/iso
        ```

        ```
        $ mkdir -p /mnt/efiboot
        ```

        ```
        $ mount -o loop rhcos-installer.x86_64.iso /mnt/iso
        ```

        ```
        $ mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
        ```

      - **efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

        ```
        $ cp /mnt/efiboot/EFI/redhat/shimx64.efi .
        ```

        ```
        $ cp /mnt/efiboot/EFI/redhat/grubx64.efi .
        ```

        ```
        $ umount /mnt/efiboot
        ```

        ```
        $ umount /mnt/iso
        ```

      - RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

   b. **grub.cfg** ファイルを編集し、以下のような引数を追加します。

      ```
      menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --class gnu --class os {
       linuxefi rhcos-<version>-live-kernel-<architecture> coreos.inst.install_dev=/dev/sda coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
       initrdefi rhcos-<version>-live-initramfs.<architecture>.img
      }
      ```

      詳細は以下のようになります。

rhcos-<version>-live-kernel-<architecture>

TFTP サーバーにアップロードした **kernel** ファイルを指定します。

http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img

HTTP サーバーにアップロードしたライブ rootfs イメージの場所を指定します。

http://<HTTP_server>/bootstrap.ign

HTTP サーバーにアップロードしたブートストラップ Ignition 設定ファイルの場所を指定します。

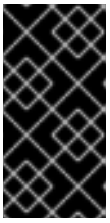
rhcos-<version>-live-initramfs.<architecture>.img

TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

UEFI ブート用に PXE サーバーを設定する方法は、Red Hat ナレッジベースの記事 How to configure/setup a PXE server for Red Hat Enterprise Linux? を参照してください。

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

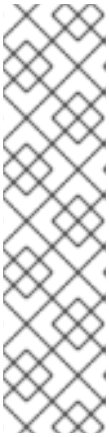
9. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

7.4.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ISO への Ignition 設定の埋め込み

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

7.4.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

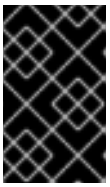
手順

1. ISO インストーラーを起動します。

2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。

3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

7.4.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。特定のアーキテクチャーの各 RHCOS ノードは、デフォルトのパーティション設定が上書きされない限り、同じパーティションレイアウトを使用します。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

以下は、OpenShift Container Platform クラスターノードへの RHCOS のインストール時に、デフォルトのパーティション設定の上書きが必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションにマウントする場合にのみ正式にサポートされます。

重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

警告

カスタムパーティションを使用すると、これらのパーティションが OpenShift Container Platform によって監視されないか、またはアラートが通知される可能性があります。デフォルトのパーティションを上書きする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

7.4.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
```

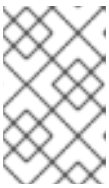
```
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のフートストラップ、コントロールフレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

7.4.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、**coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。

注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

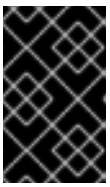
この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

7.4.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

  重要

  これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

  PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは手動で作成され、Red Hat でサポートされていないため、可能な場合は使用しないようにする必要があります。この方法では、Ignition 設定はライブインストールメディアに渡され、起動時にすぐに実行され、RHCOS システムのディスクへのインストール前後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

7.4.11.3.3.1. RHCOS ISO へのライブインストール Ignition 設定の埋め込み

ライブインストール Ignition 設定を RHCOS ISO イメージに直接埋め込むことができます。ISO イメージが起動すると、埋め込み設定が自動的に適用されます。

手順

1. 以下のイメージミラーページから **coreos-installer** バイナリーをダウンロードします: https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/

2. RHCOS ISO イメージおよび Ignition 設定ファイルを取得し、それらを **/mnt** などのアクセス可能なディレクトリーにコピーします。

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 以下のコマンドを実行して Ignition 設定を ISO に埋め込みます。

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

その ISO を使用して、指定されたライブインストール Ignition 設定を使用して RHCOS をインストールできるようになります。

重要

coreos-installer iso ignition embed を使用して、**openshift-installer** によって生成されるファイル (例: **bootstrap.ign**、**master.ign** および **worker.ign**) を埋め込むことはサポートされておらず、かつ推奨されていません。

4. 埋め込み Ignition 設定の内容を表示し、これをファイルに転送するには、以下を実行します。

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

出力例

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 設定を削除し、再利用できるように ISO をその初期状態に戻すには、以下を実行します。

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

別の Ignition 設定を ISO に埋め込むか、または ISO を初期状態で使用することができるようになりました。

7.4.11.3.4. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

7.4.11.3.4.1. ISO インストールのネットワークおよびボンディングのオプション

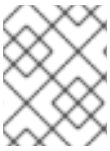
ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを

設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定

静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定され

静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

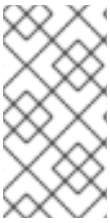
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

  ```
  ip=::10.10.10.254::::
  ```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

  ```
  rd.route=20.20.20.0/24:20.20.20.254:enp2s0
  ```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、network_interfacesは物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

7.4.11.3.4.2. ISO インストールの coreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表7.44 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**
$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。

-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--append-karg <arg>…	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>…	インストール済みシステムからデフォルトのカーネル引数を削除します。
-n、--copy-network	インストール環境からネットワーク設定をコピーします。  **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。
--save-partindex <id>…	この数または範囲でパーティションを保存します。
--insecure	署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。デフォルトは **x86_64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。

coreos-install install サブコマンド引数	
引数	**説明**
<device>	宛先デバイス。
coreos-installer ISO Ignition サブコマンド	
サブコマンド	**説明**
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer iso ignition show <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を表示します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。
coreos-installer ISO Ignition サブコマンドオプション	
オプション	**説明**
-f、--force	既存の Ignition 設定を上書きします。
-i、--ignition-file <path>	使用される Ignition 設定。デフォルトは **stdin** です。
-o、--output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。
coreos-installer PXE Ignition サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。
coreos-installer PXE Ignition サブコマンドオプション	
オプション	**説明**

これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
-i、--ignition-file <path>	使用される Ignition 設定。デフォルトは **stdin** です。
-o, --output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。

7.4.11.3.4.3. ISO または PXE インストールのcoreos.inst ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表7.45 coreos.inst ブートオプション

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。**sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。

引数	説明
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

7.4.11.4. RHCOS のカーネル引数でのマルチパスの有効化

RHCOS はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。

OpenShift Container Platform 4.8 以降でプロビジョニングされたノードのマルチパスを有効にできます。インストール後のサポートはマシン設定を使用してマルチパスをアクティベートすることで利用できますが、インストール中にマルチパスを有効にすることをお勧めします。

非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。

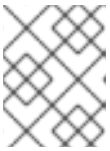
重要

IBM Z および LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z および LinuxONE への z/VM を使用したクラスターのインストール**の RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始を参照してください。

以下の手順では、インストール時にマルチパスを有効にし、**coreos-installer install** コマンドにカーネル引数を追加して、インストール済みシステム自体が初回起動からマルチパスを使用できるようにします。

前提条件

- バージョン 4.8 以降を使用する OpenShift Container Platform クラスターが実行中である。



注記

OpenShift Container Platform は、4.6 以前からアップグレードされたノードでの day-2 アクティビティーとしてのマルチパスの有効化をサポートしません。

- 管理者権限を持つユーザーとしてクラスターにログインしている。

手順

1. マルチパスを有効にして **multipathd** デーモンを起動するには、以下のコマンドを実行します。

```
$ mpathconf --enable && systemctl start multipathd.service
```

   - 必要に応じて、PXE または ISO を起動する場合は、カーネルコマンドラインから **rd.multipath=default** を追加することで、マルチパスを有効にできます。

2. **coreos-installer** プログラムを呼び出してカーネル引数を追加します。

   - マシンに接続されているマルチパスデバイスが 1 つしかない場合は、このデバイスは **/dev/mapper/mpatha** のパスで利用できます。以下に例を示します。

```
$ coreos-installer install /dev/mapper/mpatha \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

   1 1 つのマルチパスデバイスのパスを指定します。

   - 複数のマルチパスデバイスがマシンに接続している場合には、より明示的に **/dev/mapper/mpatha** を使用する代わりに、**/dev/disk/by-id** で利用可能な World Wide Name (WWN) シンボリックリンクを使用することが推奨されます。以下に例を示します。

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ 1
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
```

```
--append-karg rw
```

1 マルチパス化されたデバイスの WWN ID を指定します。例: **0xx194e957fcedb4841**

特別な **coreos.inst.*** 引数を使用してライブインストーラーを指定する場合に、このシンボリックリンクを **coreos.inst.install_dev** カーネル引数として使用することもできます。詳細は、Installing RHCOS and starting the OpenShift Container Platform bootstrap process を参照してください。

3. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) を一覧表示してカーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

出力例

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

7.4.11.5. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。

注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

7.4.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をイ

ンストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

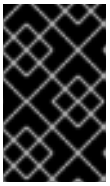
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

関連情報

- インストールログの監視と、インストールの問題が発生した場合の診断データの取得の詳細については、インストールの進捗の監視 を参照してください。

7.4.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.4.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

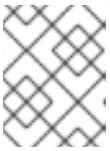
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
master-2 Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

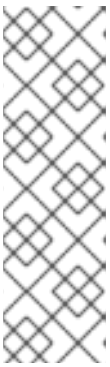
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

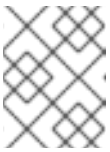
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

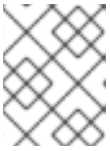
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.4.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
```

```
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

関連情報

- OpenShift Container Platform インストールの失敗時のデータの収集についての詳細は、失敗したインストールのログの収集 を参照してください。
- クラスター全体で Operator Pod の正常性を確認し、診断用に Operator ログを収集する手順の詳細は、Operator 関連の問題のトラブルシューティング を参照してください。

7.4.15.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

7.4.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

7.4.15.2.1. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、イメージレジストリー Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **ManagementState** イメージレジストリー Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

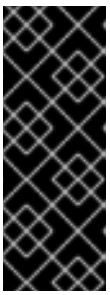
```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

7.4.15.2.2. ベアメタルおよび他の手動インストールの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージがある。



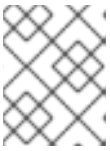
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

7.4.15.2.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

7.4.15.2.4. ブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

> **重要**
>
> ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

7.4.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
```

```
operator-lifecycle-manager-catalog        4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver  4.8.2     True        False         False      32m
service-ca                                4.8.2     True        False         False      38m
storage                                   4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver        apiserver-67b9g                             1/1     Running     0
3m
openshift-apiserver        apiserver-ljcmx                             1/1     Running     0
1m
```

```
openshift-apiserver                apiserver-z25h4                              1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
Running     0          5m
...
```

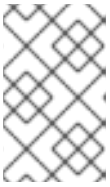
b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```
(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。



注記

マルチパスを使用してインストールする場合は、後で有効にすると問題が発生する可能性があるので、後ではなく、インストール時に有効にすることを強く推奨します。

詳細は、**ベアメタルへのインストール** の RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

4. Cluster registration ページでクラスターを登録します。

7.4.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

7.4.18. 次のステップ

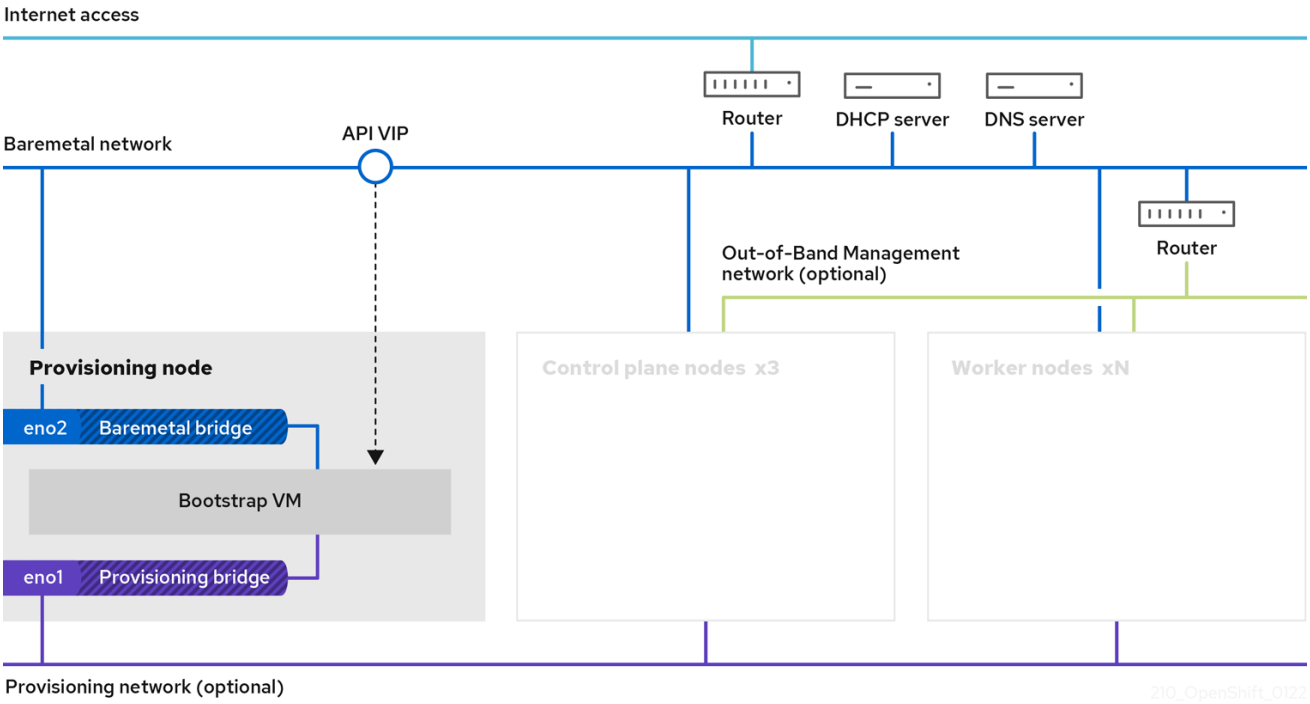
- インストールを検証します。
- クラスターをカスタマイズします。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。

- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

第8章 インストーラーでプロビジョニングされるクラスターのベアメタルへのデプロイ

8.1. 概要

ベアメタルノードへのインストーラーによってプロビジョニングされたインストールは、OpenShift Container Platform クラスターが実行されるインフラストラクチャーをデプロイおよび設定します。このガイドでは、インストーラーによってプロビジョニングされたベアメタルのインストールを正常に行うための方法論を提供します。次の図は、デプロイメントのフェーズ 1 におけるインストール環境を示しています。

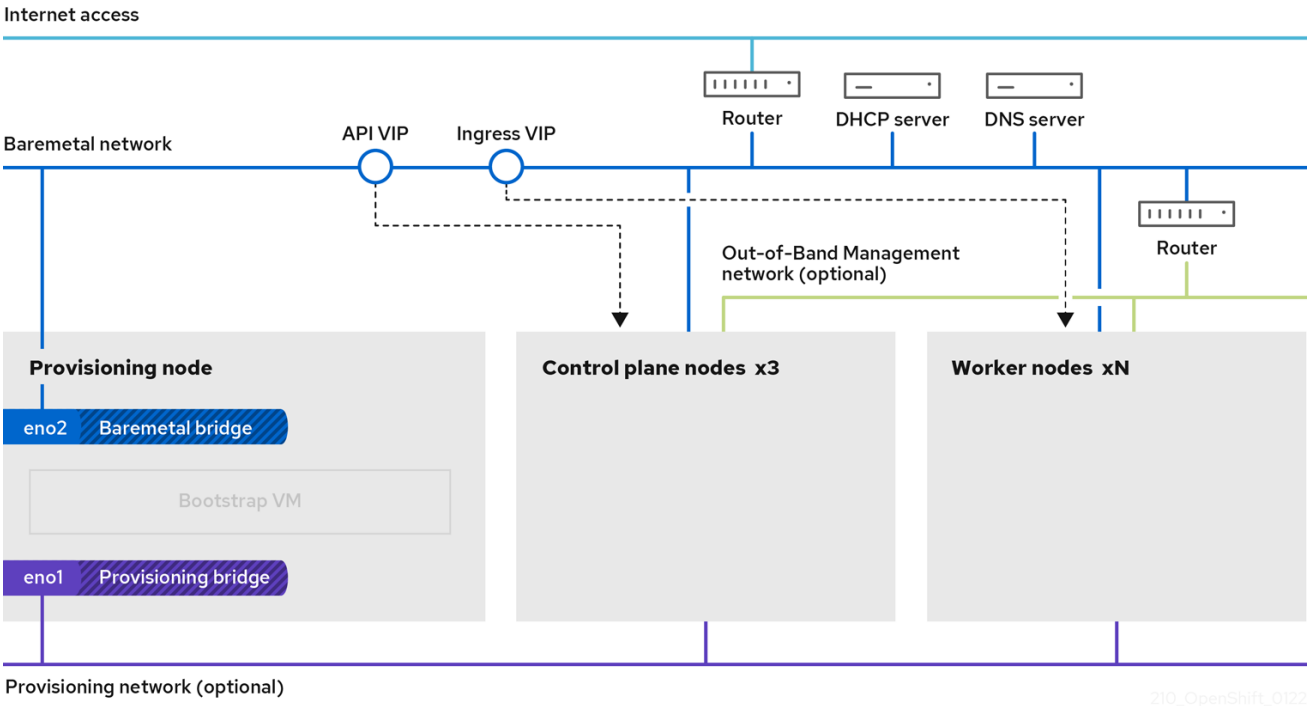


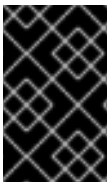
プロビジョニングノードは、インストール後に削除できます。

- **プロビジョナー**: インストールプログラムを実行し、新しい OpenShift Container Platform クラスターのコントローラーをデプロイするブートストラップ VM をホストする物理マシン。
- **Bootstrap VM**: OpenShift Container Platform クラスターのデプロイプロセスで使用される仮想マシン。
- **ネットワークブリッジ**: ブートストラップ VM は、ネットワークブリッジ **eno1** および **eno2** を介して、ベアメタルネットワークとプロビジョニングネットワーク (存在する場合) に接続します。

デプロイメントのフェーズ 2 では、プロビジョナーがブートストラップ VM を自動的に破棄し、仮想 IP アドレス (VIP) を適切なノードに移動します。API VIP はコントロールプレーンノードに移動し、Ingress VIP はワーカーノードに移動します。

次の図は、デプロイメントのフェーズ 2 を示しています:





重要

provisioning ネットワークは任意ですが、PXE ブートには必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。

8.2. 前提条件

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、以下が必要です。

1. 1 つの Red Hat Enterprise Linux (RHEL) 8.x がインストールされているプロビジョナーノード。プロビジョニングノードは、インストール後に削除できます。
2. 3 つのコントロールプレーンノード
3. ベースボード管理コントローラー (BMC) の各ノードへのアクセス。
4. 1 つ以上のネットワーク:
    a. 1 つの必須 のルーティング可能なネットワーク
    b. 1 つのオプションのノードのプロビジョニング用のネットワーク
    c. 1 つのオプションの管理ネットワーク

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールを開始する前に、ハードウェア環境が以下の要件を満たしていることを確認してください。

8.2.1. ノードの要件

インストーラーでプロビジョニングされるインストールには、ハードウェアノードの各種の要件があります。

- **CPU アーキテクチャー**: すべてのノードで **x86_64** CPU アーキテクチャーを使用する必要があります。

- **同様のノード**: Red Hat では、ノードにロールごとに同じ設定を指定することを推奨しています。つまり Red Hat では、同じ CPU、メモリー、ストレージ設定の同じブランドおよびモデルのノードを使用することを推奨しています。

- **ベースボード管理コントローラー**: **provisioner** ノードは、各 OpenShift Container Platform クラスターノードのベースボード管理コントローラー (BMC) にアクセスできる必要があります。IPMI、RedFish、または独自のプロトコルを使用できます。

- **最新の生成**: ノードは最新の生成されたノードである必要があります。インストーラーでプロビジョニングされるインストールは、ノード間で互換性を確保する必要のある BMC プロトコルに依存します。また、RHEL 8 は RAID コントローラーの最新のドライバーが同梱されています。ノードは **provisioner** ノード用に RHEL 8 を、コントローラープレーンおよびワーカーノード用に RHCOS 8 をサポートするのに必要な新しいバージョンのノードであることを確認します。

- **レジストリーノード**: (オプション) 非接続のミラーリングされていないレジストリーを設定する場合、レジストリーは独自のノードに置くことが推奨されます。

- **プロビジョナーノード**: インストーラーでプロビジョニングされるインストールには 1 つの **provisioner** ノードが必要です。

- **コントロールプレーン**: インストーラーでプロビジョニングされるインストールには、高可用性を実現するために 3 つのコントロールプレーンノードが必要です。3 つのコントロールプレーンノードのみで OpenShift Container Platform クラスターをデプロイして、コントロールプレーンノードをワーカーノードとしてスケジュール可能にすることができます。小規模なクラスターでは、開発、実稼働およびテスト時の管理者および開発者に対するリソース効率が高くなります。

- **ワーカーノード**: 必須ではありませんが、一般的な実稼働クラスターには複数のワーカーノードがあります。

重要

ワーカーノードが 1 つしかないクラスターは、パフォーマンスが低下した状態のルーターおよび Ingress トラフィックでデプロイされるので、デプロイしないでください。

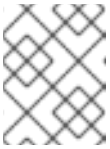
- **ネットワークインターフェイス**: 各ノードでは、ルーティング可能な **baremetal** ネットワークに 1 つ以上のネットワークインターフェイスが必要です。デプロイメントに **provisioning** ネットワークを使用する場合、各ノードに **provisioning** ネットワーク用に 1 つのネットワークインターフェイスが必要になります。**provisioning** ネットワークの使用はデフォルト設定です。ネットワークインターフェイスの命名は、プロビジョニングネットワーク用のコントロールプレーンノード全体で一貫している必要があります。たとえば、コントロールプレーンノードがプロビジョニングネットワークに **eth0** NIC を使用する場合は、他のコントロールプレーンノードもこれを使用する必要があります。

- Unified Extensible Firmware Interface (UEFI):インストーラーでプロビジョニングされるインストールでは、**provisioning** ネットワークで IPv6 アドレスを使用する場合に、すべての OpenShift Container Platform ノードで UEFI ブートが必要になります。さらに、UEFI Device PXE Settings (UEFI デバイス PXE 設定) は **provisioning** ネットワーク NIC で IPv6 プロトコルを使用するように設定する必要がありますが、**provisioning** ネットワークを省略すると、この要件はなくなります。

- **Secure Boot:** 多くの実稼働シナリオでは、UEFI ファームウェアドライバー、EFI アプリケーション、オペレーティングシステムなど、信頼できるソフトウェアのみを使用してノードが起動することを確認するために、Secure Boot が有効にされているノードが必要です。手動または管理対象の Secure Boot を使用してデプロイすることができます。

  1. **手動:** 手動で Secure Boot を使用して OpenShift Container Platform クラスターをデプロイするには、UEFI ブートモードおよび Secure Boot を各コントロールプレーンノードおよび各ワーカーノードで有効にする必要があります。Red Hat は、インストーラーでプロビジョニングされるインストールで Redfish 仮想メディアを使用している場合にのみ、手動で有効にした UEFI および Secure Boot で、Secure Boot をサポートします。詳細は、ノードの設定セクションの手動での Secure Boot のノードの設定を参照してください。

  2. **管理対象:** 管理対象 Secure Boot で OpenShift Container Platform クラスターをデプロイするには、**install-config.yaml** ファイルで **bootMode** の値を **UEFISecureBoot** に設定する必要があります。Red Hat は、第 10 世代 HPE ハードウェアおよび 13 世代 Dell ハードウェア (ファームウェアバージョン **2.75.75.75** 以上を実行) で管理対象 Secure Boot を使用したインストーラーでプロビジョニングされるインストールのみをサポートします。管理対象 Secure Boot を使用したデプロイには、Redfish 仮想メディアは必要ありません。詳細は、OpenShift インストールの環境のセットアップセクションの管理対象 Secure Boot の設定を参照してください。



注記

Red Hat は、自己生成したキーを使用する Secure Boot をサポートしません。

8.2.2. OpenShift 仮想化のためのベアメタルクラスターの計画

OpenShift 仮想化を使用する場合は、ベアメタルクラスターをインストールする前に、いくつかの要件を認識することが重要です。

- ライブマイグレーション機能を使用する場合は、**クラスターのインストール時に** 複数のワーカーノードが必要です。これは、ライブマイグレーションではクラスターレベルの高可用性 (HA) フラグを true に設定する必要があるためです。HA フラグは、クラスターのインストール時に設定され、後で変更することはできません。クラスターのインストール時に定義されたワーカーノードが 2 つ未満の場合、クラスターの存続期間中、HA フラグは false に設定されます。

注記

単一ノードのクラスターに OpenShift Virtualization をインストールできますが、単一ノードの OpenShift は高可用性をサポートしていません。

- ライブマイグレーションには共有ストレージが必要です。OpenShift Virtualization のストレージは、ReadWriteMany (RWX) アクセスモードをサポートし、使用する必要があります。

- Single Root I/O Virtualization (SR-IOV) を使用する予定の場合は、ネットワークインターフェイスコントローラー (NIC) が OpenShift Container Platform でサポートされていることを確認してください。

関連情報

- OpenShift Virtualization のクラスターの準備

- Single Root I/O Virtualization (SR-IOV) ハードウェアネットワークについて

- 仮想マシンの SR-IOV ネットワークデバイスの設定

8.2.3. 仮想メディアを使用したインストールのファームウェア要件

インストーラーでプロビジョニングされる OpenShift Container Platform クラスターのインストーラーは、Redfish 仮想メディアとのハードウェアおよびファームウェアの互換性を検証します。以下の表は、Redfish 仮想メディアを使用してデプロイされるインストーラーでプロビジョニングされる OpenShift Container Platform クラスターで機能するようにテストおよび検証された最小ファームウェアバージョンの一覧です。

表8.1 Redfish 仮想メディアのファームウェアの互換性

ハードウェア	モデル	管理	ファームウェアのバージョン
HP	第 10 世代	iLO5	2.63 以降
Dell	第 14 世代	iDRAC 9	v4.20.20.20 - v4.40.00.00 のみ
	第 13 世代	iDRAC 8	v2.75.75.75 以降

注記

Red Hat は、ファームウェア、ハードウェア、またはその他のサードパーティーコンポーネントの組み合わせをすべてテストしません。サードパーティーサポートの詳細は、Red Hat サードパーティーサポートポリシー を参照してください。

ファームウェアの更新については、ノードのハードウェアドキュメントを参照するか、ハードウェアベンダーにお問い合わせください。

HP サーバーの場合、Ironic は、仮想メディアで iLO4 をサポートしないので、Redfish 仮想メディアは、iLO4 を実行する第 9 世代のシステムではサポートされません。

Dell サーバーの場合、OpenShift Container Platform クラスターノードについて、iDRAC コンソールで AutoAttach が有効にされていることを確認します。メニューパスは以下のようになります。**Configuration → Virtual Media → Attach Mode → AutoAttach**iDRAC 9 ファームウェアバージョン **04.40.00.00** では、仮想コンソールプラグインがデフォルトで **eHTML5** になり、**InsertVirtualMedia** ワークフローで問題が発生します。この問題を回避するには、プラグインを **HTML5** に設定します。メニューパスは以下の通りです。**Configuration → Virtual console → Plug-in Type → HTML5**

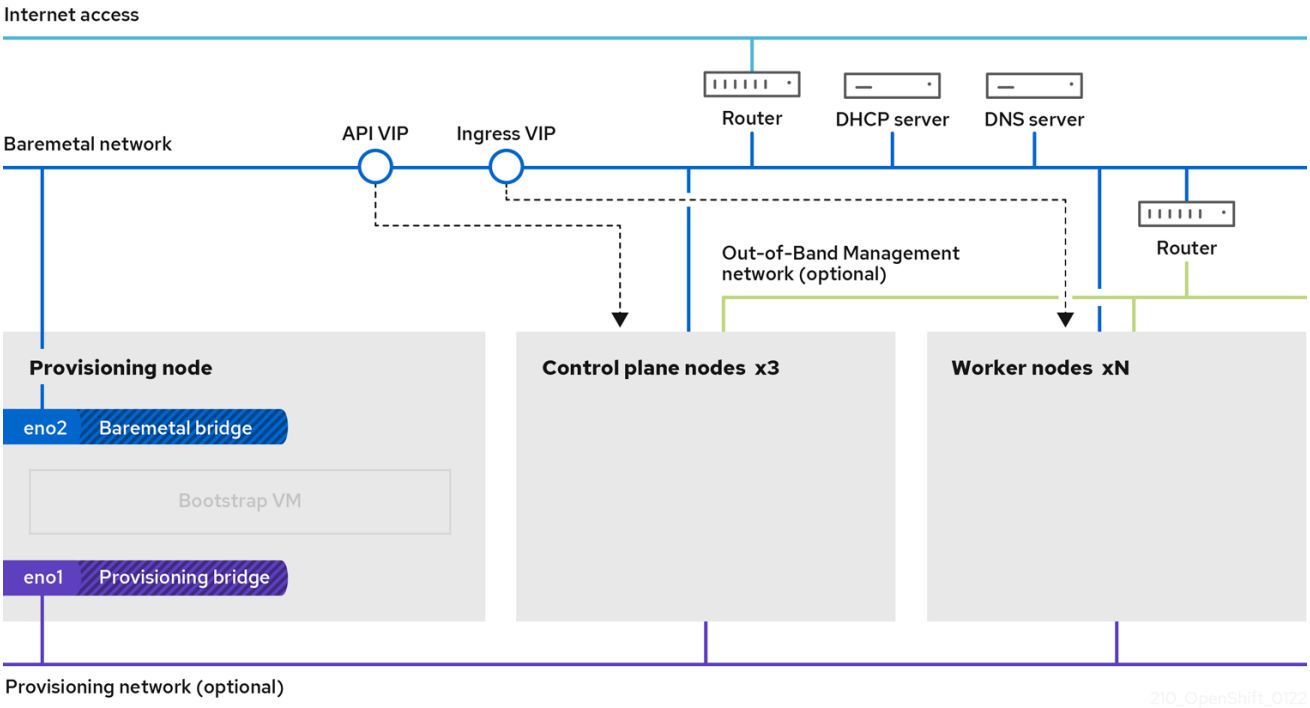
重要

インストーラーは、仮想メディアを使用したインストール時にノードのファームウェアが上記のバージョンよりも古いバージョンの場合にノードでインストールを開始しません。

8.2.4. ネットワーク要件

OpenShift Container Platform のインストーラーでプロビジョニングされるインストールには、複数のネットワーク要件があります。まず、インストーラーでプロビジョニングされるインストールでは、各ベアメタルノードにオペレーティングシステムをプロビジョニングするためのルーティング不可能な

provisioning ネットワークをオプションで使用します。次に、インストーラーでプロビジョニングされるインストールでは、ルーティング可能な **baremetal** ネットワークを使用します。



8.2.4.1. ネットワーク MTU の増加

OpenShift Container Platform をデプロイする前に、ネットワークの最大伝送単位 (MTU) を 1500 以上に増やします。MTU が 1500 未満の場合、ノードの起動に使用される Ironic イメージが Ironic インスペクター Pod との通信に失敗し、検査が失敗する可能性があります。これが発生すると、インストールにノードを使用できないため、インストールが停止します。

8.2.4.2. NIC の設定

OpenShift Container Platform は、2 つのネットワークを使用してデプロイします。

- **provisioning**: **provisioning** ネットワークは、OpenShift Container Platform クラスターの一部である基礎となるオペレーティングシステムを各ノードにプロビジョニングするために使用されるオプションのルーティング不可能なネットワークです。各クラスターノードの **provisioning** ネットワークのネットワークインターフェイスには、BIOS または UEFI が PXE ブートに設定されている必要があります。
  provisioningNetworkInterface 設定は、コントロールプレーンノード上の **provisioning** ネットワークの NIC 名を指定します。これは、コントロールプレーンノードと同じでなければなりません。**bootMACAddress** 設定設定は、**provisioning** ネットワーク用に各ノードで特定の NIC を指定する手段を提供します。

  provisioning ネットワークは任意ですが、PXE ブートには必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。

- **baremetal**: **baremetal** ネットワークはルーティング可能なネットワークです。NIC が **provisioning** ネットワークを使用するように設定されていない場合には、**baremetal** ネットワークとのインターフェイスには任意の NIC を使用することができます。

重要

VLAN を使用する場合、それぞれの NIC は、適切なネットワークに対応する別個の VLAN 上にある必要があります。

8.2.4.3. DNS 要件

クライアントは、**baremetal** ネットワークで OpenShift Container Platform クラスターにアクセスします。ネットワーク管理者は、正規名の拡張がクラスター名であるサブドメインまたはサブゾーンを設定する必要があります。

```
<cluster_name>.<base_domain>
```

以下に例を示します。

```
test-cluster.example.com
```

OpenShift Container Platform には、クラスターメンバーシップ情報を使用して A/AAAA レコードを生成する機能が含まれます。これにより、ノード名が IP アドレスに解決されます。ノードが API に登録されると、クラスターは CoreDNS-mDNS を使用せずにこれらのノード情報を分散できます。これにより、マルチキャスト DNS に関連付けられたネットワークトラフィックがなくなります。

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード Ingress API

A/AAAA レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。Red Hat Enterprise Linux Core OS(RHCOS) は、逆引きレコードまたは DHCP を使用して、すべてのノードのホスト名を設定します。

インストーラーがプロビジョニングしたインストールには、クラスターメンバーシップ情報を使用して A/AAAA レコードを生成する機能が含まれています。これにより、ノード名が IP アドレスに解決されます。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表8.2 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	A/AAAA レコードと PTR レコード。API ロードバランサーを識別します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	ワイルドカード A/AAAA レコードは、アプリケーションの Ingress ロードバランサーを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するノードをターゲットにします。Ingress Controller Pod は、デフォルトでワーカーノードで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。

ヒント

digコマンドを使用して、DNS 解決を確認できます。

8.2.4.4. Dynamic Host Configuration Protocol (DHCP) の要件

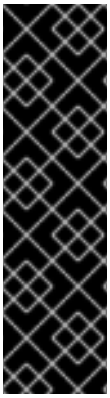
デフォルトでは、インストーラーでプロビジョニングされるインストールは、**provisioning** ネットワーク用に DHCP を有効にして **ironic-dnsmasq** をデプロイします。**provisioningNetwork** 設定が、デフォルト値の **managed** に設定されている場合、**provisioning** ネットワーク上で他の DHCP サーバーを実行することはできません。**provisioning** ネットワーク上で DHCP サーバーを実行している場合は、**install-config.yaml** ファイルで **provisioningNetwork** 設定を **unmanaged** に設定する必要があります。

ネットワーク管理者は、外部 DHCP サーバー上の **baremetal** ネットワーク用に、OpenShift Container Platform クラスター内の各ノードの IP アドレスを予約する必要があります。

8.2.4.5. DHCP サーバーを使用するノードの IP アドレスの確保

baremetal ネットワークの場合、ネットワーク管理者は以下を含む多数の IP アドレスを予約する必要があります。

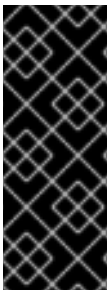
1. 2 つの一意の仮想 IP アドレス。
    - API エンドポイントの 1 つの仮想 IP アドレス。
    - ワイルドカード Ingress エンドポイントの 1 つの仮想 IP アドレス
2. プロビジョナーノードの 1 つの IP アドレス
3. 各コントロールプレーン (マスター) ノード 1 つの IP アドレス
4. 各ワーカーノードの 1 つの IP アドレス (適用可能な場合)



IP アドレスの予約し、それらを静的 IP アドレスにする

一部の管理者は、各ノードの IP アドレスが DHCP サーバーがない状態で一定になるように静的 IP アドレスの使用を選択します。OpenShift Container Platform クラスターで静的 IP アドレスを使用するには、IP アドレスを無限リースで予約します。デプロイメント時に、インストーラーは DHCP で割り当てられたアドレスから静的 IP アドレスに NIC を再設定します。無限ではない DHCP リースを持つ NIC は、DHCP を使用するように設定された状態になります。

IP アドレスを無限リースで設定することは、Machine Config Operator を使用してデプロイされるネットワーク設定と互換性がありません。

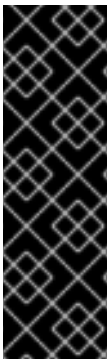


DHCP サーバーで無限のリースを提供できることの確認

rfc2131 で指定されているように無限リースを適切に設定するには、DHCP サーバーでの DHCP 有効期限を 4294967295 秒に指定する必要があります。DHCP の無限リース時間に返された値がそれよりも小さい値であった場合には、ノードはエラーを報告し、ノードに永続的な IP は設定されません。RHEL 8 では **dhcpd** は無限のリースが提供されません。プロビジョナーノードを使用して、リース時間が無限の動的 IP アドレスを提供する場合は、 **dhcpd** ではなく **dnsmasq** を使用します。

外部ロードバランサーとコントロールプレーンノード間のネットワーク

外部の負荷分散サービスとコントロールプレーンノードは同じ L2 ネットワークで実行する必要があります。また、VLAN を使用して負荷分散サービスとコントロールプレーンノード間のトラフィックをルーティングする際に同じ VLAN で実行する必要があります。



デプロイ後に IP アドレスを手動で変更しないでください。

デプロイメント後にワーカーノードの IP アドレスを手動で変更しないでください。デプロイメント後にワーカーノードの IP アドレスを変更するには、ワーカーノードがスケジュール対象外としてマークし、Pod を退避してノードを削除し、これを新規 IP アドレスで再作成する必要があります。詳細は、ノードの操作を参照してください。デプロイメント後にコントロールプレーンノードの IP アドレスを変更するには、サポートにお問い合わせください。

ストレージインターフェイスには DHCP 予約が必要です。

以下の表は、完全修飾ドメイン名の具体例を示しています。API および Nameserver アドレスは、正式名の拡張子で始まります。コントロールプレーンおよびワーカーノードのホスト名は例であるため、任意のホストの命名規則を使用することができます。

使用法	ホスト名	IP
API	**api.<cluster_name>.<base_domain>**	**<ip>**
Ingress LB (アプリケーション)	***.apps.<cluster_name>.<base_domain>**	**<ip>**
プロビジョナーノード	**provisioner.<cluster_name>.<base_domain>**	**<ip>**

使用法	ホスト名	IP
Master-0	**openshift-master-0.<cluster_name>.<base_domain>**	**<ip>**
Master-1	**openshift-master-1.<cluster_name>-.<base_domain>**	**<ip>**
Master-2	**openshift-master-2.<cluster_name>.<base_domain>**	**<ip>**
Worker-0	**openshift-worker-0.<cluster_name>.<base_domain>**	**<ip>**
Worker-1	**openshift-worker-1.<cluster_name>.<base_domain>**	**<ip>**
Worker-n	**openshift-worker-n.<cluster_name>.<base_domain>**	**<ip>**

注記

DHCP 予約を作成しない場合には、インストーラーは、Kubernetes API ノード、プロビジョナーノード、コントロールプレーンノード、およびワーカーノードのホスト名を設定するために逆引き DNS 解決を必要とします。

8.2.4.6. ネットワークタイムプロトコル (NTP)

クラスター内の各 OpenShift Container Platform ノードは NTP サーバーにアクセスできる必要があります。OpenShift Container Platform ノードは NTP を使用してクロックを同期します。たとえば、クラスターノードは、検証を必要とする SSL 証明書を使用します。これは、ノード間の日付と時刻が同期していない場合に失敗する可能性があります。

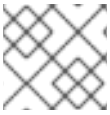
重要

各クラスターノードの BIOS 設定で一貫性のあるクロックの日付と時刻の形式を定義してください。そうしないと、インストールが失敗する可能性があります。

切断されたクラスター上で NTP サーバーとして機能するようにコントロールプレーンノードを再設定し、コントロールプレーンノードから時間を取得するようにワーカーノードを再設定することができます。

8.2.4.7. ステートドリブンのネットワーク設定要件 (テクノロジープレビュー)

OpenShift Container Platform は、 **kubernetes-nmstate** を使用し、クラスターノードのセカンダリーネットワークインターフェイスで、インストール後の追加のステートドリブンのネットワーク設定をサポートします。たとえば、システム管理者は、ストレージネットワークのインストール後に、クラスターノードにセカンダリーネットワークインターフェイスを設定することができます。



注記

設定は、Pod のスケジュール前に行われる必要があります。

ステートドリブンのネットワーク設定では、**kubernetes-nmstate** をインストールする必要があり、クラスターノード上で Network Manager を実行する必要もあります。詳細は、**OpenShift Virtualization > Kubernetes NMState (テクノロジープレビュー)** を参照してください。

8.2.4.8. 帯域外管理 IP アドレスのポートアクセス

帯域外管理 IP アドレスは、ノードとは別のネットワーク上にあります。インストール中に帯域外管理がベアメタルノードと通信できるようにするには、帯域外管理 IP アドレスアドレスに **TCP6180** ポートへのアクセスを許可する必要があります。

8.2.5. ノードの設定

provisioning ネットワークを使用する場合のノードの設定
クラスター内の各ノードには、適切なインストールを行うために以下の設定が必要です。

警告

ノード間で一致していないと、インストールに失敗します。

クラスターノードには 3 つ以上の NIC を追加できますが、インストールプロセスでは最初の 2 つの NIC のみに焦点が当てられます。

NIC	ネットワーク	VLAN
NIC1	**provisioning**	<provisioning_vlan>
NIC2	**baremetal**	<baremetal_vlan>

NIC1 は、OpenShift Container Platform クラスターのインストールにのみ使用されるルーティング不可能なネットワーク (**provisioning**) です。

プロビジョナーノードでの Red Hat Enterprise Linux (RHEL) 8.x インストールプロセスは異なる可能性があります。ローカルの Satellite サーバー、PXE サーバー、PXE 対応の NIC2 を使用して Red Hat Enterprise Linux (RHEL) 8.x をインストールするには、以下のようになります。

PXE	ブート順序
NIC1 PXE 対応の **provisioning** ネットワーク	1
NIC2 **baremetal** ネットワークPXE 対応はオプションです。	2

注記

他のすべての NIC で PXE が無効になっていることを確認します。

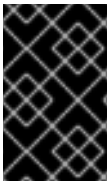
コントロールプレーンおよびワーカーノードを以下のように設定します。

PXE	ブート順序
NIC1 PXE 対応 (プロビジョニングネットワーク)	1

provisioning ネットワークを使用しないノードの設定
インストールプロセスには、1 つの NIC が必要です。

NIC	ネットワーク	VLAN
NICx	**baremetal**	<baremetal_vlan>

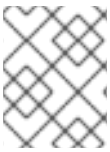
NICx は、OpenShift Container Platform クラスターのインストールに使用されるルーティング可能なネットワーク (**baremetal**) であり、インターネットにルーティング可能です。



重要

provisioning ネットワークは任意ですが、PXE ブートには必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。

手動での Secure Boot のノードの設定
Secure Boot は、ノードが UEFI ファームウェアドライバー、EFI アプリケーション、オペレーティングシステムなどの信頼できるソフトウェアのみを使用していることを確認できない場合は、ノードの起動を阻止します。



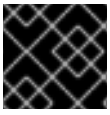
注記

Red Hat は、RedFish 仮想メディアを使用してデプロイする場合にのみ、手動で設定された Secure Boot をサポートします。

Secure Boot を手動で有効にするには、ノードのハードウェアガイドを参照し、以下を実行してください。

手順

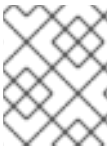
1. ノードを起動し、BIOS メニューを入力します。
2. ノードのブートモードを UEFI Enabled に設定します。
3. Secure Boot を有効にします。



重要

Red Hat は、自己生成したキーを使用する Secure Boot をサポートしません。

Fujitsu iRMC の互換性サポートモジュールの設定
互換性サポートモジュール (CSM) 設定は、UEFI システムとのレガシー BIOS 後方互換性をサポートします。Fujitsu iRMC を使用してクラスターをデプロイする場合は、CSM を設定する必要があります。そうしないと、インストールが失敗する可能性があります。



注記

特定のノードタイプの CSM の設定については、ノードのハードウェアガイドを参照してください。

前提条件

- セキュアブートコントロールを無効化したことを確認する。**Security → Secure Boot Configuration → Secure Boot Control** で、この機能を無効化できます。

手順

1. ノードを起動し、BIOS メニューを選択します。
2. **Advanced** タブで、リストから **CSM Configuration** を選択します。
3. **Launch CSM** オプションを有効にして、次の値を設定します。

項目	値
起動オプションフィルター	UEFI およびレガシー
Launch PXE OpROM Policy	UEFI のみ
ストレージ OpROM ポリシーの起動	UEFI のみ
その他の PCI デバイス ROM の優先順位	UEFI のみ

8.2.6. アウトオブバンド管理 (Out-of-band Management: 帯域外管理)

ノードには通常、ベースボード管理コントローラー (BMC) が使用する追加の NIC があります。これらの BMC は provisioner ノードからアクセスできる必要があります。

各ノードは、アウトバウンド管理でアクセスできるようにする必要があります。アウトバウンド管理ネットワークを使用する場合、 provisioner ノードには、OpenShift Container Platform 4 の正常なインストールを実行するためにアウトバウンドネットワークへのアクセスが必要になります。

このアウトバウンド管理設定については、本書では扱いません。アウトバウンド管理には、別個の管理ネットワークを設定することを推奨します。ただし、**provisioning** ネットワークまたは **baremetal** ネットワークの使用は有効なオプションになります。

8.2.7. インストールに必要なデータ

OpenShift Container Platform クラスターのインストール前に、すべてのクラスターノードから以下の情報を収集します。

- アウトバウンド管理 IP

- 例
  - Dell (iDRAC) IP
  - HP (iLO) IP
  - Fujitsu (iRMC) IP

provisioning ネットワークを使用する場合

- NIC (**provisioning**) MAC アドレス
- NIC (**baremetal**) MAC アドレス

provisioning ネットワークを省略する場合

- NIC (**baremetal**) MAC アドレス

8.2.8. ノードの検証チェックリスト

provisioning ネットワークを使用する場合

❑ NIC1 VLAN が **provisioning** ネットワークについて設定されている。

❑ **provisioning** ネットワークの NIC1 は、プロビジョナー、コントロールプレーン (マスター)、およびワーカーノードで PXE 対応として使用できる。

❑ NIC2 VLAN が **baremetal** ネットワークについて設定されている。

❑ PXE が他のすべての NIC で無効にされている。

❑ DNS は API および Ingress エンドポイントで設定されている。

❑ コントロールプレーンおよびワーカーノードが設定されている。

❑ すべてのノードがアウトオブバンド管理 (Out-of-band Management: 帯域外管理) でアクセス可能である。

❑ (オプション) 別個の管理ネットワークが作成されている。

❑ インストールに必要なデータ。

provisioning ネットワークを省略する場合

❑ NIC1 VLAN が **baremetal** ネットワークについて設定されている。

❑ DNS は API および Ingress エンドポイントで設定されている。

❑ コントロールプレーンおよびワーカーノードが設定されている。

❑ すべてのノードがアウトオブバンド管理 (Out-of-band Management: 帯域外管理) でアクセス可能である。

❑ (オプション) 別個の管理ネットワークが作成されている。

❑ インストールに必要なデータ。

8.3. OPENSHIFT インストールの環境のセットアップ

8.3.1. RHEL のプロビジョナーノードへのインストール

ネットワーク設定が完了すると、次の手順では、プロビジョナーノードに RHEL 8.x をインストールします。インストーラーは、OpenShift Container Platform クラスターをインストールする間にプロビジョナーノードをオーケレーターとして使用します。本書の目的上、RHEL のプロビジョナーノードへのインストールは対象外です。ただし、オプションには、RHEL Satellite サーバー、PXE、またはインストールメディアの使用も含まれますが、これらに限定されません。

8.3.2. OpenShift Container Platform インストールのプロビジョナーノードの準備

環境を準備するには、以下の手順を実行します。

手順

1. **ssh** でプロビジョナーノードにログインします。
2. root 以外のユーザー (**kni**) を作成し、そのユーザーに **sudo** 権限を付与します。

```
# useradd kni
# passwd kni
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
# chmod 0440 /etc/sudoers.d/kni
```

3. 新規ユーザーの **ssh** キーを作成します。

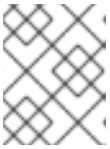
```
# su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ""
```

4. プロビジョナーノードで新規ユーザーとしてログインします。

```
# su - kni
$
```

5. Red Hat Subscription Manager を使用してプロビジョナーノードを登録します。

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms --
enable=rhel-8-for-x86_64-baseos-rpms
```



注記

Red Hat Subscription Manager についての詳細は、Using and Configuring Red Hat Subscription Manager を参照してください。

6. 以下のパッケージをインストールします。

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. ユーザーを変更して、新たに作成したユーザーに **libvirt** グループを追加します。

```
$ sudo usermod --append --groups libvirt <user>
```

–

8. **firewalld** を再起動して、**http** サービスを有効にします。

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --zone=public --add-service=http --permanent
$ sudo firewall-cmd --reload
```

9. **libvirtd** サービスを開始して、これを有効にします。

```
$ sudo systemctl enable libvirtd --now
```

10. **default** ストレージプールを作成して、これを起動します。

```
$ sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
$ sudo virsh pool-start default
$ sudo virsh pool-autostart default
```

11. ネットワークの設定

注記

Web コンソールからネットワークを設定することもできます。

baremetal ネットワーク NIC 名をエクスポートします。

```
$ export PUB_CONN=<baremetal_nic_name>
```

baremetal ネットワークを設定します。

```
$ sudo nohup bash -c "
    nmcli con down \"$PUB_CONN\"
    nmcli con delete \"$PUB_CONN\"
    # RHEL 8.1 appends the word \"System\" in front of the connection, delete in case it exists
    nmcli con down \"System $PUB_CONN\"
    nmcli con delete \"System $PUB_CONN\"
    nmcli connection add ifname baremetal type bridge con-name baremetal
    nmcli con add type bridge-slave ifname \"$PUB_CONN\" master baremetal
    pkill dhclient;dhclient baremetal
"
```

provisioning ネットワークを使用してデプロイする場合は、**provisioning** ネットワークの NIC 名をエクスポートします。

```
$ export PROV_CONN=<prov_nic_name>
```

provisioning ネットワークを使用してデプロイする場合は、**provisioning** ネットワークを設定します。

```
$ sudo nohup bash -c "
    nmcli con down \"$PROV_CONN\"
    nmcli con delete \"$PROV_CONN\"
    nmcli connection add ifname provisioning type bridge con-name provisioning
    nmcli con add type bridge-slave ifname \"$PROV_CONN\" master provisioning
```

```
    nmcli connection modify provisioning ipv6.addresses fd00:1101::1/64 ipv6.method manual
    nmcli con down provisioning
    nmcli con up provisioning
"
```

注記

これらのステップの実行後に **ssh** 接続が切断される可能性があります。

IPv6 アドレスには、**baremetal** ネットワーク経由でルーティング可能でない限り、任意のアドレスを使用できます。

IPv6 アドレスを使用する場合に UEFI PXE 設定が有効にされており、UEFI PXE 設定が IPv6 プロトコルに設定されていることを確認します。

12. **provisioning** ネットワーク接続で IPv4 アドレスが設定されている。

```
$ nmcli connection modify provisioning ipv4.addresses 172.22.0.254/24 ipv4.method manual
```

13. **provisioner** ノードに対して再度 **ssh** を実行します (必要な場合)。

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

14. 接続ブリッジが適切に作成されていることを確認します。

```
$ sudo nmcli con show
```

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eno1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eno1
bridge-slave-eno2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eno2
```

15. **pull-secret.txt** ファイルを作成します。

```
$ vim pull-secret.txt
```

Web ブラウザーで、Install OpenShift on Bare Metal with installer-provisioned infrastructure に移動し、**Downloads** セクションまでスクロールダウンします。**Copy pull secret** をクリックします。**pull-secret.txt** ファイルにコンテンツを貼り付け、そのコンテンツを **kni** ユーザーのホームディレクトリーに保存します。

8.3.3. OpenShift Container Platform インストーラーの取得

インストーラーの **latest-4.x** バージョンを使用して、OpenShift Container Platform の最新の一般公開バージョンをデプロイします。

```
$ export VERSION=latest-4.8
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

8.3.4. OpenShift Container Platform インストールの展開

インストーラーの取得後、次のステップとしてこれを展開します。

手順

1. 環境変数を設定します。

```
$ export cmd=openshift-baremetal-install
$ export pullsecret_file=~/pull-secret.txt
$ export extract_dir=$(pwd)
```

2. **oc** バイナリーを取得します。

```
$ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. インストーラーを実行します。

```
$ sudo cp oc /usr/local/bin
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to "${extract_dir}" ${RELEASE_IMAGE}
$ sudo cp openshift-baremetal-install /usr/local/bin
```

8.3.5. RHCOS イメージキャッシュの作成 (オプション)

イメージのキャッシュを使用するには、ブートストラップ仮想マシンによって使用される Red Hat Enterprise Linux CoreOS (RHCOS) イメージと、異なるノードをプロビジョニングするためにインストーラーによって使用される RHCOS イメージという 2 つのイメージをダウンロードする必要があります。イメージのキャッシュはオプションですが、帯域幅が制限されたネットワークでインストーラーを実行する場合にとくに役立ちます。

帯域幅が制限されたネットワークでインストーラーを実行し、RHCOS イメージのダウンロードに 15 分から 20 分を超える時間がかかる場合、インストーラーはタイムアウトします。このような場合、Web サーバーでイメージをキャッシュすることができます。

イメージを含むコンテナーをインストールします。

手順

1. **podman** をインストールします。

```
$ sudo dnf install -y podman
```

2. RHCOS イメージのキャッシュに使用されるファイアウォールのポート **8080** を開きます。

```
$ sudo firewall-cmd --add-port=8080/tcp --zone=public --permanent
```

```
$ sudo firewall-cmd --reload
```

3. **bootstraposimage** および **clusterosimage** を保存するディレクトリーを作成します。

```
$ mkdir /home/kni/rhcos_image_cache
```

4. 新規に作成されたディレクトリーに適切な SELinux コンテキストを設定します。

```
$ sudo semanage fcontext -a -t httpd_sys_content_t "/home/kni/rhcos_image_cache(/.*)?"
```

```
$ sudo restorecon -Rv rhcos_image_cache/
```

5. インストーラーからコミット ID を取得します。

```
$ export COMMIT_ID=$(/usr/local/bin/openshift-baremetal-install version | grep '^built from commit' | awk '{print $4}')
```

ID は、インストーラーがダウンロードする必要のあるイメージを判別します。

6. インストーラーがノードにデプロイする RHCOS イメージの URI を取得します。

```
$ export RHCOS_OPENSTACK_URI=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq .images.openstack.path | sed 's/"//g')
```

7. インストーラーがブートストラップ仮想マシンにデプロイする RHCOS イメージの URI を取得します。

```
$ export RHCOS_QEMU_URI=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq .images.qemu.path | sed 's/"//g')
```

8. イメージが公開されるパスを取得します。

```
$ export RHCOS_PATH=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq .baseURI | sed 's/"//g')
```

9. ブートストラップ仮想マシンにデプロイされる RHCOS イメージの SHA ハッシュを取得します。

```
$ export RHCOS_QEMU_SHA_UNCOMPRESSED=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq -r '.images.qemu["uncompressed-sha256"]')
```

10. ノードにデプロイされる RHCOS イメージの SHA ハッシュを取得します。

```
$ export RHCOS_OPENSTACK_SHA_COMPRESSED=$(curl -s -S https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq -r '.images.openstack.sha256')
```

11. イメージをダウンロードして、それらを **/home/kni/rhcos_image_cache** ディレクトリーに配置します。

```
$ curl -L ${RHCOS_PATH}${RHCOS_QEMU_URI} -o /home/kni/rhcos_image_cache/${RHCOS_QEMU_URI}
```

```
$ curl -L ${RHCOS_PATH}${RHCOS_OPENSTACK_URI} -o
/home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_URI}
```

12. SELinux タイプが、新しく作成されたファイルの **httpd_sys_content_t** であることを確認します。

```
$ ls -Z /home/kni/rhcos_image_cache
```

13. Pod を作成します。

```
$ podman run -d --name rhcos_image_cache \
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
quay.io/centos7/httpd-24-centos7:latest
```

上記のコマンドは、デプロイメントのイメージを提供する **rhcos_image_cache** という名前のキャッシュ Web サーバーを作成します。最初のイメージ **${RHCOS_PATH}${RHCOS_QEMU_URI}?sha256=${RHCOS_QEMU_SHA_UNCOMPRESSED}** は **bootstrapOSImage** で、2 つ目のイメージ **${RHCOS_PATH}${RHCOS_OPENSTACK_URI}?sha256=${RHCOS_OPENSTACK_SHA_COMPRESSED}** は **install-config.yaml** ファイルの **clusterOSImage** です。

14. **bootstrapOSImage** および **clusterOSImage** 設定を生成します。

```
$ export BAREMETAL_IP=$(ip addr show dev baremetal | awk '/inet /{print $2}' | cut -d"/" -f1)
```

```
$ export RHCOS_OPENSTACK_SHA256=$(zcat
/home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_URI} | sha256sum | awk '{print
$1}')
```

```
$ export RHCOS_QEMU_SHA256=$(zcat
/home/kni/rhcos_image_cache/${RHCOS_QEMU_URI} | sha256sum | awk '{print $1}')
```

```
$ export
CLUSTER_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_OPENSTACK_URI}?
sha256=${RHCOS_OPENSTACK_SHA256}"
```

```
$ export
BOOTSTRAP_OS_IMAGE="http://${BAREMETAL_IP}:8080/${RHCOS_QEMU_URI}?
sha256=${RHCOS_QEMU_SHA256}"
```

```
$ echo "${RHCOS_OPENSTACK_SHA256}  ${RHCOS_OPENSTACK_URI}" >
/home/kni/rhcos_image_cache/rhcos-ootpa-latest.qcow2.sha256sum
```

```
$ echo "    bootstrapOSImage=${BOOTSTRAP_OS_IMAGE}"
```

```
$ echo "    clusterOSImage=${CLUSTER_OS_IMAGE}"
```

15. **platform.baremetal** 下の **install-config.yaml** ファイルに必要な設定を追加します。

```
platform:
  baremetal:
    bootstrapOSImage: http://<BAREMETAL_IP>:8080/<RHCOS_QEMU_URI>?sha256=
<RHCOS_QEMU_SHA256>
    clusterOSImage: http://<BAREMETAL_IP>:8080/<RHCOS_OPENSTACK_URI>?sha256=
<RHCOS_OPENSTACK_SHA256>
```

詳細は、設定ファイルのセクションを参照してください。

8.3.6. 設定ファイル

8.3.6.1. install-config.yaml ファイルの設定

install-config.yaml ファイルには、追加の詳細情報が必要です。それらの情報のほとんどは、インストーラーおよび結果として作成されるクラスターが利用可能なハードウェアを完全に管理するのに必要な利用可能なハードウェアについての十分な情報として提供されます。

1. **install-config.yaml** を設定します。**pullSecret** および **sshKey** など、環境に合わせて適切な変数を変更します。

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster-name>
networking:
  machineCIDR: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2 (1)
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api-ip>
    ingressVIP: <wildcard-ip>
    provisioningNetworkCIDR: <CIDR>
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (2)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        rootDeviceHints:
         deviceName: "/dev/disk/by-id/<disk_id>" (3)
      - name: <openshift-master-1>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (4)
```

```
      username: <user>
      password: <password>
    bootMACAddress: <NIC1-mac-address>
    rootDeviceHints:
     deviceName: "/dev/disk/by-id/<disk_id>" 5
  - name: <openshift-master-2>
    role: master
    bmc:
      address: ipmi://<out-of-band-ip> 6
      username: <user>
      password: <password>
    bootMACAddress: <NIC1-mac-address>
    rootDeviceHints:
     deviceName: "/dev/disk/by-id/<disk_id>" 7
  - name: <openshift-worker-0>
    role: worker
    bmc:
      address: ipmi://<out-of-band-ip> 8
      username: <user>
      password: <password>
    bootMACAddress: <NIC1-mac-address>
  - name: <openshift-worker-1>
    role: worker
    bmc:
      address: ipmi://<out-of-band-ip>
      username: <user>
      password: <password>
    bootMACAddress: <NIC1-mac-address>
    rootDeviceHints:
     deviceName: "/dev/disk/by-id/<disk_id>" 9
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

1 OpenShift Container Platform クラスターの一部であるワーカーノードの数に基づいてワーカーマシンをスケーリングします。

2 4 6 8 その他のオプションについては、BMC アドレス指定のセクションを参照してください。

3 5 7 9 インストールディスクドライブへのパスを設定します (例: **/dev/disk/by-id/wwn-0x64cd98f04fde100024684cf3034da5c2**)。

2. クラスター設定を保存するディレクトリーを作成します。

```
$ mkdir ~/clusterconfigs
$ cp install-config.yaml ~/clusterconfigs
```

3. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源がオフになっていることを確認します。

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

4. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これを削除します。

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

8.3.6.2. install-config.yaml ファイル内でのプロキシー設定 (オプション)

プロキシーを使用して OpenShift Container Platform クラスターをデプロイするには、 **install-config.yaml** ファイルに以下の変更を加えます。

```
apiVersion: v1
baseDomain: <domain>
proxy:
  httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
  httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
  noProxy: <WILDCARD_OF_DOMAIN>,<PROVISIONING_NETWORK/CIDR>,
<BMC_ADDRESS_RANGE/CIDR>
```

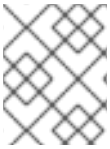
以下は、値を含む **noProxy** の例です。

```
noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

プロキシーを有効な状態にして、対応するキー/値のペアでプロキシーの適切な値を設定します。

主な留意事項:

- プロキシーに HTTPS プロキシーがない場合、 **httpsProxy** の値を **https://** から **http://** に変更します。
- provisioning ネットワークを使用する場合、これを **noProxy** 設定に含めます。そうしない場合、インストーラーは失敗します。
- プロビジョナーノード内の環境変数としてすべてのプロキシー設定を設定します。たとえば、**HTTP_PROXY**、**HTTPS_PROXY**、および **NO_PROXY** が含まれます。



注記

IPv6 でプロビジョニングする場合、**noProxy** 設定で CIDR アドレスブロックを定義することはできません。各アドレスを個別に定義する必要があります。

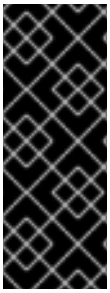
8.3.6.3. provisioning ネットワークがない install-config.yaml ファイルの変更 (オプション)

provisioning ネットワークなしに OpenShift Container Platformmake をデプロイするには、 **install-config.yaml** ファイルに以下の変更を加えます。

```
platform:
  baremetal:
    apiVIP: <apiVIP>
```

```
ingressVIP: <ingress/wildcard VIP>
provisioningNetwork: "Disabled" 1
```

1 **provisioningNetwork** 設定を追加して、必要な場合はこれを **Disabled** に設定します。



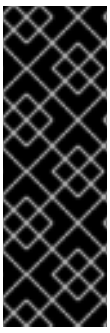
重要

PXE ブートには **provisioning** ネットワークが必要です。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。詳細は、BMC addressing for HPE iLO セクションの Redfish virtual media for HPE iLO、または BMC addressing for Dell iDRAC セクションの Redfish virtual media for Dell iDRAC セクションを参照してください。

8.3.6.4. dual-stack ネットワーク用の install-config.yaml ファイルの変更 (オプション)

デュアルスタックネットワークを仕様して OpenShift Container Platform クラスターをデプロイするには、**install-config.yaml** ファイルで **machineNetwork**、**clusterNetwork**、および **serviceNetwork** 設定を編集します。それぞれの設定には、それぞれ 2 つの CIDR エントリーが必要です。最初の CIDR エントリーは IPv4 設定で、2 つ目の CIDR エントリーは IPv6 設定であることを確認します。

```
machineNetwork:
- cidr: {{ extcidrnet }}
- cidr: {{ extcidrnet6 }}
clusterNetwork:
- cidr: 10.128.0.0/14
  hostPrefix: 23
- cidr: fd02::/48
  hostPrefix: 64
serviceNetwork:
- 172.30.0.0/16
- fd03::/112
```



重要

デュアルスタックネットワークを使用する場合は、API VIP アドレスおよび Ingress VIP アドレスはプライマリー IP アドレスファミリーである必要があります。現在、Red Hat は、IPv6 をプライマリー IP アドレスファミリーとして使用するデュアルスタック VIP またはデュアルスタックネットワークをサポートしていません。ただし、Red Hat は、IPv4 をプライマリー IP アドレスファミリーとして使用するデュアルスタックネットワークをサポートしています。したがって、IPv4 エントリーは、IPv6 エントリーの **前** になければなりません。

8.3.6.5. install-config.yaml ファイルでの管理対象 Secure Boot の設定 (オプション)

redfish、**redfish-virtualmedia**、**idrac-virtualmedia** などの Redfish BMC アドレスを使用して、インストーラーでプロビジョニングされたクラスターをデプロイする際に管理対象 Secure Boot を有効にすることができます。管理対象 Secure Boot を有効にするには、**bootMode** 設定を各ノードに追加します。

例

```
hosts:
  - name: openshift-master-0
```

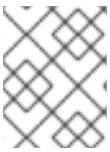
```
  role: master
  bmc:
    address: redfish://<out_of_band_ip> 1
    username: <user>
    password: <password>
  bootMACAddress: <NIC1_mac_address>
  rootDeviceHints:
   deviceName: "/dev/sda"
  bootMode: UEFISecureBoot 2
```

1 **bmc.address** 設定は、**redfish**、**redfish-virtualmedia**、または **idrac-virtualmedia** をプロトコルとして使用することを確認します。詳細は、BMC addressing for HPE iLO または BMC addressing for Dell iDRAC を参照してください。

2 **bootMode** 設定は、デフォルトで **UEFI** となります。これを **UEFISecureBoot** に変更して、管理対象 Secure Boot を有効にします。

注記

ノードが管理対象 Secure Boot をサポートできるようにするには、前提条件のノードの設定を参照してください。ノードが管理対象 Secure Boot に対応していない場合には、ノードの設定セクションの手動での Secure Boot のノードの設定を参照してください。Secure Boot を手動で設定するには、Redfish 仮想メディアが必要です。



注記

IPMI は Secure Boot 管理機能を提供しないため、Red Hat では IPMI による Secure Boot のサポートはありません。

8.3.6.6. 追加の install-config パラメーター

install-config.yaml ファイルに必要なパラメーター **hosts** パラメーターおよび **bmc** パラメーターについては、以下の表を参照してください。

表8.3 必須パラメーター

パラメーター	デフォルト	説明
baseDomain		クラスターのドメイン名。例: **example.com**
bootMode	**UEFI**	ノードのブートモード。オプションは、**legacy**、**UEFI**、および **UEFISecureBoot** です。**bootMode** が設定されていない場合は、ノードを検査する間に Ironic がこれを設定します。

パラメーター	デフォルト	説明
sshKey		**sshKey** 設定には、コントロールプレーンノードおよびワーカーノードにアクセスするために必要な **~/.ssh/id_rsa.pub** ファイルのキーが含まれます。通常、このキーは **provisionaer** ノードのキーです。
pullSecret		**pullSecret** 設定には、プロビジョナーノードの準備の際に Install OpenShift on Bare Metal ページからダウンロードしたプルシークレットのコピーが含まれます。
metadata: name:		OpenShift Container Platform クラスターに指定される名前。例: **openshift**
networking: machineCIDR:		外部ネットワークの公開 CIDR (Classless Inter-Domain Routing)。例: **10.0.0.0/24**
compute: - name: worker		OpenShift Container Platform クラスターでは、ノードがゼロであってもワーカー (またはコンピュート) ノードの名前を指定する必要があります。
compute: replicas: 2		レプリカは、OpenShift Container Platform クラスターのワーカー (またはコンピュート) ノードの数を設定します。
controlPlane: name: master		OpenShift Container Platform クラスターには、コントロールプレーン (マスター) ノードの名前が必要です。
controlPlane: replicas: 3		レプリカは、OpenShift Container Platform クラスターの一部として含まれるコントロールプレーン (マスター) ノードの数を設定します。

パラメーター	デフォルト	説明
provisioningNetworkInterface		**provisioning** ネットワークに接続されたノード上のネットワークインターフェイス名。OpenShift Container Platform 4.9 以降のリリースのために、NIC の名前を識別するために **provisioningNetworkInterface** 設定を使用する代わりに、Ironic が NIC の IP アドレスを識別できるように **bootMACAddress** 設定を使用します。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
apiVIP		(オプション) Kubernetes API 通信の仮想 IP アドレス。 この設定は、Machine Network からの予約済み IP として **install-config.yaml** ファイルで提供するか、デフォルト名が正しく解決されるように DNS で事前設定する必要があります。**install-config.yaml** ファイルの **apiVIP** 設定設定に値を追加するときは、FQDN ではなく仮想 IP アドレスを使用してください。デュアルスタックネットワークを使用する場合には、IP アドレスはプライマリー IPv4 ネットワークからのものである必要があります。設定されていない場合、インストーラーは **api.<cluster_name>.<base_domain>** を使用して DNS から IP アドレスを取得します。
disableCertificateVerification	**False**	**redfish** および **redfish-virtualmedia** では、このパラメーターで BMC アドレスを管理する必要があります。BMC アドレスに自己署名証明書を使用する場合は、値が **True** である必要があります。

パラメーター	デフォルト	説明
ingressVIP		(オプション) Ingress トラフィックの仮想 IP アドレス。 この設定は、Machine Network からの予約済み IP として **install-config.yaml** ファイルで提供するか、デフォルト名が正しく解決されるように DNS で事前設定する必要があります。**install-config.yaml** ファイルの **ingressVIP** 設定設定に値を追加するときは、FQDN ではなく仮想 IP アドレスを使用してください。デュアルスタックネットワークを使用する場合には、IP アドレスはプライマリー IPv4 ネットワークからのものである必要があります。設定されていない場合、インストーラーは **test.apps.<cluster_name>.<base_domain>** を使用して DNS から IP アドレスを取得します。

表8.4 オプションのパラメーター

パラメーター	デフォルト	説明
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	**provisioning** ネットワークでノードの IP 範囲を定義します。
provisioningNetworkCIDR	**172.22.0.0/24**	プロビジョニングに使用するネットワークの CIDR。このオプションは、**provisioning** ネットワークでデフォルトのアドレス範囲を使用しない場合に必要です。
clusterProvisioningIP	**provisioningNetworkCIDR** の 3 番目の IP アドレス。	プロビジョニングサービスが実行されるクラスター内の IP アドレス。デフォルトは、**provisioning** サブネットの 3 番目の IP アドレスに設定されます。例: **172.22.0.3**.
bootstrapProvisioningIP	**provisioningNetworkCIDR** の 2 番目の IP アドレス	インストーラーがコントロールプレーン (マスター) ノードをデプロイしている間にプロビジョニングサービスが実行されるブートストラップ仮想マシンの IP アドレス。デフォルトは、**provisioning** サブネットの 2 番目の IP アドレスに設定されます。例: **172.22.0.2** または **2620:52:0:1307::2**
externalBridge	**baremetal**	**baremetal** ネットワークに接続されているハイパーバイザーの **baremetal** ブリッジの名前。

パラメーター	デフォルト	説明
provisioningBridge	**provisioning**	**provisioning** ネットワークに接続されている **provisioner** ホストの **provisioning** ブリッジの名前。
defaultMachinePlatform		プラットフォーム設定なしでマシンプールに使用されるデフォルト設定。
bootstrapOSImage		ブートストラップノードのデフォルトのオペレーティングシステムイメージを上書きするための URL。URL にはイメージの SHA-256 ハッシュが含まれている必要があります。例: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>**
clusterOSImage		クラスターノードのデフォルトオペレーティングシステムを上書きするための URL。URL には、イメージの SHA-256 ハッシュを含める必要があります。例: **https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>**
provisioningNetwork		**provisioningNetwork** 設定は、クラスターが **provisioning** ネットワークを使用するかどうかを決定します。存在する場合、設定はクラスターがネットワークを管理するかどうかも決定します。 **Disabled**: **provisioning** ネットワークの要件を無効にするには、このパラメーターを **Disabled** に設定します。**Disabled** に設定すると、仮想メディアベースのプロビジョニングのみを使用するか、または支援付きインストーラーを使用してクラスターを起動する必要があります。**Disabled** にして、電源管理を使用する場合、BMC は **baremetal** ネットワークからアクセスできる必要があります。**Disabled** の場合は、プロビジョニングサービスに使用される **baremetal** ネットワークで 2 つの IP アドレスを指定する必要があります。 **Managed**: DHCP、TFTP などを含むプロビジョニングネットワークを完全に管理するには、このパラメーターを **Managed**(デフォルト) に設定します。 **Unmanaged**: このパラメーターを **Unmanaged** に設定してプロビジョニングネットワークを引き続き有効にしますが、DHCP の手動設定を行います。仮想メディアのプロビジョニングが推奨されますが、必要に応じて PXE を引き続き利用できます。
httpProxy		このパラメーターを、環境内で使用する適切な HTTP プロキシーに設定します。
httpsProxy		このパラメーターを、環境内で使用する適切な HTTPS プロキシーに設定します。

パラメーター	デフォルト	説明
noProxy		このパラメーターを、環境内のプロキシーの使用に対する例外の一覧に設定します。

ホスト

hosts パラメーターは、クラスターのビルドに使用される個別のベアメタルアセットの一覧です。

表8.5 ホスト

名前	デフォルト	詳細
name		詳細情報に関連付ける **BareMetalHost** リソースの名前。例: **openshift-master-0**
role		ベアメタルノードのロール。**master** または **worker** のいずれか。
bmc		ベースボード管理コントローラーの接続詳細。詳細は、BMC アドレスのセクションを参照してください。
bootMACAddress		ホストが **provisioning** ネットワークに使用する NIC の MAC アドレス。Ironic は、**bootMACAddress** 設定を使用して IP アドレスを取得します。次に、ホストにバインドします。 **注記** **provisioning** ネットワークを無効にした場合は、ホストから有効な MAC アドレスを提供する必要があります。

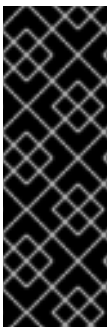
8.3.6.7. BMC アドレス指定

ほとんどのベンダーは、Intelligent Platform Management Interface(IPMI) でベースボード管理コントローラー (BMC) アドレスに対応しています。IPMI は通信を暗号化しません。これは、セキュリティーが保護された管理ネットワークまたは専用の管理ネットワークを介したデータセンター内での使用に適しています。ベンダーを確認して、Redfish ネットワークブートをサポートしているかどうかを確認します。Redfish は、コンバージド、ハイブリッド IT および Software Defined Data Center (SDDC) 向け

のシンプルでセキュアな管理を行います。Redfish は人による判読が可能、かつ機械対応が可能であり、インターネットや Web サービスの標準を活用して、最新のツールチェーンに情報を直接公開します。ハードウェアが Redfish ネットワークブートに対応していない場合には、IPMI を使用します。

IPMI
IPMI を使用するホストは **ipmi://<out-of-band-ip>:<port>** アドレス形式を使用します。これは、指定がない場合はポート **623** にデフォルトで設定されます。以下の例は、**install-config.yaml** ファイル内の IPMI 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
```



重要

BMC アドレス指定に IPMI を使用して PXE ブートする場合は、**provisioning** ネットワークが必要です。**provisioning** ネットワークなしでは、PXE ブートホストを行うことはできません。**provisioning** ネットワークなしでデプロイする場合、**redfish-virtualmedia** や **idrac-virtualmedia** などの仮想メディア BMC アドレス指定オプションを使用する必要があります。詳細は、BMC addressing for HPE iLO セクションの Redfish virtual media for HPE iLO、または BMC addressing for Dell iDRAC セクションの Redfish virtual media for Dell iDRAC セクションを参照してください。

Redfish ネットワークブート
Redfish を有効にするには、**redfish://** または **redfish+http://** を使用して TLS を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
```

```
      bmc:
        address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
        username: <user>
        password: <password>
        disableCertificateVerification: True
```

8.3.6.8. Dell iDRAC の BMC アドレス指定

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

```
platform:
  baremetal:
    hosts:
      - name: <hostname>
        role: <master | worker>
        bmc:
          address: <address> 1
          username: <user>
          password: <password>
```

1 **address** 設定はプロトコルを指定します。

Dell ハードウェアの場合、Red Hat は統合 Dell Remote Access Controller (iDRAC) 仮想メディア、Redfish ネットワークブート、および IPMI をサポートします。

表8.6 Dell iDRAC の BMC アドレス形式

プロトコル	アドレスのフォーマット
iDRAC 仮想メディア	**idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1**
Redfish ネットワークブート	**redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1**
IPMI	**ipmi://<out-of-band-ip>**

重要

idrac-virtualmedia を Redfish 仮想メディアのプロトコルとして使用します。**redfish-virtualmedia** は Dell ハードウェアでは機能しません。Dell の **idrac-virtualmedia** は、Dell の OEM 拡張機能が含まれる Redfish 標準を使用します。

詳細は、以下のセクションを参照してください。

Dell iDRAC の Redfish 仮想メディア

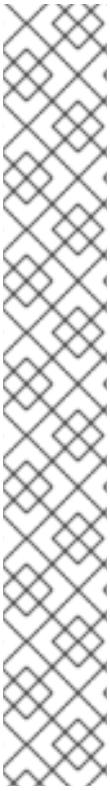
Dell サーバーの RedFish 仮想メディアについては、 **address** 設定で **idrac-virtualmedia://** を使用します。**redfish-virtualmedia://** を使用しても機能しません。

以下の例は、**install-config.yaml** ファイル内で iDRAC 仮想メディアを使用する方法を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



注記

現時点で Redfish は、ベアメタルデプロイメント上のインストーラーでプロビジョニングされるインストール向け iDRAC ファームウェアのバージョン **4.20.20.20** から **04.40.00.00** のある Dell でのみサポートされます。バージョン **04.40.00.00** には既知の問題があります。iDRAC 9 ファームウェアバージョン **04.40.00.00** では、仮想コンソールプラグインがデフォルトで **eHTML5** になり、InsertVirtualMedia ワークフローで問題が発生します。この問題を回避するには、プラグインを **HTML5** に設定します。メニューパスは以下の通りです。**Configuration → Virtual console → Plug-in Type → HTML5**

OpenShift Container Platform クラスターノードについて、iDRAC コンソールで AutoAttach が有効にされていることを確認します。メニューパスは以下のようになります。**Configuration → Virtual Media → Attach Mode → AutoAttach**

idrac-virtualmedia:// を Redfish 仮想メディアのプロトコルとして使用します。**redfish-virtualmedia://** の使用は、 **idrac-virtualmedia://** プロトコルが **idrac** ハードウェアタイプおよび Ironic の Redfish プロトコルに対応しているため、Dell ハードウェアでは機能しません。Dell の **idrac-virtualmedia://** プロトコルは、Dell の OEM 拡張機能が含まれる Redfish 標準を使用します。Ironic は、WSMAN プロトコルのある **idrac** タイプもサポートします。したがって、Dell ハードウェア上の仮想メディアで Redfish を使用する際に予期しない動作を回避するために、**idrac-virtualmedia://** を指定する必要があります。

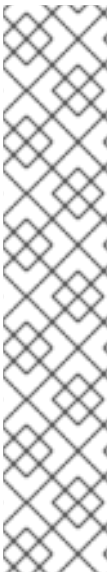
iDRAC の Redfish ネットワークブート

RedFish を有効にするには、**redfish://** または **redfish+http://** を使用してトランスポート層セキュリティー (TLS) を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



注記

現時点で Redfish は、ベアメタルデプロイメント上のインストーラーでプロビジョニングされるインストール向け iDRAC ファームウェアのバージョン **4.20.20.20** から **04.40.00.00** の Dell ハードウェアでのみサポートされます。バージョン **04.40.00.00** には既知の問題があります。iDRAC 9 ファームウェアバージョン **04.40.00.00** では、仮想コンソールプラグインがデフォルトで **eHTML5** になり、**InsertVirtualMedia** ワークフローで問題が発生します。この問題を回避するには、プラグインを **HTML5** に設定します。メニューパスは以下の通りです。**Configuration → Virtual console → Plug-in Type → HTML5**

OpenShift Container Platform クラスターノードについて、iDRAC コンソールで AutoAttach が有効にされていることを確認します。メニューパスは以下のようになります。**Configuration → Virtual Media → Attach Mode → AutoAttach**

redfish:// URL プロトコルは、Ironic の **redfish** ハードウェアタイプに対応します。

8.3.6.9. HPE iLO の BMC アドレス指定

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

```
platform:
```

```
baremetal:
  hosts:
    - name: <hostname>
      role: <master | worker>
      bmc:
        address: <address> 1
        username: <user>
        password: <password>
```

1 **address** 設定はプロトコルを指定します。

HPE integrated Lights Out (iLO) の場合、Red Hat は Redfish 仮想メディア、Redfish ネットワークブート、および IPMI をサポートします。

表8.7 HPE iLO の BMC アドレス形式

プロトコル	アドレスのフォーマット
Redfish 仮想メディア	**redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1**
Redfish ネットワークブート	**redfish://<out-of-band-ip>/redfish/v1/Systems/1**
IPMI	**ipmi://<out-of-band-ip>**

詳細は、以下のセクションを参照してください。

HPE iLO の Redfish 仮想メディア

HPE サーバーについて RedFish Virtual Media を有効にするには、**address** 設定で **redfish-virtualmedia://** を使用します。以下の例は、**install-config.yaml** ファイル内で Redfish 仮想メディアを使用する方法を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
```

```
      role: master
      bmc:
        address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
        username: <user>
        password: <password>
        disableCertificateVerification: True
```

注記

Ironic は、仮想メディアで iLO4 をサポートしないので、Redfish 仮想メディアは、iLO4 を実行する第 9 世代のシステムではサポートされません。

HPE iLO の Redfish ネットワークブート

Redfish を有効にするには、**redfish://** または **redfish+http://** を使用して TLS を無効にします。インストーラーには、ホスト名または IP アドレスとシステム ID へのパスの両方が必要です。以下の例は、**install-config.yaml** ファイル内の RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

アウトバウンド管理のアドレスについて認証局の証明書を使用することが推奨されますが、自己署名証明書を使用する場合には、**bmc** 設定に **disableCertificateVerification: True** を含める必要があります。以下の例は、**install-config.yaml** ファイル内の **disableCertificateVerification: True** 設定パラメーターを使用する RedFish 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

8.3.6.10. Fujitsu iRMC の BMC アドレス指定

それぞれの **bmc** エントリーの **address** フィールドは、URL スキーム内のコントローラーのタイプやネットワーク上のその場所を含む、OpenShift Container Platform クラスターノードに接続する URL です。

```
platform:
  baremetal:
    hosts:
      - name: <hostname>
```

```
      role: <master | worker>
      bmc:
        address: <address> 1
        username: <user>
        password: <password>
```

1 **address** 設定はプロトコルを指定します。

Fujitsu ハードウェアの場合、Red Hat は、統合 Remote Management Controller (iRMC) および IPMI をサポートします。

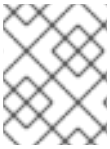
表8.8 Fujitsu iRMC の BMC アドレス形式

プロトコル	アドレスのフォーマット
iRMC	**irmc://<out-of-band-ip>**
IPMI	**ipmi://<out-of-band-ip>**

iRMC

Fujitsu ノードは **irmc://<out-of-band-ip>** を使用し、デフォルトではポート **443** に設定されます。以下の例は、**install-config.yaml** ファイル内の iRMC 設定を示しています。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: irmc://<out-of-band-ip>
          username: <user>
          password: <password>
```



注記

現在 Fujitsu は、ベアメタルへのインストーラーでプロビジョニングされるインストール用に iRMC S5 ファームウェアバージョン 3.05P 以降をサポートしています。

8.3.6.11. ルートデバイスのヒント

rootDeviceHints パラメーターは、インストーラーが Red Hat Enterprise Linux CoreOS (RHCOS) イメージを特定のデバイスにプロビジョニングできるようにします。インストーラーは、検出順にデバイスを検査し、検出された値をヒントの値と比較します。インストーラーは、ヒント値に一致する最初に検出されたデバイスを使用します。この設定は複数のヒントを組み合わせることができますが、デバイスは、インストーラーがこれを選択できるようにすべてのヒントに一致する必要があります。

表8.9 サブフィールド

サブフィールド	説明
deviceName	**/dev/vda** などの Linux デバイス名を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
hctl	**0:0:0:0** などの SCSI バスアドレスを含む文字列。ヒントは、実際の値と完全に一致する必要があります。
model	ベンダー固有のデバイス識別子を含む文字列。ヒントは、実際の値のサブ文字列になります。
vendor	デバイスのベンダーまたは製造元の名前が含まれる文字列。ヒントは、実際の値のサブ文字列になります。
serialNumber	デバイスのシリアル番号を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
minSizeGigabytes	デバイスの最小サイズ (ギガバイト単位) を表す整数。
wwn	一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnWithExtension	ベンダー拡張が追加された一意のストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
wwnVendorExtension	一意のベンダーストレージ ID を含む文字列。ヒントは、実際の値と完全に一致する必要があります。
rotational	デバイスがローテーションするディスクである (true) か、そうでないか (false) を示すブール値。

使用例

```
    - name: master-0
      role: master
      bmc:
        address: ipmi://10.10.0.3:6203
        username: admin
        password: redhat
      bootMACAddress: de:ad:be:ef:00:40
      rootDeviceHints:
        deviceName: "/dev/sda"
```

8.3.6.12. OpenShift Container Platform マニフェストの作成

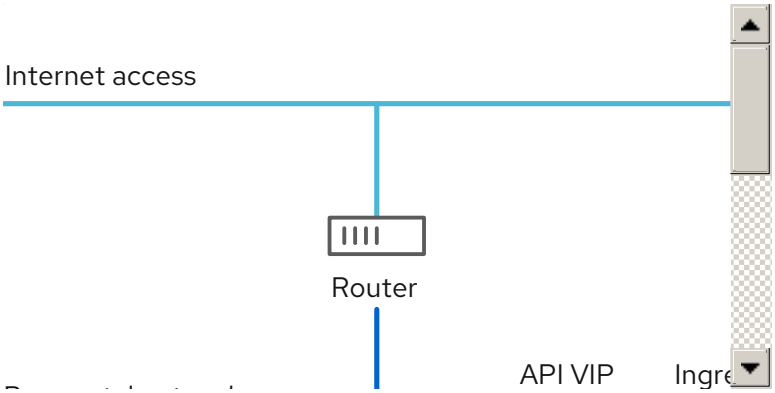
1. OpenShift Container Platform マニフェストを作成します。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory
because its dependencies are dirty and it needs to be regenerated
```

8.3.6.13. 非接続クラスター向けの NTP 設定 (オプション)

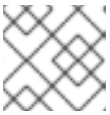
OpenShift Container Platform は、クラスターノードに **chrony** Network Time Protocol (NTP) サービスをインストールします。



OpenShift Container Platform ノードは、正しく実行するために日時について合意する必要があります。ワーカーノードがコントロールプレーンノードの NTP サーバーから日付と時刻を取得すると、ルーティング可能なネットワークに接続されていないクラスターのインストールおよび操作が有効になるため、上位の stratum の NTP サーバーにアクセスできなくなります。

手順

1. コントロールプレーンノードの **chrony.conf** ファイルのコンテンツを含む Butane 設定 (**99-master-chrony-conf-override.bu**) を作成します。



注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

Butane 設定例

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-master-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
```

```
    contents:
      inline: |
        # Use public servers from the pool.ntp.org project.
        # Please consider joining the pool (https://www.pool.ntp.org/join.html).

        # The Machine Config Operator manages this file
        server openshift-master-0.<cluster-name>.<domain> iburst 1
        server openshift-master-1.<cluster-name>.<domain> iburst
        server openshift-master-2.<cluster-name>.<domain> iburst

        stratumweight 0
        driftfile /var/lib/chrony/drift
        rtcsync
        makestep 10 3
        bindcmdaddress 127.0.0.1
        bindcmdaddress ::1
        keyfile /etc/chrony.keys
        commandkey 1
        generatecommandkey
        noclientlog
        logchange 0.5
        logdir /var/log/chrony

        # Configure the control plane nodes to serve as local NTP servers
        # for all worker nodes, even if they are not in sync with an
        # upstream NTP server.

        # Allow NTP client access from the local network.
        allow all
        # Serve time even if not synchronized to a time source.
        local stratum 3 orphan
```

1 **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

2. Butane を使用して、コントロールプレーンノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-master-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-master-chrony-conf-override.bu -o 99-master-chrony-conf-override.yaml
```

3. コントロールプレーンノードの NTP サーバーを参照するワーカーノードの **chrony.conf** ファイルのコンテンツを含む Butane 設定 (**99-worker-chrony-conf-override.bu**) を作成します。

Butane 設定例

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-worker-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
```

```
- path: /etc/chrony.conf
  mode: 0644
  overwrite: true
  contents:
    inline: |
      # The Machine Config Operator manages this file.
      server openshift-master-0.<cluster-name>.<domain> iburst 1
      server openshift-master-1.<cluster-name>.<domain> iburst
      server openshift-master-2.<cluster-name>.<domain> iburst

      stratumweight 0
      driftfile /var/lib/chrony/drift
      rtcsync
      makestep 10 3
      bindcmdaddress 127.0.0.1
      bindcmdaddress ::1
      keyfile /etc/chrony.keys
      commandkey 1
      generatecommandkey
      noclientlog
      logchange 0.5
      logdir /var/log/chrony
```

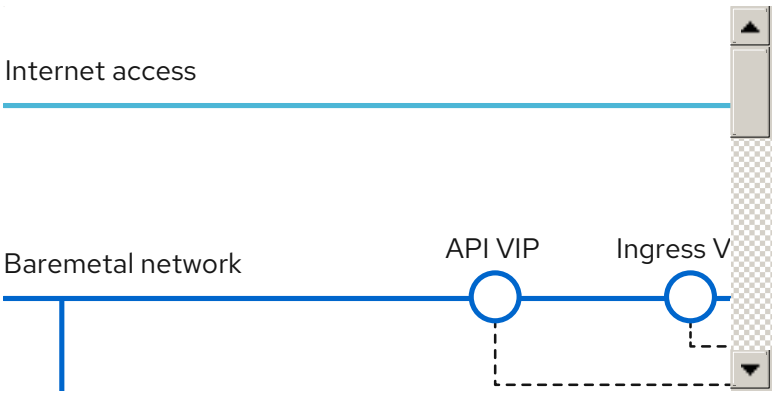
1 **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

4. Butane を使用して、ワーカーノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-worker-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-worker-chrony-conf-override.bu -o 99-worker-chrony-conf-override.yaml
```

8.3.6.14. (オプション) コントロールプレーンで実行されるネットワークコンポーネントの設定

コントロールプレーンノードでのみ実行されるようにネットワークコンポーネントを設定します。デフォルトで、OpenShift Container Platform はマシン設定プールの任意のノードが **ingressVIP** 仮想 IP アドレスをホストできるようにします。ただし、多くの環境は、コントロールプレーンノードとは異なるサブネットにワーカーノードをデプロイします。リモートワーカーを別々のサブネットにデプロイする場合は、コントロールプレーンノード専用に **ingressVIP** 仮想 IP アドレスを配置する必要があります。



手順

1. **install-config.yaml** ファイルを保存するディレクトリーに移動します。

```
$ cd ~/clusterconfigs
```

2. **manifests** サブディレクトリーに切り替えます。

```
$ cd manifests
```

3. **cluster-network-avoid-workers-99-config.yaml** という名前のファイルを作成します。

```
$ touch cluster-network-avoid-workers-99-config.yaml
```

4. エディターで **cluster-network-avoid-workers-99-config.yaml** ファイルを開き、Operator 設定を記述するカスタムリソース (CR) を入力します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 50-worker-fix-ipi-rwn
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/kubernetes/manifests/keepalived.yaml
          mode: 0644
          contents:
            source: data:,
```

このマニフェストは、 **ingressVIP** 仮想 IP アドレスをコントロールプレーンノードに配置します。また、このマニフェストは、コントロールプレーンノードにのみ以下のプロセスをデプロイします。

- **openshift-ingress-operator**
- **keepalived**

5. **cluster-network-avoid-workers-99-config.yaml** ファイルを保存します。

6. **manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルを作成します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/master: ""
```

7. **manifests** ディレクトリーのバックアップの作成を検討してください。インストーラーは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

8. **cluster-scheduler-02-config.yml** マニフェストを変更し、**mastersSchedulable** フィールドを **true** に設定して、コントロールプレーンノードをスケジュール対象にします。デフォルトでは、コントロールプレーンノードはスケジュール対象ではありません。以下に例を示します。

```
$ sed -i "s;mastersSchedulable: false;mastersSchedulable: true;g"
clusterconfigs/manifests/cluster-scheduler-02-config.yml
```

注記

この手順の完了後にコントロールプレーンノードをスケジュールできない場合には、クラスターのデプロイに失敗します。

8.3.7. 非接続レジストリーの作成 (オプション)

インストールレジストリーのローカルコピーを使用して OpenShift Container Platform クラスターをインストールする必要がある場合があります。これにより、クラスターノードがインターネットにアクセスできないネットワーク上にあるため、ネットワークの効率が高まる可能性があります。

レジストリーのローカルまたはミラーリングされたコピーには、以下が必要です。

- レジストリーの証明書。これには、自己署名証明書を使用できます。
- システム上のコンテナーが提供する Web サーバー。
- 証明書およびローカルリポジトリーの情報が含まれる更新されたプルシークレット。

注記

レジストリーノードでの非接続レジストリーの作成はオプションです。以下のセクションでは、それらの手順はレジストリーノードで非接続レジストリーを作成する場合にのみ実行する必要があるためにオプションであることを示しています。レジストリーノードで非接続レジストリーを作成する際に、(optional) というラベルが付けられたすべての後続のサブセクションを実行する必要があります。

8.3.7.1. ミラーリングされたレジストリーをホストするためのレジストリーノードの準備 (オプション)

レジストリーノードに以下の変更を加えます。

手順

1. レジストリーノードでファイアウォールポートを開きます。

```
$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt  --permanent
$ sudo firewall-cmd --add-port=5000/tcp --zone=public   --permanent
$ sudo firewall-cmd --reload
```

2. レジストリーノードに必要なパッケージをインストールします。

```
$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. リポジトリー情報が保持されるディレクトリー構造を作成します。

```
$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

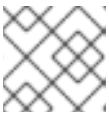
8.3.7.2. 自己署名証明書の生成 (オプション)

レジストリーノードの自己署名証明書を生成し、これを **/opt/registry/certs** ディレクトリーに配置します。

手順

1. 必要に応じて証明書情報を調整します。

```
$ host_fqdn=$( hostname --long )
$ cert_c="<Country Name>"   # Country Name (C, 2 letter code)
$ cert_s="<State>"          # Certificate State (S)
$ cert_l="<Locality>"       # Certificate Locality (L)
$ cert_o="<Organization>"   # Certificate Organization (O)
$ cert_ou="<Org Unit>"      # Certificate Organizational Unit (OU)
$ cert_cn="${host_fqdn}"    # Certificate Common Name (CN)

$ openssl req \
    -newkey rsa:4096 \
    -nodes \
    -sha256 \
    -keyout /opt/registry/certs/domain.key \
    -x509 \
    -days 365 \
    -out /opt/registry/certs/domain.crt \
    -addext "subjectAltName = DNS:${host_fqdn}" \
    -subj "/C=${cert_c}/ST=${cert_s}/L=${cert_l}/O=${cert_o}/OU=${cert_ou}/CN=${cert_cn}"
```



注記

<Country Name> を置き換える場合には、2 文字のみを使用します。例: **US**

2. レジストリーノードの **ca-trust** を新規証明書で更新します。

```
$ sudo cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

8.3.7.3. レジストリー podman コンテナーの作成 (オプション)

レジストリーコンテナーは、証明書、認証ファイルの **/opt/registry** ディレクトリーを使用し、そのデータファイルを保存するためにこれを使用します。

レジストリーコンテナーは **httpd** を使用し、認証に **htpasswd** ファイルが必要になります。

手順

1. コンテナーが使用する **htpasswd** ファイルを **/opt/registry/auth** に作成します。

```
$ htpasswd -bBc /opt/registry/auth/htpasswd <user> <passwd>
```

-

<user> をユーザー名に、**<passwd>** をパスワードに置き換えます。

2. レジストリーコンテナーを作成し、起動します。

```
$ podman create \
  --name ocpdiscon-registry \
  -p 5000:5000 \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" \
  -e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
  -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
  -e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
  -e "REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true" \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/auth:/auth:z \
  -v /opt/registry/certs:/certs:z \
  docker.io/library/registry:2
```

```
$ podman start ocpdiscon-registry
```

8.3.7.4. pull-secret のコピーおよび更新 (オプション)

プロビジョナーノードからレジストリーノードにプルシークレットファイルをコピーし、これを新規レジストリーノードの認証情報を含めるように変更します。

手順

1. **pull-secret.txt** ファイルをコピーします。

```
$ scp kni@provisioner:/home/kni/pull-secret.txt pull-secret.txt
```

2. **host_fqdn** 環境変数をレジストリーノードの完全修飾ドメイン名で更新します。

```
$ host_fqdn=$( hostname --long )
```

3. **htpasswd** ファイルの作成に使用される **http** 認証情報の base64 エンコーディングで **b64auth** 環境変数を更新します。

```
$ b64auth=$( echo -n '<username>:<passwd>' | openssl base64 )
```

<username> をユーザー名に、**<passwd>** をパスワードに置き換えます。

4. **base64** 承認文字列を使用するように **AUTHSTRING** 環境変数を設定します。**$USER** 変数は、現行ユーザーの名前が含まれる環境変数です。

```
$ AUTHSTRING="{\"$host_fqdn:5000\": {\"auth\": \"$b64auth\",\"email\":
\"$USER@redhat.com\"}}"
```

5. **pull-secret.txt** ファイルを更新します。

```
$ jq ".auths += $AUTHSTRING" < pull-secret.txt > pull-secret-update.txt
```

8.3.7.5. リポジトリーのミラーリング (オプション)

手順

1. **oc** バイナリーをプロビジョナーノードからレジストリーノードにコピーします。

```
$ sudo scp kni@provisioner:/usr/local/bin/oc /usr/local/bin
```

2. 必要な環境変数を設定します。
   a. リリースバージョンを設定します。

```
$ VERSION=<release_version>
```

   <release_version> には、インストールする OpenShift Container Platform のバージョンに対応するタグ (**4.8** など) を指定します。

   b. ローカルレジストリー名とホストポートを設定します。

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

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

   c. ローカルリポジトリー名を設定します。

```
$ LOCAL_REPO='<local_repository_name>'
```

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

3. リモートインストールイメージをローカルリポジトリーにミラーリングします。

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.txt \
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

8.3.7.6. install-config.yaml ファイルを、非接続レジストリーを使用するように変更します (オプション)。

プロビジョナーノードでは、**install-config.yaml** ファイルは **pull-secret-update.txt** ファイルから新たに作成された pull-secret を使用する必要があります。**install-config.yaml** ファイルには、非接続レジストリーノードの証明書およびレジストリー情報も含まれる必要があります。

手順

1. 非接続レジストリーノードの証明書を **install-config.yaml** ファイルに追加します。証明書は **"additionalTrustBundle: |"** 行に従い、通常は 2 つのスペースで適切にインデントされる必要があります。

```
$ echo "additionalTrustBundle: |" >> install-config.yaml
$ sed -e 's/^/  /' /opt/registry/certs/domain.crt >> install-config.yaml
```

2. レジストリーのミラー情報を **install-config.yaml** ファイルに追加します。

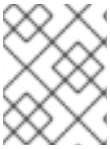
```
$ echo "imageContentSources:" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev" >> install-config.yaml
```

注記

registry.example.com をレジストリーの完全修飾ドメイン名に置き換えます。

8.3.8. ワーカーノードでのルーターのデプロイ

インストール時に、インストーラーはルーター Pod をワーカーノードにデプロイします。デフォルトで、インストーラーは 2 つのルーター Pod をインストールします。初期クラスターにワーカーノードが 1 つしかない場合や、デプロイされたクラスターで、OpenShift Container Platform クラスター内のサービスに対して予定される外部トラフィックを処理する追加のルーターが必要な場合、**yaml** ファイルを作成して適切なルーターレプリカ数を設定できます。



注記

デフォルトで、インストーラーは 2 つのルーターをデプロイします。クラスターにワーカーノードが 2 つ以上ある場合、このセクションを省略できます。

注記

クラスターにワーカーノードがない場合、インストーラーはデフォルトで 2 つのルーターをコントロールプレーンノードにデプロイします。クラスターにワーカーノードがない場合、このセクションを省略できます。

手順

1. **router-replicas.yaml** ファイルを作成します。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: <num-of-router-pods>
  endpointPublishingStrategy:
    type: HostNetwork
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/worker: ""
```

注記

<num-of-router-pods> を適切な値に置き換えます。1 つのワーカーノードのみを使用している場合、**replicas:** を **1** に設定します。4 つ以上のワーカーノードを使用している場合、**replicas:** のデフォルトの値 **2** を随時増やすことができます。

2. **router-replicas.yaml** ファイルを保存し、これを **clusterconfigs/openshift** ディレクトリーにコピーします。

```
cp ~/router-replicas.yaml clusterconfigs/openshift/99_router-replicas.yaml
```

8.3.9. インストールの検証チェックリスト

❑ OpenShift Container Platform インストーラーが取得されている。

❑ OpenShift Container Platform インストーラーが展開されている。

❑ **install-config.yaml** の必須パラメーターが設定されている。

❑ **install-config.yaml** の **hosts** パラメーターが設定されている。

❑ **install-config.yaml** の **bmc** パラメーターが設定されている。

❑ **bmc address** フィールドで設定されている値の変換が適用されている。

❑ 非接続レジストリーを作成している (オプション)。

❑ (オプション) 非接続レジストリー設定が使用されている場合にこれを検証する。

❑ (オプション) ルーターをワーカーノードにデプロイしている。

8.3.10. OpenShift Container Platform インストーラーを使用したクラスターのデプロイ

OpenShift Container Platform インストーラーを実行します。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

8.3.11. インストール後

デプロイメントプロセスで、**tail** コマンドを install ディレクトリーフォルダーの **.openshift_install.log** ログファイルに対して実行して、インストールの全体のステータスを確認できます。

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

8.3.12. 静的 IP アドレス設定の検証

クラスターノードの DHCP 予約で無限リースが指定されている場合、インストーラーがノードを正常にプロビジョニングした後に、dispatcher スクリプトはノードのネットワーク設定をチェックします。ネットワーク設定に無限 DHCP リースが含まれているとスクリプトが判断すると、DHCP リースの IP アドレスを静的 IP アドレスとして使用して新規接続を作成します。

注記

dispatcher スクリプトは、クラスター内の他のノードのプロビジョニングの進行中に、正常にプロビジョニングされたノードで実行される場合があります。

ネットワーク設定が正しく機能していることを確認します。

手順

1. ノードのネットワークインターフェイス設定を確認してください。
2. DHCP サーバーをオフにし、OpenShift Container Platform ノードを再起動して、ネットワーク設定が適切に機能していることを確認します。

8.3.13. ベアメタルにクラスターを再インストールする準備

ベアメタルにクラスターを再インストールする前に、クリーンアップ操作を実行する必要があります。

手順

1. ブートストラップ、コントロールプレーンノード (マスターとも呼ばれる)、およびワーカーノードのディスクを削除または再フォーマットします。ハイパーバイザー環境で作業している場合は、削除したディスクを追加する必要があります。
2. 以前のインストールで生成されたアーティファクトを削除します。

```
$ cd ; /bin/rm -rf auth/ bootstrap.ign master.ign worker.ign metadata.json \
.openshift_install.log .openshift_install_state.json
```

3. 新しいマニフェストと Ignition 設定ファイルを生成します。詳細は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。
4. インストールプログラムが作成した新規ブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これにより、以前の Ignition ファイルが上書きされます。

関連情報

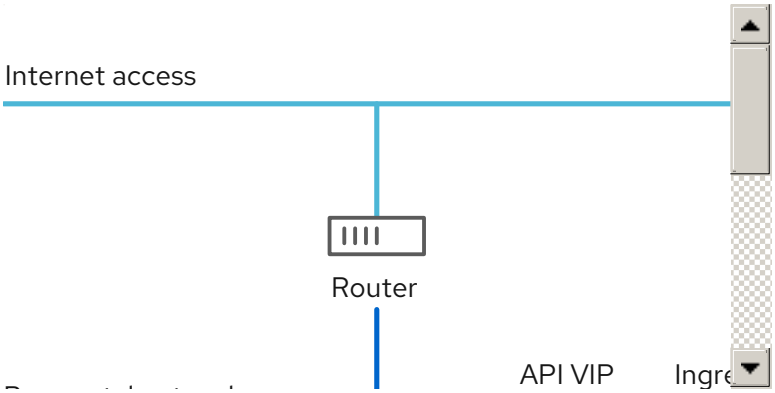
- 異なるリリースチャネルの概要については、OpenShift Container Platform アップグレードチャネルおよびリリース について参照してください。

8.4. インストーラーでプロビジョニングされるインストール後の設定

インストーラーでプロビジョニングされるクラスターを正常にデプロイしたら、以下のインストール後の手順を考慮してください。

8.4.1. 非接続クラスター向けの NTP 設定 (オプション)

OpenShift Container Platform は、クラスターノードに **chrony** Network Time Protocol (NTP) サービスをインストールします。以下の手順を使用して、コントロールプレーンノードで NTP サーバーを設定し、デプロイメントが正常に完了した後にコントロールプレーンノードの NTP クライアントとしてワーカーノードを設定します。



OpenShift Container Platform ノードは、正しく実行するために日時について合意する必要があります。ワーカーノードがコントロールプレーンノードの NTP サーバーから日付と時刻を取得すると、ルーティング可能なネットワークに接続されていないクラスターのインストールおよび操作が有効になるため、上位の stratum の NTP サーバーにアクセスできなくなります。

手順

1. コントロールプレーンノードの **chrony.conf** ファイルのコンテンツを含む Butane 設定 (**99-master-chrony-conf-override.bu**) を作成します。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

Butane 設定例

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-master-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: master
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # Use public servers from the pool.ntp.org project.
          # Please consider joining the pool (https://www.pool.ntp.org/join.html).

          # The Machine Config Operator manages this file
          server openshift-master-0.<cluster-name>.<domain> iburst (1)
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
```

```
      keyfile /etc/chrony.keys
      commandkey 1
      generatecommandkey
      noclientlog
      logchange 0.5
      logdir /var/log/chrony

      # Configure the control plane nodes to serve as local NTP servers
      # for all worker nodes, even if they are not in sync with an
      # upstream NTP server.

      # Allow NTP client access from the local network.
      allow all
      # Serve time even if not synchronized to a time source.
      local stratum 3 orphan
```

1 **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

2. Butane を使用して、コントロールプレーンノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-master-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-master-chrony-conf-override.bu -o 99-master-chrony-conf-override.yaml
```

3. コントロールプレーンノードの NTP サーバーを参照するワーカーノードの **chrony.conf** ファイルのコンテンツを含む Butane 設定 (**99-worker-chrony-conf-override.bu**) を作成します。

Butane 設定例

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-worker-chrony-conf-override
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          # The Machine Config Operator manages this file.
          server openshift-master-0.<cluster-name>.<domain> iburst 1
          server openshift-master-1.<cluster-name>.<domain> iburst
          server openshift-master-2.<cluster-name>.<domain> iburst

          stratumweight 0
          driftfile /var/lib/chrony/drift
          rtcsync
          makestep 10 3
          bindcmdaddress 127.0.0.1
          bindcmdaddress ::1
```

```
    keyfile /etc/chrony.keys
    commandkey 1
    generatecommandkey
    noclientlog
    logchange 0.5
    logdir /var/log/chrony
```

1 **<cluster-name>** はクラスターの名前に置き換え、**<domain>** は完全修飾ドメイン名に置き換える必要があります。

4. Butane を使用して、ワーカーノードに配信される設定が含まれる **MachineConfig** オブジェクトファイル (**99-worker-chrony-conf-override.yaml**) を生成します。

```
$ butane 99-worker-chrony-conf-override.bu -o 99-worker-chrony-conf-override.yaml
```

5. **99-master-chrony-conf-override.yaml** ポリシーをコントロールプレーンノードに適用します。

```
$ oc apply -f 99-master-chrony-conf-override.yaml
```

出力例

```
machineconfig.machineconfiguration.openshift.io/99-master-chrony-conf-override created
```

6. **99-worker-chrony-conf-override.yaml** ポリシーをワーカーノードに適用します。

```
$ oc apply -f 99-worker-chrony-conf-override.yaml
```

出力例

```
machineconfig.machineconfiguration.openshift.io/99-worker-chrony-conf-override created
```

7. 適用された NTP 設定のステータスを確認します。

```
$ oc describe machineconfigpool
```

8.4.2. インストール後のプロビジョニングネットワークの有効化

ベアメタルクラスター用の支援付きインストーラーおよびインストーラーでプロビジョニングされるインストールは、**provisioning** ネットワークなしでクラスターをデプロイする機能を提供します。この機能は、概念実証クラスターや、各ノードのベースボード管理コントローラーが **baremetal** ネットワークを介してルーティング可能な場合に Redfish 仮想メディアのみを使用してデプロイするなどのシナリオ向けです。

OpenShift Container Platform 4.8 以降では、Cluster Baremetal Operator (CBO) を使用してインストール後に **provisioning** ネットワークを有効にすることができます。

前提条件

- すべてのワーカーノードおよびコントロールプレーンノードに接続されている専用の物理ネットワークが存在する必要があります。

- ネイティブのタグなしの物理ネットワークを分離する必要があります。
- **provisioningNetwork** 設定が **Managed** に設定されている場合、ネットワークには DHCP サーバーを含めることはできません。
- OpenShift Container Platform 4.9 の **provisioningInterface** 設定を省略し、**bootMACAddress** 設定を使用できます。

手順

1. **provisioningInterface** 設定を設定する場合、まずクラスターノードのプロビジョニングインターフェイス名を特定します。たとえば、**eth0** または **eno1** などです。
2. クラスターノードの **provisioning** ネットワークインターフェイスで Preboot eXecution Environment (PXE) を有効にします。
3. **provisioning** ネットワークの現在の状態を取得して、これをプロビジョニングカスタムリソース (CR) ファイルに保存します。

```
$ oc get provisioning -o yaml > enable-provisioning-nw.yaml
```

4. プロビジョニング CR ファイルを変更します。

```
$ vim ~/enable-provisioning-nw.yaml
```

provisioningNetwork 設定までスクロールダウンして、これを **Disabled** から **Managed** に変更します。次に、**provisioningNetwork** 設定の後に、**provisioningOSDownloadURL**、**provisioningIP**、**provisioningNetworkCIDR**、**provisioningDHCPRange**、**provisioningInterface**、および **watchAllNameSpaces** 設定を追加します。各設定に適切な値を指定します。

```
apiVersion: v1
items:
- apiVersion: metal3.io/v1alpha1
  kind: Provisioning
  metadata:
    name: provisioning-configuration
  spec:
    provisioningNetwork: 1
    provisioningOSDownloadURL: 2
    provisioningIP: 3
    provisioningNetworkCIDR: 4
    provisioningDHCPRange: 5
    provisioningInterface: 6
    watchAllNameSpaces: 7
```

1. **provisioningNetwork** は、**Managed**、**Unmanaged**、または **Disabled** のいずれかになります。**Managed** に設定すると、Metal3 はプロビジョニングネットワークを管理し、CBO は設定済みの DHCP サーバーで Metal3 Pod をデプロイします。**Unmanaged** に設定すると、システム管理者は DHCP サーバーを手動で設定します。
2. **provisioningOSDownloadURL** は有効な sha256 チェックサムを持つ有効な HTTPS URL で、Metal3 Pod が **.qcow2.gz** または **.qcow2.xz** で終わる qcow2 オペレーティングシステムイメージをダウンロードできるようにします。このフィールドは、プロビジョニング

ネットワークが **Managed**、**Unmanaged**、または **Disabled** のいずれの場合でも必要です。例: **http://192.168.0.1/images/rhcos-<version>.x86_64.qcow2.gz?sha256=<sha>**.

3 **provisioningIP** は、DHCP サーバーおよび ironic がネットワークのプロビジョニングために使用する静的 IP アドレスです。この静的 IP アドレスは、DHCP 範囲外の **provisioning** 内でなければなりません。この設定を設定する場合は、**provisioning** ネットワークが **Disabled** の場合でも、有効な IP アドレスが必要です。静的 IP アドレスは metal3 Pod にバインドされます。metal3 Pod が失敗し、別のサーバーに移動する場合、静的 IP アドレスも新しいサーバーに移動します。

4 Classless Inter-Domain Routing (CIDR) アドレス。この設定を設定する場合は、**provisioning** ネットワークが **Disabled** の場合でも、有効な CIDR アドレスが必要です。例: **192.168.0.1/24**

5 DHCP の範囲。この設定は、 **Managed** プロビジョニングネットワークにのみ適用されます。**provisioning** ネットワークが **Disabled** の場合は、この設定を省略します。例: **192.168.0.64, 192.168.0.253**

6 クラスターノードの **provisioning** インターフェイス用の NIC 名。**provisioningInterface** 設定は、**Managed** および **Unmanaged** プロビジョニングネットワークにのみ適用されます。**provisioning** ネットワークが **Disabled** の場合に、**provisioningInterface** 設定が省略されます。代わりに **bootMACAddress** 設定を使用するように **provisioningInterface** 設定を省略します。

7 metal3 がデフォルトの **openshift-machine-api** namespace 以外の namespace を監視するようにするには、この設定を **true** に設定します。デフォルト値は **false** です。

5. 変更をプロビジョニング CR ファイルに保存します。
6. プロビジョニング CR ファイルをクラスターに適用します。

```
$ oc apply -f enable-provisioning-nw.yaml
```

8.4.3. 外部ロードバランサーの設定

OpenShift Container Platform クラスターを設定し、デフォルトのロードバランサーの代わりに外部ロードバランサーを使用することができます。

前提条件

- ロードバランサーでは、システムの任意のユーザーが TCP をポート 6443、443、および 80 が利用できる必要があります。
- それぞれのコントロールプレーンノード間で API ポート 6443 を負荷分散します。
- すべてのコンピュートノード間でアプリケーションポート 443 と 80 を負荷分散します。
- ロードバランサーでは、Ignition 起動設定をノードに提供するために使用されるポート 22623 はクラスター外に公開されません。
- ロードバランサーはクラスター内のすべてのマシンにアクセスできる必要があります。このアクセスを許可する方法には、以下が含まれます。
  - ロードバランサーをクラスターのマシンのサブネットに割り当てます。

- ロードバランサーを使用するマシンに Floating IP アドレスを割り当てます。

重要

外部の負荷分散サービスとコントロールプレーンノードは同じ L2 ネットワークで実行する必要があります。また、VLAN を使用して負荷分散サービスとコントロールプレーンノード間のトラフィックをルーティングする際に同じ VLAN で実行する必要があります。

手順

1. ポート 6443、443、および 80 でロードバランサーからクラスターへのアクセスを有効にします。
   たとえば、以下の HAProxy 設定に留意してください。

   サンプル HAProxy 設定のセクション

```
...
listen my-cluster-api-6443
    bind 0.0.0.0:6443
    mode tcp
    balance roundrobin
    server my-cluster-master-2 192.0.2.2:6443 check
    server my-cluster-master-0 192.0.2.3:6443 check
    server my-cluster-master-1 192.0.2.1:6443 check
listen my-cluster-apps-443
        bind 0.0.0.0:443
        mode tcp
        balance roundrobin
        server my-cluster-worker-0 192.0.2.6:443 check
        server my-cluster-worker-1 192.0.2.5:443 check
        server my-cluster-worker-2 192.0.2.4:443 check
listen my-cluster-apps-80
        bind 0.0.0.0:80
        mode tcp
        balance roundrobin
        server my-cluster-worker-0 192.0.2.7:80 check
        server my-cluster-worker-1 192.0.2.9:80 check
        server my-cluster-worker-2 192.0.2.8:80 check
```

2. ロードバランサーでクラスター API およびアプリケーションの DNS サーバーにレコードを追加します。以下に例を示します。

```
<load_balancer_ip_address> api.<cluster_name>.<base_domain>
<load_balancer_ip_address> apps.<cluster_name>.<base_domain>
```

3. コマンドラインで **curl** を使用して、外部ロードバランサーおよび DNS 設定が機能することを確認します。

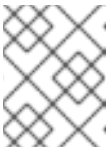
   a. クラスター API がアクセス可能であることを確認します。

```
$ curl https://<loadbalancer_ip_address>:6443/version --insecure
```

   設定が正しい場合は、応答として JSON オブジェクトを受信します。

```
{
  "major": "1",
  "minor": "11+",
  "gitVersion": "v1.11.0+ad103ed",
  "gitCommit": "ad103ed",
  "gitTreeState": "clean",
  "buildDate": "2019-01-09T06:44:10Z",
  "goVersion": "go1.10.3",
  "compiler": "gc",
  "platform": "linux/amd64"
}
```

b. クラスターアプリケーションがアクセス可能であることを確認します。



注記

Web ブラウザーで OpenShift Container Platform コンソールを開き、アプリケーションのアクセスを確認することもできます。

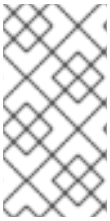
```
$ curl http://console-openshift-console.apps.<cluster_name>.<base_domain> -I -L --
insecure
```

設定が正しい場合は、HTTP 応答を受信します。

```
HTTP/1.1 302 Found
content-length: 0
location: https://console-openshift-console.apps.<cluster-name>.<base domain>/
cache-control: no-cacheHTTP/1.1 200 OK
referrer-policy: strict-origin-when-cross-origin
set-cookie: csrf-
token=39HoZgztDnzjJkq/JuLJMeoKNXlfiVv2YgZc09c3TBOBU4Nl6kDXaJH1LdicNhN1UsQ
Wzon4Dor9GWGfopaTEQ==; Path=/; Secure
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-frame-options: DENY
x-xss-protection: 1; mode=block
date: Tue, 17 Nov 2020 08:42:10 GMT
content-type: text/html; charset=utf-8
set-cookie:
1e2670d92730b515ce3a1bb65da45062=9b714eb87e93cf34853e87a92d6894be; path=/;
HttpOnly; Secure; SameSite=None
cache-control: private
```

8.5. クラスターの拡張

インストーラーでプロビジョニングされる OpenShift Container Platform クラスターのデプロイ後に、以下の手順を使用してワーカーノードの数を拡張することができます。それぞれの候補となるワーカーノードが前提条件を満たしていることを確認します。

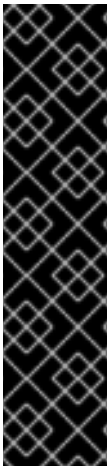


注記

RedFish Virtual Media を使用してクラスターを拡張するには、最低限のファームウェア要件を満たす必要があります。RedFish Virtual Media を使用したクラスターの拡張時についての詳細は、**前提条件**セクションの**仮想メディアを使用したインストールのファームウェア要件**を参照してください。

8.5.1. ベアメタルノードの準備

クラスターを拡張するには、DHCP サーバーが必要です。各ノードに DHCP 予約が必要です。



IP アドレスの予約し、それらを静的 IP アドレスにする

一部の管理者は、各ノードの IP アドレスが DHCP サーバーがない状態で一定になるように静的 IP アドレスの使用を選択します。OpenShift Container Platform クラスターで静的 IP アドレスを使用するには、**DHCP サーバーの IP アドレスを無限リースで予約** します。インストーラーがノードを正常にプロビジョニングした後に、dispatcher スクリプトがノードのネットワーク設定をチェックします。dispatcher スクリプトがネットワーク設定に DHCP 無限リースが含まれていることを検出すると、DHCP の無限リースの IP アドレスを使用して接続を静的 IP 接続として再作成します。DHCP 無限リースのない NIC は変更されないままになります。

IP アドレスを無限リースで設定することは、Machine Config Operator を使用してデプロイされるネットワーク設定と互換性がありません。

ベアメタルノードを準備するには、プロビジョナーノードから以下の手順を実行する必要があります。

手順

1. **oc** バイナリーを取得します (必要な場合)。これはプロビジョナーノード上にあるはずです。

```
$ curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$VERSION/openshift-client-linux-$VERSION.tar.gz | tar zxvf - oc
```

```
$ sudo cp oc /usr/local/bin
```

2. ベースボード管理コントローラーを使用してベアメタルノードの電源を切り、オフになっていることを確認します。

3. ベアメタルノードのベースボード管理コントローラーのユーザー名およびパスワードを取得します。次に、ユーザー名とパスワードから **base64** 文字列を作成します。

```
$ echo -ne "root" | base64
```

```
$ echo -ne "password" | base64
```

4. ベアメタルノードの設定ファイルを作成します。

```
$ vim bmh.yaml
```

```
---
apiVersion: v1
kind: Secret
```

```
metadata:
  name: openshift-worker-<num>-bmc-secret
type: Opaque
data:
  username: <base64-of-uid>
  password: <base64-of-pwd>
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-worker-<num>
spec:
  online: true
  bootMACAddress: <NIC1-mac-address>
  bmc:
    address: <protocol>://<bmc-ip>
    credentialsName: openshift-worker-<num>-bmc-secret
```

2 つの **name** フィールドおよび **credentialsName** フィールドのベアメタルのワーカー数の **<num>** を置き換えます。**<base64-of-uid>** を、ユーザー名の **base64** 文字列に置き換えます。**<base64-of-pwd>** を、パスワードの **base64** 文字列に置き換えます。**<NIC1-mac-address>** を、ベアメタルの最初の NIC の MAC アドレスに置き換えます。

追加の BMC 設定オプションについては、BMC アドレスのセクションを参照してください。**<protocol>** を、IPMI、 RedFish その他の BMC プロトコルに置き換えマス。 **<bmc-ip>** を、ベアメタルノードのベースボード管理コントローラー (BMC) の IP アドレスに置き換えます。

注記

既存のベアメタルノードの MAC アドレスが、プロビジョニングしようとしているベアメタルホストの MAC アドレスと一致する場合、Ironic インストールは失敗します。ホストの登録、検査、クリーニング、または他の Ironic 手順が失敗する場合、ベアメタル Operator はインストールを継続的に再試行します。詳細は、ホストが重複する MAC アドレスの診断 を参照してください。

5. ベアメタルノードを作成します。

```
$ oc -n openshift-machine-api create -f bmh.yaml
```

```
secret/openshift-worker-<num>-bmc-secret created
baremetalhost.metal3.io/openshift-worker-<num> created
```

ここで、**<num>** はワーカー数です。

6. ベアメタルノードの電源をオンにし、これを検査します。

```
$ oc -n openshift-machine-api get bmh openshift-worker-<num>
```

ここで、**<num>** はワーカーノード数です。

```
NAME                 STATUS   PROVISIONING STATUS   CONSUMER   BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-<num>   OK       ready                        ipmi://<out-of-band-ip>   unknown
```

```
true
```

8.5.2. ベアメタルコントロールプレーンノードの交換

以下の手順を使用して、インストーラーによってプロビジョニングされた OpenShift Container Platform コントロールプレーンノードを置き換えます。

重要

既存のコントロールプレーンホストから **BareMetalHost** オブジェクト定義を再利用する場合は、**externallyProvisioned** フィールドを **true** に設定したままにしないでください。

既存のコントロールプレーン **BareMetalHost** オブジェクトが、OpenShift Container Platform インストールプログラムによってプロビジョニングされた場合には、**externallyProvisioned** フラグが **true** に設定されている可能性があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- etcd のバックアップを取得している。

重要

問題が発生した場合にクラスターを復元できるように、この手順を実行する前に etcd のバックアップを作成してください。etcd バックアップの作成に関する詳細は、**関連情報** セクションを参照してください。

手順

1. Bare Metal Operator が使用可能であることを確認します。

```
$ oc get clusteroperator baremetal
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
baremetal 4.8.0     True        False         False      3d15h
```

2. 古い **BareMetalHost** オブジェクトおよび **Machine** オブジェクトを削除します。

```
$ oc delete bmh -n openshift-machine-api <host_name>
$ oc delete machine -n openshift-machine-api <machine_name>
```

<host_name> をホストの名前に、**<machine_name>** をマシンの名前に置き換えます。マシン名は **CONSUMER** フィールドの下に表示されます。

BareMetalHost オブジェクトと **Machine** オブジェクトを削除すると、マシンコントローラーは **Node** オブジェクトを自動的に削除します。

3. 新しい **BareMetalHost** オブジェクトとシークレットを作成して BMC 認証情報を保存します。

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: control-plane-<num>-bmc-secret (1)
  namespace: openshift-machine-api
data:
  username: <base64_of_uid> (2)
  password: <base64_of_pwd> (3)
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: control-plane-<num> (4)
  namespace: openshift-machine-api
spec:
  automatedCleaningMode: disabled
  bmc:
    address: <protocol>://<bmc_ip> (5)
    credentialsName: control-plane-<num>-bmc-secret (6)
  bootMACAddress: <NIC1_mac_address> (7)
  bootMode: UEFI
  externallyProvisioned: false
  hardwareProfile: unknown
  online: true
EOF
```

(1) (4) (6) **name** フィールドと **credentialsName** フィールドにあるベアメタルノードのコントロールプレーン数の **<num>** を置き換えます。

(2) **<base64_of_uid>** を、ユーザー名の **base64** 文字列に置き換えます。

(3) **<base64_of_pwd>>** を、パスワードの **base64** 文字列に置き換えます。

(5) **<protocol>** を **redfish**、**redfish-virtualmedia**、**idrac-virtualmedia** などの BMC プロトコルに置き換えます。**<bmc_ip>** を、ベアメタルノードのベースボード管理コントローラー (BMC) の IP アドレスに置き換えます。その他の BMC 設定オプションについては、**関連情報** セクションのBMC アドレス指定を参照してください。

(7) **<NIC1_mac_address>** を、ベアメタルの最初の NIC の MAC アドレスに置き換えます。

検査が完了すると、**BareMetalHost** オブジェクトが作成され、プロビジョニングできるようになります。

4. 利用可能な **BareMetalHost** オブジェクトを表示します。

```
$ oc get bmh -n openshift-machine-api
```

出力例

```
NAME                          STATE                  CONSUMER          ONLINE   ERROR   AGE
control-plane-1.example.com   available              control-plane-1   true             1h10m
control-plane-2.example.com   externally provisioned control-plane-2   true
```

```
4h53m
control-plane-3.example.com   externally provisioned   control-plane-3          true
4h53m
compute-1.example.com         provisioned              compute-1-ktmmx          true
4h53m
compute-1.example.com         provisioned              compute-2-l2zmb          true
4h53m
```

コントロールプレーンノード用の **MachineSet** オブジェクトがないため、代わりに **Machine** オブジェクトを作成する必要があります。別のコントロールプレーン **Machine** オブジェクトから **providerSpec** をコピーできます。

5. **Machine** オブジェクトを作成します。

```
$ cat <<EOF | oc apply -f -
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  annotations:
    metal3.io/BareMetalHost: openshift-machine-api/control-plane-<num> 1
  labels:
    machine.openshift.io/cluster-api-cluster: control-plane-<num> 2
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
  name: control-plane-<num> 3
  namespace: openshift-machine-api
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: baremetal.cluster.k8s.io/v1alpha1
      customDeploy:
        method: install_coreos
      hostSelector: {}
      image:
        checksum: ""
        url: ""
      kind: BareMetalMachineProviderSpec
      metadata:
        creationTimestamp: null
      userData:
        name: master-user-data-managed
EOF
```

1 2 3 **name** フィールド、**labels** フィールド、および **annotations** フィールドにあるベアメタルノードのコントロールプレーン数の **<num>** を置き換えます。

6. **BareMetalHost** オブジェクトを表示するには、次のコマンドを実行します。

```
$ oc get bmh -A
```

出力例

```
NAME                 STATE                CONSUMER             ONLINE   ERROR   AGE
```

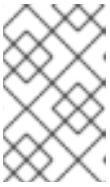
```
control-plane-1.example.com   provisioned              control-plane-1   true   2h53m
control-plane-2.example.com   externally provisioned   control-plane-2   true
5h53m
control-plane-3.example.com   externally provisioned   control-plane-3   true
5h53m
compute-1.example.com         provisioned              compute-1-ktmmx   true
5h53m
compute-2.example.com         provisioned              compute-2-l2zmb   true
5h53m
```

7. RHCOS のインストール後、**BareMetalHost** がクラスターに追加されていることを確認します。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS      ROLES     AGE   VERSION
control-plane-1.example.com   available   master    4m2s  v1.18.2
control-plane-2.example.com   available   master    141m  v1.18.2
control-plane-3.example.com   available   master    141m  v1.18.2
compute-1.example.com         available   worker    87m   v1.18.2
compute-2.example.com         available   worker    87m   v1.18.2
```



注記

新しいコントロールプレーンノードの交換後、新しいノードで実行されている etcd Pod は **crashloopback** ステータスになります。詳細は、**関連情報** セクションの正常でない etcd メンバーの置き換えを参照してください。

関連情報

- 正常でない etcd メンバーの置き換え
- etcd のバックアップ
- BMC アドレス指定

8.5.3. クラスター内の新しいホストをプロビジョニングする際の重複する MAC アドレスの診断

クラスター内の既存のベアメタルノードの MAC アドレスが、クラスターに追加しようとしているベアメタルホストの MAC アドレスと一致する場合、ベアメタル Operator はホストを既存のノードに関連付けます。ホストの登録、検査、クリーニング、または他の Ironic 手順が失敗する場合、ベアメタル Operator はインストールを継続的に再試行します。障害が発生したベアメタルホストの登録エラーが表示されます。

openshift-machine-api namespace で実行されているベアメタルホストを調べることで、重複する MAC アドレスを診断できます。

前提条件

- ベアメタルに OpenShift Container Platform クラスターをインストールします。

- OpenShift Container Platform CLI (**oc**) をインストールします。
- **cluster-admin** 権限を持つユーザーとしてログインしている。

手順

プロビジョニングに失敗したベアメタルホストが既存のノードと同じ MAC アドレスを持つかどうかを判断するには、以下を実行します。

1. **openshift-machine-api** namespace で実行されているベアメタルホストを取得します。

```
$ oc get bmh -n openshift-machine-api
```

出力例

```
NAME                 STATUS   PROVISIONING STATUS      CONSUMER
openshift-master-0   OK       externally provisioned   openshift-zpwpq-master-0
openshift-master-1   OK       externally provisioned   openshift-zpwpq-master-1
openshift-master-2   OK       externally provisioned   openshift-zpwpq-master-2
openshift-worker-0   OK       provisioned              openshift-zpwpq-worker-0-lv84n
openshift-worker-1   OK       provisioned              openshift-zpwpq-worker-0-zd8lm
openshift-worker-2   error    registering
```

2. 障害が発生したホストのステータスに関する詳細情報を表示するには、**<bare_metal_host_name>** をホストの名前に置き換えて、以下のコマンドを実行します。

```
$ oc get -n openshift-machine-api bmh <bare_metal_host_name> -o yaml
```

出力例

```
...
status:
  errorCount: 12
  errorMessage: MAC address b4:96:91:1d:7c:20 conflicts with existing node openshift-
worker-1
  errorType: registration error
...
```

8.5.4. ベアメタルノードのプロビジョニング

ベアメタルノードをプロビジョニングするには、プロビジョナーノードから以下の手順を実行する必要があります。

手順

1. ベアメタルノードをプロビジョニングする前に、**PROVISIONING STATUS** が **ready** であることを確認します。

```
$  oc -n openshift-machine-api get bmh openshift-worker-<num>
```

ここで、**<num>** はワーカーノード数です。

```
NAME                 STATUS   PROVISIONING STATUS   CONSUMER   BMC
```

```
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-<num>   OK       ready                    ipmi://<out-of-band-ip>   unknown
true
```

2. ワーカーノードの数を取得します。

```
$ oc get nodes
```

```
NAME                                STATUS   ROLES          AGE     VERSION
provisioner.openshift.example.com   Ready    master         30h     v1.16.2
openshift-master-1.openshift.example.com   Ready    master         30h     v1.16.2
openshift-master-2.openshift.example.com   Ready    master         30h     v1.16.2
openshift-master-3.openshift.example.com   Ready    master         30h     v1.16.2
openshift-worker-0.openshift.example.com   Ready    master         30h     v1.16.2
openshift-worker-1.openshift.example.com   Ready    master         30h     v1.16.2
```

3. マシンセットを取得します。

```
$ oc get machinesets -n openshift-machine-api
```

```
NAME                        DESIRED   CURRENT   READY   AVAILABLE   AGE
...
openshift-worker-0.example.com     1         1         1       1          55m
openshift-worker-1.example.com     1         1         1       1          55m
```

4. ワーカーノードの数を 1 つずつ増やします。

```
$ oc scale --replicas=<num> machineset <machineset> -n openshift-machine-api
```

<num> を、ワーカーノードの新たな数に置き換えます。**<machineset>** を、直前の手順で設定されたマシン名に置き換えます。

5. ベアメタルノードのステータスを確認します。

```
$ oc -n openshift-machine-api get bmh openshift-worker-<num>
```

ここで、**<num>** はワーカーノード数です。ステータスは **ready** から **provisioning** に変わります。

```
NAME                 STATUS   PROVISIONING STATUS   CONSUMER                BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-<num>   OK       provisioning         openshift-worker-<num>-65tjz
ipmi://<out-of-band-ip>   unknown          true
```

provisioning ステータスは、OpenShift Container Platform クラスターがノードをプロビジョニングするまでそのままになります。この場合、30 分以上の時間がかかる場合があります。ノードのプロビジョニング後に、ステータスは **provisioned** に変わります。

```
NAME                 STATUS   PROVISIONING STATUS   CONSUMER                BMC
HARDWARE PROFILE   ONLINE   ERROR
openshift-worker-<num>   OK       provisioned          openshift-worker-<num>-65tjz
ipmi://<out-of-band-ip>   unknown          true
```

6. プロビジョニングが完了したら、ベアメタルノードが準備状態であることを確認します。

```
$ oc get nodes
```

```
NAME                                        STATUS   ROLES   AGE     VERSION
provisioner.openshift.example.com           Ready    master  30h     v1.16.2
openshift-master-1.openshift.example.com    Ready    master  30h     v1.16.2
openshift-master-2.openshift.example.com    Ready    master  30h     v1.16.2
openshift-master-3.openshift.example.com    Ready    master  30h     v1.16.2
openshift-worker-0.openshift.example.com    Ready    master  30h     v1.16.2
openshift-worker-1.openshift.example.com    Ready    master  30h     v1.16.2
openshift-worker-<num>.openshift.example.com  Ready    worker  3m27s   v1.16.2
```

kubelet を確認することもできます。

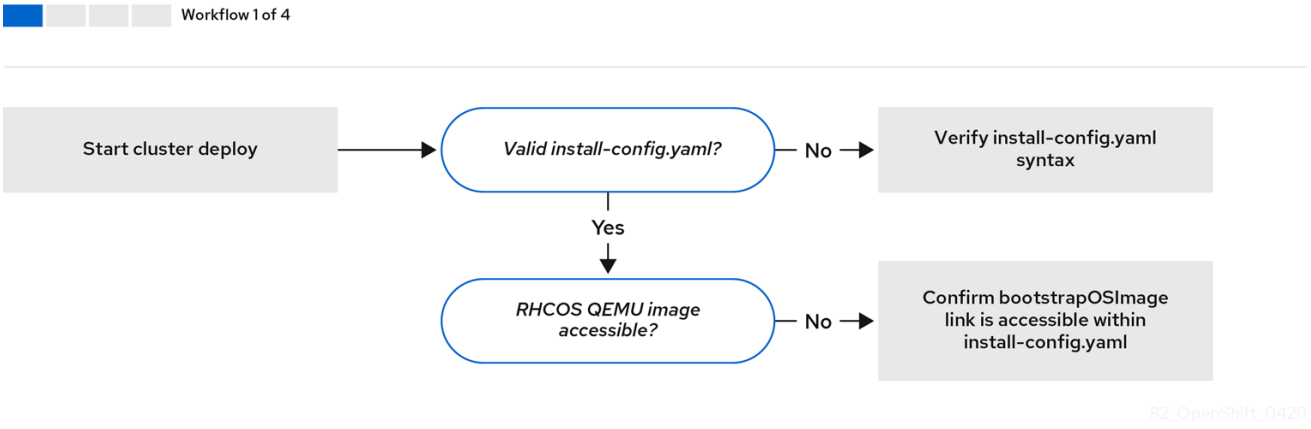
```
$ ssh openshift-worker-<num>
```

```
[kni@openshift-worker-<num>]$ journalctl -fu kubelet
```

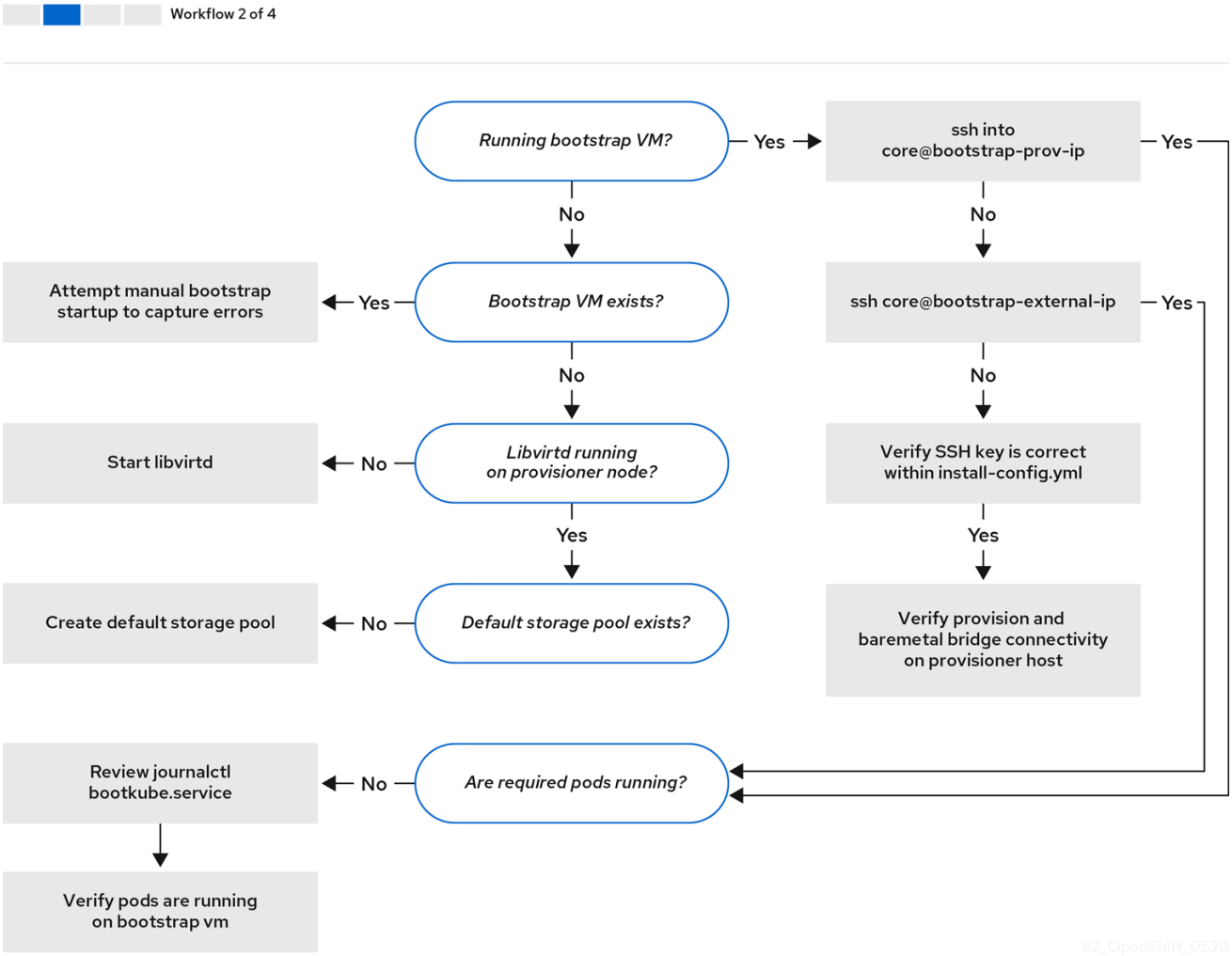
8.6. トラブルシューティング

8.6.1. インストーラーワークフローのトラブルシューティング

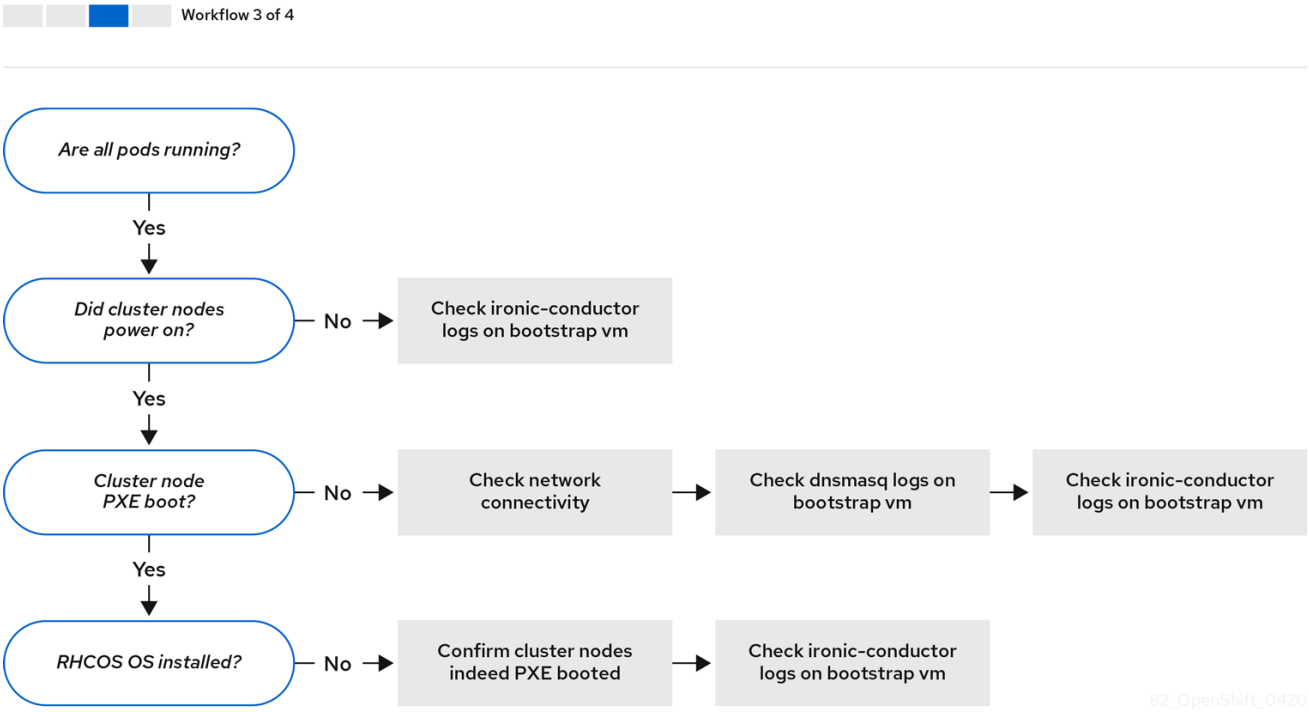
インストール環境のトラブルシューティングを行う前に、ベアメタルへのインストーラーでプロビジョニングされるインストールの全体的なフローを理解することは重要です。以下の図は、環境におけるステップごとのトラブルシューティングフローを示しています。



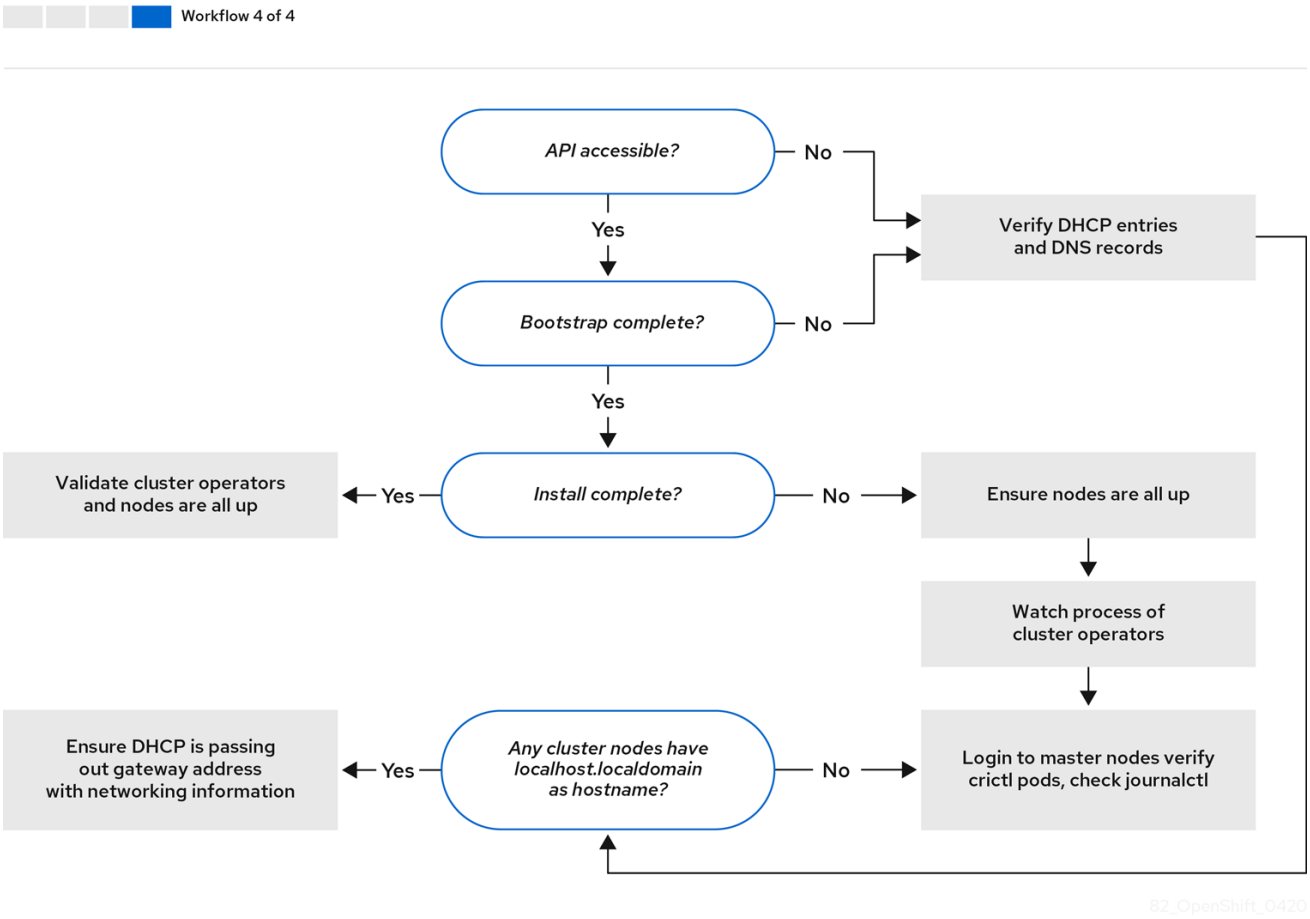
ワークフロー 1/4 は、**install-config.yaml** ファイルにエラーがある場合や Red Hat Enterprise Linux CoreOS (RHCOS) イメージにアクセスできない場合のトラブルシューティングのワークフローを説明しています。トラブルシューティングについての提案は、Troubleshooting **install-config.yaml** を参照してください。



ワークフロー 2/4 は、ブートストラップ仮想マシンの問題、クラスターノードを起動できないブートストラップ仮想マシン、および ログの検査 についてのトラブルシューティングのワークフローを説明しています。**provisioning** ネットワークなしに OpenShift Container Platform クラスターをインストールする場合は、このワークフローは適用されません。



ワークフロー 3/4 は、PXE ブートしないクラスターノード のトラブルシューティングのワークフローを説明しています。RedFish 仮想メディアを使用してインストールする場合、各ノードは、インストーラーがノードをデプロイするために必要な最小ファームウェア要件を満たしている必要があります。詳細は、**前提条件**セクションの**仮想メディアを使用したインストールのファームウェア要件**を参照してください。



ワークフロー 4/4 は、アクセスできない API から 検証済みのインストール までのトラブルシューティングのワークフローを説明します。

8.6.2. install-config.yaml のトラブルシューティング

install-config.yaml 設定ファイルは、OpenShift Container Platform クラスターの一部であるすべてのノードを表します。このファイルには、**apiVersion**、**baseDomain**、**imageContentSources**、および仮想 IP アドレスのみで設定されるがこれらに制限されない必要なオプションが含まれます。OpenShift Container Platform クラスターのデプロイメントの初期段階でエラーが発生した場合、エラーは **install-config.yaml** 設定ファイルにある可能性があります。

手順

1. YAML-tips のガイドラインを使用します。
2. syntax-check を使用して YAML 構文が正しいことを確認します。
3. Red Hat Enterprise Linux CoreOS (RHCOS) QEMU イメージが適切に定義され、**install-config.yaml** で提供される URL 経由でアクセスできることを確認します。以下に例を示します。

```
$ curl -s -o /dev/null -I -w "%{http_code}\n" http://webserver.example.com:8080/rhcos-44.81.202004250133-0-qemu.x86_64.qcow2.gz?sha256=7d884b46ee54fe87bbc3893bf2aa99af3b2d31f2e19ab5529c60636fbd0f1ce7
```

出力が **200** の場合、ブートストラップ仮想マシンイメージを保存する Web サーバーからの有効な応答があります。

8.6.3. ブートストラップ仮想マシンの問題

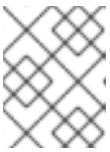
OpenShift Container Platform インストールプログラムは、OpenShift Container Platform クラスターノードのプロビジョニングを処理するブートストラップノードの仮想マシンを起動します。

手順

1. インストールプログラムをトリガー後の約 10 分から 15 分後に、**virsh** コマンドを使用してブートストラップ仮想マシンが機能していることを確認します。

```
$ sudo virsh list
```

```
 Id    Name                           State
----------------------------------------------------
 12    openshift-xf6fq-bootstrap      running
```



注記

ブートストラップ仮想マシンの名前は常にクラスター名で始まり、その後にランダムな文字セットが続き、bootstrap という単語で終わります。

ブートストラップ仮想マシンが 10 - 15 分後に実行されていない場合は、実行されない理由についてトラブルシューティングします。発生する可能性のある問題には以下が含まれます。

2. **libvirtd** がシステムで実行されていることを確認します。

```
$ systemctl status libvirtd
```

```
● libvirtd.service - Virtualization daemon
  Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2020-03-03 21:21:07 UTC; 3 weeks 5 days ago
    Docs: man:libvirtd(8)
          https://libvirt.org
Main PID: 9850 (libvirtd)
   Tasks: 20 (limit: 32768)
  Memory: 74.8M
  CGroup: /system.slice/libvirtd.service
          ├─ 9850 /usr/sbin/libvirtd
```

ブートストラップ仮想マシンが動作している場合は、これにログインします。

3. **virsh console** コマンドを使用して、ブートストラップ仮想マシンの IP アドレスを見つけます。

```
$ sudo virsh console example.com
```

```
Connected to domain example.com
Escape character is ^]
Red Hat Enterprise Linux CoreOS 43.81.202001142154.0 (Ootpa) 4.3
SSH host key: SHA256:BRWJktXZgQQRY5zjuAV0IKZ4WM7i4TiUyMVanqu9Pqg (ED25519)
SSH host key: SHA256:7+iKGA7VtG5szmk2jB5gl/5EZ+SNcJ3a2g23o0lnIio (ECDSA)
SSH host key: SHA256:DH5VWhvhvagOTaLsYiVNse9ca+ZSW/30OOMed8rIGOc (RSA)
ens3:  fd35:919d:4042:2:c7ed:9a9f:a9ec:7
ens4: 172.22.0.2 fe80::1d05:e52e:be5d:263f
localhost login:
```

重要

provisioning ネットワークなしで OpenShift Container Platform クラスターをデプロイする場合、**172.22.0.2** などのプライベート IP アドレスではなく、パブリック IP アドレスを使用する必要があります。

4. IP アドレスを取得したら、**ssh** コマンドを使用してブートストラップ仮想マシンにログインします。

注記

直前の手順のコンソール出力では、**ens3** で提供される IPv6 IP アドレスまたは **ens4** で提供される IPv4 IP を使用できます。

```
$ ssh core@172.22.0.2
```

ブートストラップ仮想マシンへのログインに成功しない場合は、以下いずれかのシナリオが発生した可能性があります。

- **172.22.0.0/24** ネットワークにアクセスできない。プロビジョナーと **provisioning** ネットワークブリッジ間のネットワーク接続を確認します。この問題は、**provisioning** ネットワークを使用している場合に発生することがあります。

- パブリックネットワーク経由でブートストラップ仮想マシンにアクセスできない。**baremetal** ネットワークで SSH を試行する際に、**provisioner** ホストの、とくに **baremetal** ネットワークブリッジについて接続を確認します。

- **Permission denied (publickey,password,keyboard-interactive)** が出される。ブートストラップ仮想マシンへのアクセスを試行すると、**Permission denied** エラーが発生する可能性があります。仮想マシンへのログインを試行するユーザーの SSH キーが **install-config.yaml** ファイル内で設定されていることを確認します。

8.6.3.1. ブートストラップ仮想マシンがクラスターノードを起動できない

デプロイメント時に、ブートストラップ仮想マシンがクラスターノードの起動に失敗する可能性があり、これにより、仮想マシンがノードに RHCOS イメージをプロビジョニングできなくなります。このシナリオは、以下の原因で発生する可能性があります。

- **install-config.yaml** ファイルに関連する問題。

- ベアメタルネットワークを使用してアウトオブバンド (out-of-band) ネットワークアクセスに関する問題

この問題を確認するには、**ironic** に関連する 3 つのコンテナーを使用できます。

- **ironic-api**

- **ironic-conductor**

- **ironic-inspector**

手順

1. ブートストラップ仮想マシンにログインします。

```
$ ssh core@172.22.0.2
```

2. コンテナーログを確認するには、以下を実行します。

```
[core@localhost ~]$ sudo podman logs -f <container-name>
```

<container-name> を、**ironic-api**、**ironic-conductor**、または **ironic-inspector** のいずれかに置き換えます。コントロールプレーンノードが PXE 経由で起動しない問題が発生した場合には、**ironic-conductor** Pod を確認してください。**ironic-conductor** Pod には、IPMI 経由でノードへのログインを試みるため、クラスターノードのブートの試行についての最も詳細な情報が含まれます。

考えられる理由

クラスターノードは、デプロイメントの開始時に **ON** 状態にある可能性があります。

解決策

IPMI でのインストールを開始する前に、OpenShift Container Platform クラスターノードの電源をオフにします。

```
$ ipmitool -I lanplus -U root -P <password> -H <out-of-band-ip> power off
```

8.6.3.2. ログの検査

RHCOS イメージのダウンロードまたはアクセスに問題が発生した場合には、最初に **install-config.yaml** 設定ファイルで URL が正しいことを確認します。

RHCOS イメージをホストする内部 Web サーバーの例

```
bootstrapOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-qemu.x86_64.qcow2.gz?sha256=9d999f55ff1d44f7ed7c106508e5deecd04dc3c06095d34d36bf1cd127837e0c
clusterOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-openstack.x86_64.qcow2.gz?sha256=a1bda656fa0892f7b936fdc6b6a6086bddaed5dafacedcd7a1e811abb78fe3b0
```

ipa-downloader および **coreos-downloader** コンテナーは、**install-config.yaml** 設定ファイルで指定されている Web サーバーまたは外部の quay.io レジストリーからリソースをダウンロードします。以下の 2 つのコンテナーが稼働していることを確認し、必要に応じてログを検査します。

- **ipa-downloader**
- **coreos-downloader**

手順

1. ブートストラップ仮想マシンにログインします。

   ```
   $ ssh core@172.22.0.2
   ```

2. ブートストラップ仮想マシン内の **ipa-downloader** および **coreos-downloader** コンテナーのステータスを確認します。

   ```
   [core@localhost ~]$ sudo podman logs -f ipa-downloader
   ```

   ```
   [core@localhost ~]$ sudo podman logs -f coreos-downloader
   ```

   ブートストラップ仮想マシンがイメージへの URL にアクセスできない場合、**curl** コマンドを使用して、仮想マシンがイメージにアクセスできることを確認します。

3. すべてのコンテナーがデプロイメントフェーズで起動されているかどうかを示す **bootkube** ログを検査するには、以下を実行します。

   ```
   [core@localhost ~]$ journalctl -xe
   ```

   ```
   [core@localhost ~]$ journalctl -b -f -u bootkube.service
   ```

4. **dnsmasq**、**mariadb**、**httpd**、および **ironic** を含むすべての Pod が実行中であることを確認します。

   ```
   [core@localhost ~]$ sudo podman ps
   ```

5. Pod に問題がある場合には、問題のあるコンテナーのログを確認します。**ironic-api** のログを確認するには、以下を実行します。

   ```
   [core@localhost ~]$ sudo podman logs <ironic-api>
   ```

8.6.4. クラスターノードが PXE ブートしない

OpenShift Container Platform クラスターノードが PXE ブートしない場合、PXE ブートしないクラスターノードで以下のチェックを実行します。この手順は、**provisioning** ネットワークなしで OpenShift Container Platform クラスターをインストールする場合には適用されません。

手順

1. **provisioning** ネットワークへのネットワークの接続を確認します。
2. PXE が **provisioning** ネットワークの NIC で有効にされており、PXE がその他のすべての NIC について無効にされていることを確認します。
3. **install-config.yaml** 設定ファイルに、適切なハードウェアプロファイルと **provisioning** ネットワークに接続された NIC のブート MAC アドレスが含まれることを確認します。以下に例を示します。

コントロールプレーンノードの設定

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: default          #control plane node settings
```

ワーカーノード設定

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: unknown          #worker node settings
```

8.6.5. API にアクセスできない

クラスターが実行されており、クライアントが API にアクセスできない場合、ドメイン名の解決の問題により API へのアクセスが妨げられる可能性があります。

手順

1. **Hostname Resolution:** クラスターノードに **localhost.localdomain** だけでなく、完全修飾ドメイン名があることを確認します。以下に例を示します。

```
$ hostname
```

ホスト名が設定されていない場合、正しいホスト名を設定します。以下に例を示します。

```
$ hostnamectl set-hostname <hostname>
```

2. **正しくない名前の解決:** 各ノードに **dig** および **nslookup** を使用して DNS サーバーに正しい名前の解決があることを確認します。以下に例を示します。

```
$ dig api.<cluster-name>.example.com
```

```
; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <<>> api.<cluster-name>.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37551
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 866929d2f8e8563582af23f05ec44203d313e50948d43f60 (good)
;; QUESTION SECTION:
;api.<cluster-name>.example.com. IN A

;; ANSWER SECTION:
api.<cluster-name>.example.com. 10800 IN A 10.19.13.86

;; AUTHORITY SECTION:
<cluster-name>.example.com. 10800 IN NS <cluster-name>.example.com.

;; ADDITIONAL SECTION:
<cluster-name>.example.com. 10800 IN A 10.19.14.247

;; Query time: 0 msec
;; SERVER: 10.19.14.247#53(10.19.14.247)
;; WHEN: Tue May 19 20:30:59 UTC 2020
;; MSG SIZE  rcvd: 140
```

前述の例の出力は、**api.<cluster-name>.example.com** VIP の適切な IP アドレスが **10.19.13.86** であることを示しています。この IP アドレスは **baremetal** 上にある必要があります。

8.6.6. 以前のインストールのクリーンアップ

以前のデプロイメントに失敗した場合、OpenShift Container Platform のデプロイを再試行する前に、失敗した試行からアーティファクトを削除します。

手順

1. OpenShift Container Platform クラスターをインストールする前に、すべてのベアメタルノードの電源をオフにします。

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

2. 以前に試行したデプロイメントにより古いブートストラップリソースが残っている場合は、これらをすべて削除します。

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

3. 以下を **clusterconfigs** ディレクトリーから削除し、Terraform が失敗することを防ぎます。

```
$ rm -rf ~/clusterconfigs/auth ~/clusterconfigs/terraform* ~/clusterconfigs/tls
~/clusterconfigs/metadata.json
```

8.6.7. レジストリーの作成に関する問題

非接続レジストリーの作成時に、レジストリーのミラーリングを試行する際に User Not Authorized エラーが発生する場合があります。このエラーは、新規の認証を既存の **pull-secret.txt** ファイルに追加できない場合に生じる可能性があります。

手順

1. 認証が正常に行われていることを確認します。

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

注記

インストールイメージのミラーリングに使用される変数の出力例:

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

RELEASE_IMAGE および **VERSION** の値は、**OpenShift インストールの環境のセットアップ**セクションの **OpenShift Installer の取得** の手順で設定されています。

2. レジストリーのミラーリング後に、非接続環境でこれにアクセスできることを確認します。

```
$ curl -k -u <user>:<password> https://registry.example.com:<registry-port>/v2/_catalog
{"repositories":["<Repo-Name>"]}
```

8.6.8. その他の問題点

8.6.8.1. runtime network not ready エラーへの対応

クラスターのデプロイメント後に、以下のエラーが発生する可能性があります。

```
`runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network
plugin returns error: Missing CNI default network`
```

Cluster Network Operator は、インストーラーによって作成される特別なオブジェクトに対応してネットワークコンポーネントをデプロイします。これは、コントロールプレーン (マスター) ノードが起動した後、ブートストラップコントロールプレーンが停止する前にインストールプロセスの初期段階で実行されます。これは、コントロールプレーン (マスター) ノードの起動の長い遅延や **apiserver** 通信の問題などの、より判別しづらいインストーラーの問題を示すことができます。

手順

1. **openshift-network-operator** namespace の Pod を検査します。

```
$ oc get all -n openshift-network-operator
```

```
NAME                                READY STATUS            RESTARTS   AGE
pod/network-operator-69dfd7b577-bg89v   0/1   ContainerCreating 0          149m
```

2. **provisioner** ノードで、ネットワーク設定が存在することを判別します。

```
$ kubectl get network.config.openshift.io cluster -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  serviceNetwork:
  - 172.30.0.0/16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
```

存在しない場合には、インストーラーはこれを作成していません。インストーラーがこれを作成しなかった理由を判別するには、以下のコマンドを実行します。

```
$ openshift-install create manifests
```

3. **network-operator** が実行されていることを確認します。

```
$ kubectl -n openshift-network-operator get pods
```

4. ログを取得します。

```
$ kubectl -n openshift-network-operator logs -l "name=network-operator"
```

3 つ以上のコントロールプレーン (マスター) ノードを持つ高可用性クラスターの場合、Operator はリーダーの選択を実行し、他の Operator はすべてスリープ状態になります。詳細は、Troubleshooting を参照してください。

8.6.8.2. クラスターノードが DHCP 経由で正しい IPv6 アドレスを取得しない

クラスターノードが DHCP 経由で正しい IPv6 アドレスを取得しない場合は、以下の点を確認してください。

1. 予約された IPv6 アドレスが DHCP 範囲外にあることを確認します。
2. DHCP サーバーの IP アドレス予約では、予約で正しい DUID (DHCP 固有識別子) が指定されていることを確認します。以下に例を示します。

```
# This is a dnsmasq dhcp reservation, 'id:00:03:00:01' is the client id and '18:db:f2:8c:d5:9f' is the MAC Address for the NIC
id:00:03:00:01:18:db:f2:8c:d5:9f,openshift-master-1,[2620:52:0:1302::6]
```

3. Route Announcement が機能していることを確認します。

4. DHCP サーバーが、IP アドレス範囲を提供する必要なインターフェイスでリッスンしていることを確認します。

8.6.8.3. クラスターノードが DHCP 経由で正しいホスト名を取得しない

IPv6 のデプロイメント時に、クラスターノードは DHCP でホスト名を取得する必要があります。**NetworkManager** はホスト名をすぐに割り当てない場合があります。コントロールプレーン (マスター) ノードは、以下のようなエラーを報告する可能性があります。

```
Failed Units: 2
  NetworkManager-wait-online.service
  nodeip-configuration.service
```

このエラーは、最初に DHCP サーバーからホスト名を受信せずにクラスターノードが起動する可能性があることを示しています。これにより、**kubelet** が **localhost.localdomain** ホスト名で起動します。エラーに対処するには、ノードによるホスト名の更新を強制します。

手順

1. **hostname** を取得します。

   ```
   [core@master-X ~]$ hostname
   ```

   ホスト名が **localhost** の場合は、以下の手順に進みます。

   注記

   X は、コントロールプレーンノード (別名マスターノード) 番号になります。

2. クラスターノードによる DHCP リースの更新を強制します。

   ```
   [core@master-X ~]$ sudo nmcli con up "<bare-metal-nic>"
   ```

   <bare-metal-nic> を、**baremetal** ネットワークに対応する有線接続に置き換えます。

3. **hostname** を再度確認します。

   ```
   [core@master-X ~]$ hostname
   ```

4. ホスト名が **localhost.localdomain** の場合は、**NetworkManager** を再起動します。

   ```
   [core@master-X ~]$ sudo systemctl restart NetworkManager
   ```

5. ホスト名がまだ **localhost.localdomain** の場合は、数分待機してから再度確認します。ホスト名が **localhost.localdomain** のままの場合は、直前の手順を繰り返します。

6. **nodeip-configuration** サービスを再起動します。

   ```
   [core@master-X ~]$ sudo systemctl restart nodeip-configuration.service
   ```

   このサービスは、正しいホスト名の参照で **kubelet** サービスを再設定します。

7. kubelet が直前の手順で変更された後にユニットファイル定義を再読み込みします。

```
[core@master-X ~]$ sudo systemctl daemon-reload
```

8. **kubelet** サービスを再起動します。

```
[core@master-X ~]$ sudo systemctl restart kubelet.service
```

9. **kubelet** が正しいホスト名で起動されていることを確認します。

```
[core@master-X ~]$ sudo journalctl -fu kubelet.service
```

再起動時など、クラスターの稼働後にクラスターノードが正しいホスト名を取得しない場合、クラスターの **csr** は保留中になります。**csr** は承認 **しません**。承認すると、他の問題が生じる可能性があります。

csr の対応

1. クラスターで CSR を取得します。

```
$ oc get csr
```

2. 保留中の **csr** に **Subject Name: localhost.localdomain** が含まれているかどうかを確認します。

```
$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' | base64 --decode | openssl req -noout -text
```

3. **Subject Name: localhost.localdomain** が含まれる **csr** を削除します。

```
$ oc delete csr <wrong_csr>
```

8.6.8.4. ルートがエンドポイントに到達しない

インストールプロセス時に、VRRP (Virtual Router Redundancy Protocol) の競合が発生する可能性があります。この競合は、特定のクラスター名を使用してクラスターデプロイメントの一部であった、以前に使用された OpenShift Container Platform ノードが依然として実行中であるものの、同じクラスター名を使用した現在の OpenShift Container Platform クラスターデプロイメントの一部ではない場合に発生する可能性があります。たとえば、クラスターはクラスター名 **openshift** を使用してデプロイされ、3 つのコントロールプレーン (マスター) ノードと 3 つのワーカーノードをデプロイします。後に、別のインストールで同じクラスター名 **openshift** が使用されますが、この再デプロイメントは 3 つのコントロールプレーン (マスター) ノードのみをインストールし、以前のデプロイメントの 3 つのワーカーノードを **ON** 状態のままにします。これにより、VRID (Virtual Router Identifier) の競合が発生し、VRRP が競合する可能性があります。

1. ルートを取得します。

```
$ oc get route oauth-openshift
```

2. サービスエンドポイントを確認します。

```
$ oc get svc oauth-openshift
```

```
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
oauth-openshift   ClusterIP   172.30.19.162   <none>        443/TCP   59m
```

3. コントロールプレーン (マスター) ノードからサービスへのアクセスを試行します。

```
[core@master0 ~]$ curl -k https://172.30.19.162
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
```

4. **provisioner** ノードからの **authentication-operator** エラーを特定します。

```
$ oc logs deployment/authentication-operator -n openshift-authentication-operator
```

```
Event(v1.ObjectReference{Kind:"Deployment", Namespace:"openshift-authentication-
operator", Name:"authentication-operator", UID:"225c5bd5-b368-439b-9155-5fd3c0459d98",
APIVersion:"apps/v1", ResourceVersion:"", FieldPath:""}): type: 'Normal' reason:
'OperatorStatusChanged' Status for clusteroperator/authentication changed: Degraded
message changed from "IngressStateEndpointsDegraded: All 2 endpoints for oauth-server
are reporting"
```

解決策

1. すべてのデプロイメントのクラスター名が一意であり、競合が発生しないことを確認します。
2. 同じクラスター名を使用するクラスターデプロイメントの一部ではない不正なノードをすべてオフにします。そうしないと、OpenShift Container Platform クラスターの認証 Pod が正常に起動されなくなる可能性があります。

8.6.8.5. 初回起動時の Ignition の失敗

初回起動時に、Ignition 設定が失敗する可能性があります。

手順

1. Ignition 設定が失敗したノードに接続します。

```
Failed Units: 1
  machine-config-daemon-firstboot.service
```

2. **machine-config-daemon-firstboot** サービスを再起動します。

```
[core@worker-X ~]$ sudo systemctl restart machine-config-daemon-firstboot.service
```

8.6.8.6. NTP が同期しない

OpenShift Container Platform クラスターのデプロイメントは、クラスターノード間の NTP の同期クロックによって異なります。同期クロックがない場合、時間の差が 2 秒を超えるとクロックのドリフトによりデプロイメントが失敗する可能性があります。

手順

1. クラスターノードの **AGE** の差異の有無を確認します。以下に例を示します。

```
$ oc get nodes
```

```
NAME                         STATUS   ROLES    AGE   VERSION
master-0.cloud.example.com   Ready    master   145m   v1.16.2
master-1.cloud.example.com   Ready    master   135m   v1.16.2
master-2.cloud.example.com   Ready    master   145m   v1.16.2
worker-2.cloud.example.com   Ready    worker   100m   v1.16.2
```

2. クロックのドリフトによる一貫性のないタイミングの遅延について確認します。以下に例を示します。

```
$ oc get bmh -n openshift-machine-api
```

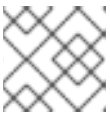
```
master-1    error registering master-1  ipmi://<out-of-band-ip>
```

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 18:20:02 UTC
           Universal time: Tue 2020-03-10 18:20:02 UTC
                 RTC time: Tue 2020-03-10 18:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

既存のクラスターでのクロックドリフトへの対応

1. ノードに配信される **chrony.conf** ファイルの内容を含む Butane 設定ファイルを作成します。以下の例で、**99-master-chrony.bu** を作成して、ファイルをコントロールプレーンノードに追加します。ワーカーノードのファイルを変更するか、ワーカーロールに対してこの手順を繰り返すことができます。



注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-master-chrony
  labels:
    machineconfiguration.openshift.io/role: master
```

```
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        server <NTP-server> iburst 1
        stratumweight 0
        driftfile /var/lib/chrony/drift
        rtcsync
        makestep 10 3
        bindcmdaddress 127.0.0.1
        bindcmdaddress ::1
        keyfile /etc/chrony.keys
        commandkey 1
        generatecommandkey
        noclientlog
        logchange 0.5
        logdir /var/log/chrony
```

1 **<NTP-server>** を NTP サーバーの IP アドレスに置き換えます。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-master-chrony.yaml**) を生成します。

```
$ butane 99-master-chrony.bu -o 99-master-chrony.yaml
```

3. **MachineConfig** オブジェクトファイルを適用します。

```
$ oc apply -f 99-master-chrony.yaml
```

4. **System clock synchronized** の値が **yes** であることを確認します。

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 19:10:02 UTC
           Universal time: Tue 2020-03-10 19:10:02 UTC
                 RTC time: Tue 2020-03-10 19:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

デプロイメントの前にクロック同期を設定するには、マニフェストファイルを生成し、このファイルを **openshift** ディレクトリーに追加します。以下に例を示します。

```
$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-chrony-configuration.yaml
```

クラスターの作成を継続します。

8.6.9. インストールの確認

インストール後に、インストーラーがノードおよび Pod を正常にデプロイしていることを確認します。

手順

1. OpenShift Container Platform クラスターノードが適切にインストールされると、以下の **Ready** 状態が **STATUS** 列に表示されます。

```
$ oc get nodes
```

```
NAME                 STATUS   ROLES          AGE  VERSION
master-0.example.com   Ready    master,worker   4h   v1.16.2
master-1.example.com   Ready    master,worker   4h   v1.16.2
master-2.example.com   Ready    master,worker   4h   v1.16.2
```

2. インストーラーによりすべての Pod が正常にデプロイされたことを確認します。以下のコマンドは、実行中の Pod、または出力の一部として完了した Pod を削除します。

```
$ oc get pods --all-namespaces | grep -iv running | grep -iv complete
```

第9章 Z/VM を使用した IBM Z および LINUXONE へのインストール

9.1. Z/VM を使用した IBM Z および LINUXONE へのインストール準備

9.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

9.1.2. z/VM を使用した OpenShift Container Platform の IBM Z または LinuxONE へのインストール方法の選択

以下の方法のいずれかを使用して、独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーに、z/VM を使用してクラスターをインストールできます。

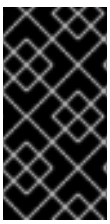
- **z/VM を使用したクラスターの IBM Z および LinuxONE へのインストール**: 独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーに、z/VM を使用した OpenShift Container Platform をインストールできます。
- **ネットワークが制限された環境での z/VM を使用したクラスターの IBM Z および LinuxONE へのインストール**: インストールリリースコンテンツの内部ミラーを使用して、ネットワークが制限または切断された環境で、独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーに z/VM を使用した OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

9.2. Z/VM を使用したクラスターの IBM Z および LINUXONE へのインストール

OpenShift Container Platform バージョン 4.8 では、独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーにクラスターをインストールできます。

注記

本書は IBM Z のみを参照しますが、これに含まれるすべての情報は LinuxONE にも適用されます。



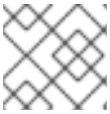
重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

9.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- クラスター用に NFS を使用して永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

9.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

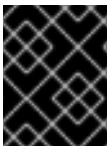
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

9.2.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表9.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

9.2.3.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

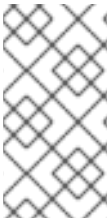
9.2.3.3. 最小の IBM Z システム環境

OpenShift Container Platform バージョン 4.8 は、以下の IBM ハードウェアにインストールできます。

- IBM z15 (すべてのモデル)、IBM z14 (すべてのモデル)、IBM z13、および IBM z13s
- LinuxONE(すべてのバージョン)

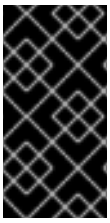
ハードウェア要件

- 6 つの IFL 相当 (これは、各クラスターで、SMT2 が有効になっている)。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。



注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。



重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR の加重管理、エンタイトルメント、および CPU 共有が重要なロールを果たします。

オペレーティングシステム要件

- z/VM 7.1 以降の 1 インスタンス

z/VM インスタンスで、以下をセットアップします。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Z ネットワーク接続の要件
IBM Z の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニ

ディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。

- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

9.2.3.4. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。
- HiperSockets。ノードに直接割り当てられるか、または z/VM ゲストに対して透過性を持たせるために z/VM VSWITCH でブリッジしてノードに割り当てられます。HiperSockets をノードに直接接続するには、RHEL 8 ゲスト経由で外部ネットワークにゲートウェイを設定し、Hipersockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- 高可用性を確保する場合は z/VM 7.1 以降の 2 または 3 インスタンス

z/VM インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用に 3 ゲスト仮想マシン (z/VM インスタンスごとに 1 つ)
- OpenShift Container Platform コンピュートマシン用に 6 以上のゲスト仮想マシン (z/VM インスタンス全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン
- オーバーコミット環境で必須コンポーネントの可用性を確保するには、CP コマンドの **SET SHARE** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM ドキュメントの SET SHARE を参照してください。

IBM Z ネットワーク接続の要件
IBM Z の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV および High Performance FICON (zHPF) を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

9.2.3.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM ドキュメントの Bridging a HiperSockets LAN with a z/VM Virtual Switch を参照してください。
- パフォーマンスの最適化については、Scaling HyperPAV alias devices on Linux guests on z/VM を参照してください。
- LPAR の加重管理とエンタイトルメントについて は、LPAR パフォーマンスのトピック を参照してください。
- IBM Z および LinuxONE 環境に推奨されるホストプラクティス

9.2.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノー

ドオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

9.2.3.6.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表9.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表9.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表9.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

9.2.3.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表9.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

9.2.3.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例9.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例9.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

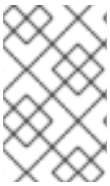
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

9.2.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。

   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

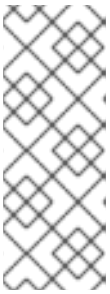
注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表9.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表9.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

9.2.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例9.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
```

```
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

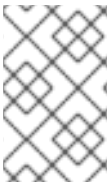
1. この例では、クラスター名は **ocp4** です。
2. ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。
3. 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。
4. ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。
6. ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。
7. ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

9.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

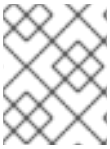
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

9.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

-

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

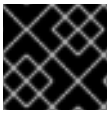
c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

9.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.2.7. インストールプログラムの取得

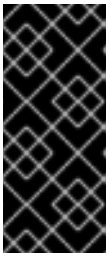
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

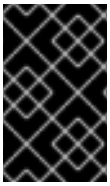
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.2.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

9.2.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

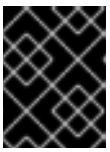
注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

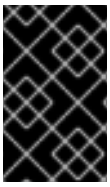
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

9.2.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

9.2.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表9.9 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値
pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

9.2.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表9.10 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23`

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
	
	複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

9.2.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表9.11 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。
	
	注記
	
	インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。
		1 つ以上のキー。以下に例を示します。
		
		`sshKey:`
		`  <key1>`
		`  <key2>`
		`  <key3>`

9.2.9.2. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

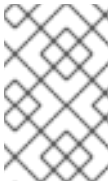
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : s390x
```

```
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。

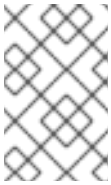
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

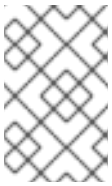
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

9.2.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
```

```
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.2.9.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

9.2.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

9.2.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表9.12 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表9.13 **defaultNetwork** オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定

以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表9.14 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
```

```
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定
以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表9.15 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表9.16 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表9.17 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

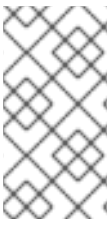
9.2.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

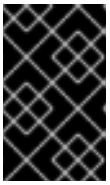
1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

9.2.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Z インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を z/VM ゲスト仮想マシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS z/VM ゲスト仮想マシンの再起動後に自動的に開始されます。

マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している HTTP または HTTPS サーバー。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) カーネル、 initramfs および rootfs ファイルを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な kernel、initramfs、および rootfs アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

注記

rootfs イメージは FCP および DASD の場合と同じです。

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **ip=** には、以下の 7 つのエントリーを指定します。

i. マシンの IP アドレス。

ii. 空の文字列。

iii. ゲートウェイ。

iv. ネットマスク。

v. **hostname.domainname** 形式のマシンホストおよびドメイン名。この値を省略して、RHCOS に決定させるようにします。

vi. ネットワークインターフェイス名。この値を省略して、RHCOS に決定させるようにします。

vii. 静的 IP アドレスを使用する場合、 **none** を指定します。

- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- DASD タイプのディスクへのインストールには、以下のタスクを実行します。

  i. **coreos.inst.install_dev=** には、**dasda** を指定します。

  ii. **rd.dasd=** を使用して、 RHCOS がインストールされる DASD を指定します。

  iii. その他のパラメーターはすべて変更しません。
  ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

  ```
  rd.neednet=1 \
  console=ttysclp0 \
  coreos.inst.install_dev=dasda \
  coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
  coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/bootstrap.ign \
  ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
  rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
  zfcp.allow_lun_scan=0 \
  rd.dasd=0.0.3490
  ```

  パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

- FCP タイプのディスクへのインストールには、以下のタスクを実行します。

  i. **rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して RHCOS がインストールされる FCP ディスクを指定します。マルチパスの場合、それぞれの追加のステップについてこのステップを繰り返します。

注記

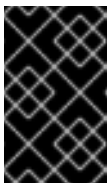
複数のパスを使用してインストールする場合は、問題が発生する可能性があるため、後でではなくインストールの直後にマルチパスを有効にする必要があります。

ii. インストールデバイスを **coreos.inst.install_dev=sda** に設定します。

注記

追加の LUN が NPIV で設定される場合は、FCP に **zfcp.allow_lun_scan=0** が必要です。CSI ドライバーを使用するために **zfcp.allow_lun_scan=1** を有効にする必要がある場合などには、各ノードが別のノードのブートパーティションにアクセスできないように NPIV を設定する必要があります。

iii. その他のパラメーターはすべて変更しません。



重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

以下は、マルチパスが設定されたワーカーノードのパラメーターファイルのサンプル **worker-1.parm** です。

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=sda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを z/VM に転送します。FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。

5. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
IBM ドキュメントの PUNCH を参照してください。

ヒント

CP PUNCH コマンドを使用するか、Linux を使用している場合は、**vmur** コマンドを使用して 2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

6. ブートストラップマシンで CMS にログインします。
7. リーダーからブートストラップマシンに対して IPL を実行します。

```
$ ipl c
```

IBM ドキュメントの IPL を参照してください。

8. クラスター内の他のマシンについてこの手順を繰り返します。

9.2.12.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

9.2.12.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**

- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

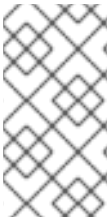
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードが使用されている場合の問題を回避するために、常にアクティブバックアップモードでオプション **fail_over_mac=1** を設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

9.2.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

9.2.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

9.2.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                    CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

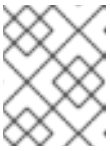
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

9.2.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
```

```
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

9.2.16.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

9.2.16.1.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

9.2.16.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

9.2.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running    0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

9.2.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

9.2.19. デバッグ情報の収集

IBM Z での OpenShift Container Platform インストールに関する特定の問題のトラブルシューティングおよびデバッグに役立つ可能性のあるデバッグ情報を収集できます。

前提条件

- **oc** CLI ツールをインストールしていること。

手順

1. クラスターにログインします。

```
$ oc login -u <username>
```

2. ハードウェア情報を収集するノードで、デバッグコンテナーを起動します。

```
$ oc debug node/<nodename>
```

3. **/host** ファイルシステムに切り替え、**toolbox** を起動します。

```
$ chroot /host
$ toolbox
```

4. **dbginfo** データを収集します。

```
$ dbginfo.sh
```

5. その後に、**scp** を使用するなどしてデータを取得できます。

関連情報

- How to generate SOSREPORT within OpenShift4 nodes without SSH

9.2.20. 次のステップ

- RHCOS でカーネル引数を使用してマルチパスを有効化します。

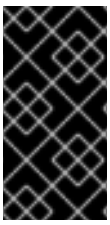
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

9.3. ネットワークが制限された環境での Z/VM を使用したクラスターの IBM Z および LINUXONE へのインストール

OpenShift Container Platform バージョン 4.8 では、制限されたネットワークでプロビジョニングする IBM Z および LinuxONE インフラストラクチャーにクラスターをインストールできます。

注記

本書では IBM Z のみについて説明していますが、すべての情報は LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

9.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得している。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

  重要

  インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- クラスター用に NFS を使用して永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

  注記

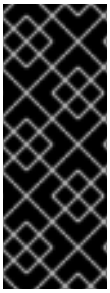
  プロキシーを設定する場合は、このサイト一覧も確認してください。

9.3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

9.3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

9.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

9.3.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表9.18 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

9.3.4.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表9.19 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

9.3.4.3. 最小の IBM Z システム環境

OpenShift Container Platform バージョン 4.8 は、以下の IBM ハードウェアにインストールできます。

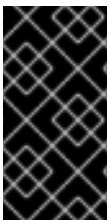
- IBM z15 (すべてのモデル)、IBM z14 (すべてのモデル)、IBM z13、および IBM z13s
- LinuxONE(すべてのバージョン)

ハードウェア要件

- 6 つの IFL 相当 (これは、各クラスターで、SMT2 が有効になっている)。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。



重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR の加重管理、エンタイトルメント、および CPU 共有が重要なロールを果たします。

オペレーティングシステム要件

- z/VM 7.1 以降の 1 インスタンス

z/VM インスタンスで、以下をセットアップします。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Z ネットワーク接続の要件
IBM Z の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

9.3.4.4. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。
- HiperSockets。ノードに直接割り当てられるか、または z/VM ゲストに対して透過性を持たせるために z/VM VSWITCH でブリッジしてノードに割り当てられます。HiperSockets をノードに直接接続するには、RHEL 8 ゲスト経由で外部ネットワークにゲートウェイを設定し、Hipersockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- 高可用性を確保する場合は z/VM 7.1 以降の 2 または 3 インスタンス

z/VM インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用に 3 ゲスト仮想マシン (z/VM インスタンスごとに 1 つ)
- OpenShift Container Platform コンピュートマシン用に 6 以上のゲスト仮想マシン (z/VM インスタンス全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン
- オーバーコミット環境で必須コンポーネントの可用性を確保するには、CP コマンドの **SET SHARE** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM ドキュメントの SET SHARE を参照してください。

IBM Z ネットワーク接続の要件
IBM Z の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM vSwitch のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV および High Performance FICON (zHPF) を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

9.3.4.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM ドキュメントの Bridging a HiperSockets LAN with a z/VM Virtual Switch を参照してください。

- パフォーマンスの最適化については、Scaling HyperPAV alias devices on Linux guests on z/VM を参照してください。
- LPAR の加重管理とエンタイトルメントについて は、LPAR パフォーマンスのトピック を参照してください。
- IBM Z および LinuxONE 環境に推奨されるホストプラクティス

9.3.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

9.3.4.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

9.3.4.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表9.20 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表9.21 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表9.22 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

9.3.4.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

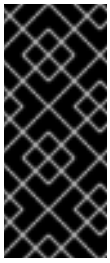
また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

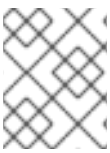
DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表9.23 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

9.3.4.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

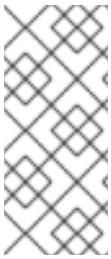
ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例9.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例9.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

9.3.4.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

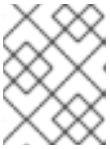
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

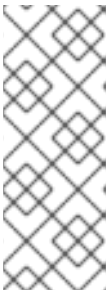
   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表9.24 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

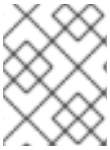
ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表9.25 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

9.3.4.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例9.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

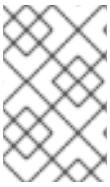
7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

9.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。

- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

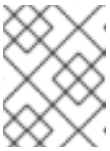
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

   注記

   DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

   注記

   DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

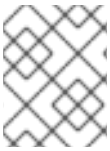
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

5. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

9.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain>
```
(1)

(1) **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

出力例

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

   a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
   ```

   出力例

   ```
   5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
   5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
   ```

   1 Kubernetes 内部 API のレコード名を指定します。

   2 Kubernetes API のレコード名を指定します。

   注記

   PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

   b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
   ```

   出力例

   ```
   96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
   ```

   c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

9.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

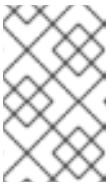
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.3.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

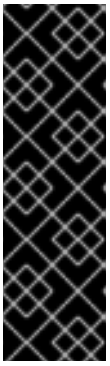
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



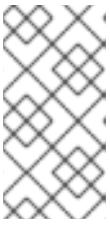
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

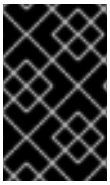
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

9.3.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

9.3.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表9.26 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

9.3.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表9.27 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

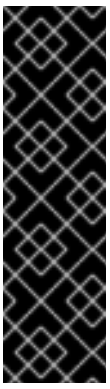
9.3.8.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表9.28 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。
	
	注記
	
	インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。
		1 つ以上のキー。以下に例を示します。
		
		`sshKey:`
		`  <key1>`
		`  <key2>`
		`  <key3>`

9.3.8.2. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

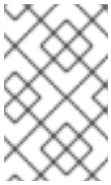
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : s390x
```

```
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



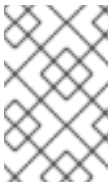
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



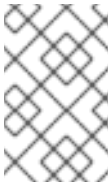
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

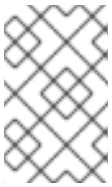
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

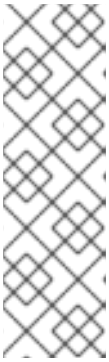
17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

9.3.8.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

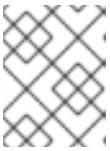
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照

するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

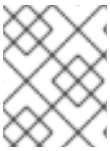


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

9.3.8.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

9.3.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

9.3.9.1. Cluster Network Operator 設定オブジェクト

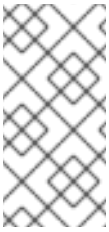
Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表9.29 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表9.30 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表9.31 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	タイプ	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表9.32 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表9.33 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

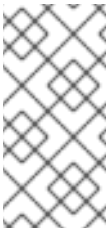
フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表9.34 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

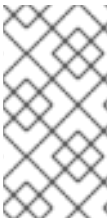
9.3.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
```

```
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

9.3.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Z インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を z/VM ゲスト仮想マシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS z/VM ゲスト仮想マシンの再起動後に自動的に開始されます。

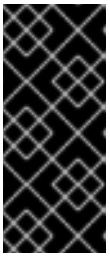
マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している HTTP または HTTPS サーバー。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) カーネル、 initramfs および rootfs ファイルを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な kernel、initramfs、および rootfs アーティファクトのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

注記

rootfs イメージは FCP および DASD の場合と同じです。

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。

- **ip=** には、以下の 7 つのエントリーを指定します。

  i. マシンの IP アドレス。

  ii. 空の文字列。

  iii. ゲートウェイ。

  iv. ネットマスク。

  v. **hostname.domainname** 形式のマシンホストおよびドメイン名。この値を省略して、RHCOS に決定させるようにします。

  vi. ネットワークインターフェイス名。この値を省略して、RHCOS に決定させるようにします。

  vii. 静的 IP アドレスを使用する場合、 **none** を指定します。

- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- DASD タイプのディスクへのインストールには、以下のタスクを実行します。

  i. **coreos.inst.install_dev=** には、**dasda** を指定します。

  ii. **rd.dasd=** を使用して、 RHCOS がインストールされる DASD を指定します。

  iii. その他のパラメーターはすべて変更しません。
  ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

  ```
  rd.neednet=1 \
  console=ttysclp0 \
  coreos.inst.install_dev=dasda \
  coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
  coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/bootstrap.ign \
  ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
  rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
  zfcp.allow_lun_scan=0 \
  rd.dasd=0.0.3490
  ```

  パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

- FCP タイプのディスクへのインストールには、以下のタスクを実行します。

  i. **rd.zfcp=<adapter>,<wwpn>,<lun>** を使用して RHCOS がインストールされる FCP ディスクを指定します。マルチパスの場合、それぞれの追加のステップについてこのステップを繰り返します。

注記

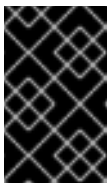
複数のパスを使用してインストールする場合は、問題が発生する可能性があるため、後でではなくインストールの直後にマルチパスを有効にする必要があります。

ii. インストールデバイスを **coreos.inst.install_dev=sda** に設定します。

注記

追加の LUN が NPIV で設定される場合は、FCP に **zfcp.allow_lun_scan=0** が必要です。CSI ドライバーを使用するために **zfcp.allow_lun_scan=1** を有効にする必要がある場合などには、各ノードが別のノードのブートパーティションにアクセスできないように NPIV を設定する必要があります。

iii. その他のパラメーターはすべて変更しません。



重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

以下は、マルチパスが設定されたワーカーノードのパラメーターファイルのサンプル **worker-1.parm** です。

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=sda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

パラメーターファイルのすべてのオプションを 1 行で記述し、改行文字がないことを確認します。

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを z/VM に転送します。FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。

5. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
IBM ドキュメントの PUNCH を参照してください。

ヒント

CP PUNCH コマンドを使用するか、Linux を使用している場合は、**vmur** コマンドを使用して 2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

6. ブートストラップマシンで CMS にログインします。
7. リーダーからブートストラップマシンに対して IPL を実行します。

```
$ ipl c
```

IBM ドキュメントの IPL を参照してください。

8. クラスター内の他のマシンについてこの手順を繰り返します。

9.3.11.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

9.3.11.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

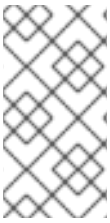
次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**

- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

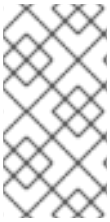
- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。



注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードが使用されている場合の問題を回避するために、常にアクティブバックアップモードでオプション **fail_over_mac=1** を設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

9.3.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

9.3.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

9.3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

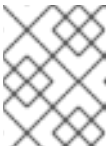
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

9.3.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.8.2     True        False         False     19m
baremetal                                  4.8.2     True        False         False     37m
cloud-credential                            4.8.2     True        False         False     40m
```

```
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

9.3.15.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

9.3.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

9.3.15.2.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

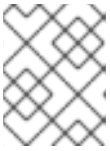
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

9.3.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

9.3.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
```

```
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                         READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running    0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running    0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

4. Cluster registration ページでクラスターを登録します。

9.3.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

9.3.18. デバッグ情報の収集

IBM Z での OpenShift Container Platform インストールに関する特定の問題のトラブルシューティングおよびデバッグに役立つ可能性のあるデバッグ情報を収集できます。

前提条件

- **oc** CLI ツールをインストールしていること。

手順

1. クラスターにログインします。

```
$ oc login -u <username>
```

2. ハードウェア情報を収集するノードで、デバッグコンテナーを起動します。

```
$ oc debug node/<nodename>
```

3. **/host** ファイルシステムに切り替え、**toolbox** を起動します。

```
$ chroot /host
$ toolbox
```

4. **dbginfo** データを収集します。

```
$ dbginfo.sh
```

5. その後に、**scp** を使用するなどしてデータを取得できます。

関連情報

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH

9.3.19. 次のステップ

- クラスターをカスタマイズします。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。

第10章 RHEL KVM を使用した IBM Z および LINUXONE へのインストール

10.1. RHEL KVM を使用した IBM Z および LINUXONE へのインストール準備

10.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

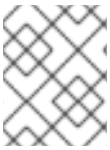
10.1.2. RHEL KVM を使用した OpenShift Container Platform の IBM Z または LinuxONE へのインストール方法の選択

以下の方法のいずれかを使用して、独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーに、RHEL KVM を使用してクラスターをインストールできます。

- **RHEL KVM を使用したクラスターの IBM Z および LinuxONE へのインストール**: 独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーに、KVM を使用して OpenShift Container Platform をインストールできます。
- **ネットワークが制限された環境での RHEL KVM を使用したクラスターの IBM Z および LinuxONE へのインストール**: インストールリリースコンテンツの内部ミラーを使用して、ネットワークが制限または切断された環境で、独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーに RHEL KVM を使用して OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

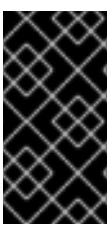
10.2. RHEL KVM を使用したクラスターの IBM Z および LINUXONE へのインストール

OpenShift Container Platform バージョン 4.8 では、独自にプロビジョニングする IBM Z または LinuxONE インフラストラクチャーにクラスターをインストールできます。



注記

本書は IBM Z のみを参照しますが、これに含まれるすべての情報は LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

10.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- クラスター用に NFS を使用して永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、 **Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- 論理パーティション (LPAR) でホストされ、RHEL 8.4 以降をベースとする RHEL Kernel Virtual Machine (KVM) システムをプロビジョニングしている。

注記

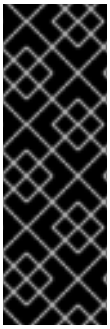
プロキシーを設定する場合は、このサイト一覧も確認してください。

10.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

10.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

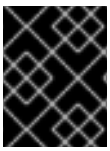
RHEL 8.4 以降をベースとする 1 つ以上の KVM ホストマシン。各 RHEL KVM ホストマシンで libvirt がインストールされ、実行している必要があります。仮想マシンは、各 RHEL KVM ホストマシンでプロビジョニングされます。

10.2.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表10.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる RHEL インスタンスにコントロールプレーンマシンを分散します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

10.2.3.2. ネットワーク接続の要件

OpenShift Container Platform インストーラーは、すべての Red Hat Enterprise Linux CoreOS (RHCOS) 仮想マシンに必要な Ignition ファイルを作成します。OpenShift Container Platform の自動インストールはブートストラップマシンで実行されます。これは各ノードで OpenShift Container Platform のインストールを開始し、Kubernetes クラスターを起動してから終了します。このブートストラップ時に、仮想マシンには Dynamic Host Configuration Protocol (DHCP) サーバーまたは静的 IP アドレスでネットワーク接続を確立している必要があります。

10.2.3.3. IBM Z ネットワーク接続の要件

RHEL KVM の IBM Z にインストールするには、以下が必要です。

- OSA または RoCE ネットワークアダプターで設定された RHEL KVM ホスト。
- libvirt または MacVTap のブリッジネットワークを使用してネットワークをゲストに接続するように設定されているいずれかの RHEL KVM ホスト。

仮想ネットワーク接続の種類 を参照してください。

10.2.3.4. ホストマシンのリソース要件

お使いの環境の RHEL KVM ホストは、OpenShift Container Platform 環境用に計画している仮想マシンをホストするために以下の要件を満たす必要があります。仮想化の使用開始 を参照してください。

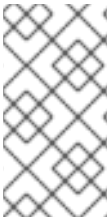
OpenShift Container Platform バージョン 4.8 は、以下の IBM ハードウェアにインストールできます。

- IBM z15 (すべてのモデル)、IBM z14 (すべてのモデル)、IBM z13、および IBM z13s
- LinuxONE(すべてのバージョン)

10.2.3.5. 最小の IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当 (これは、各クラスターで、SMT2 が有効になっている)。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。



注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR の加重管理、エンタイトルメント、および CPU 共有が重要なロールを果たします。

オペレーティングシステム要件

- libvirt で管理される、KVM で RHEL 8.4 以降を実行する 1 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

10.2.3.6. 最小リソース要件

それぞれのクラスターの仮想マシンは、以下の最小要件を満たしている必要があります。

仮想マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

10.2.3.7. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

オペレーティングシステム要件

- 高可用性が必要な場合は、libvirt で管理される、KVM で RHEL 8.4 以降を実行する 2 または 3 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コンピュートプレーンマシン用に 3 つのゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- OpenShift Container Platform コンピュートマシン用に 6 つ以上のゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン
- オーバーコミット環境で必須コンポーネントの可用性を確保するには、**cpu_shares** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM ドキュメントの schedinfo を参照してください。

10.2.3.8. 優先されるリソース要件

各クラスターの仮想マシンについての優先される要件は以下の通りです。

仮想マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB

仮想マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ
コントロールプレーン	RHCOS	8	16 GB	120 GB
コンピュート	RHCOS	6	8 GB	120 GB

10.2.3.9. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM Z および LinuxONE 環境に推奨されるホストプラクティス

10.2.3.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスがユーザーによってプロビジョニングされるインフラストラクチャーで利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノー

ドオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

10.2.3.10.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

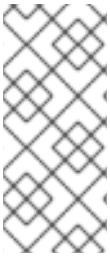
10.2.3.10.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。



注記

RHEL KVM ホストは、libvirt または MacVTap のブリッジネットワークを使用して、ネットワークを仮想マシンに接続するように設定される必要があります。仮想マシンには、RHEL KVM ホストに接続されているネットワークへのアクセスがある必要があります。KVM 内の仮想ネットワーク (ネットワークアドレス変換 (NAT) など) はサポートされる設定ではありません。

表10.2 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート

プロトコル	ポート	説明
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-9999	ポート **9100-9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000-32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表10.3 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表10.4 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

10.2.3.11. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表10.5 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>. <base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

10.2.3.11.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例10.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例10.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

10.2.3.12. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

   表10.6 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

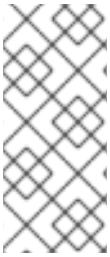
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

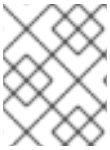
表10.7 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

10.2.3.12.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例10.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1) この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

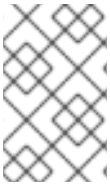
7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

10.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。

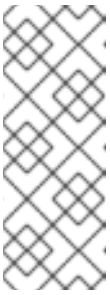
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

   注記

   DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始**のセクションを参照してください。

   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項については、**DHCP を使用したクラスターノードのホスト名の設定** 参照してください。

   注記

   DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. Red Hat Enterprise Linux CoreOS (RHCOS) の 高速インストールまたは Red Hat Enterprise Linux CoreOS (RHCOS) のフルインストールのいずれかの実行を選択します。フルインストールでは、HTTP または HTTPS サーバーを設定し、Ignition ファイルを提供し、イメージをクラスターノードにインストールする必要があります。高速インストールの場合、HTTP または HTTPS サーバーは必要はありませんが、DHCP サーバーが必要です。高速インストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成およびフルインストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成のセクションを参照してください。

3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。

5. クラスターに必要な DNS インフラストラクチャーを設定します。

   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

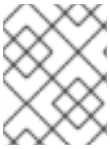
b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

10.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain>
```
(1)

(1) **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

出力例

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

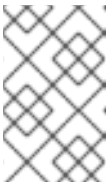
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

10.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

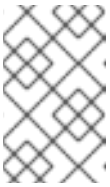
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.2.7. インストールプログラムの取得

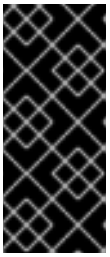
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

10.2.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

10.2.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

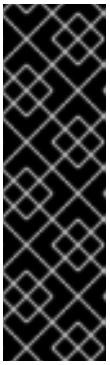
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

   ```
   $ mkdir <installation_directory>
   ```

   重要

   ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

   注記

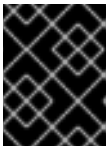
   この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

   注記

   一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

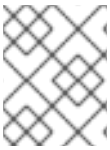


重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

10.2.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

10.2.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表10.8 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

10.2.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表10.9 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

10.2.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表10.10 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。
	
	注記
	
	インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。
		1 つ以上のキー。以下に例を示します。
		
		`sshKey:`
		`  <key1>`
		`  <key2>`
		`  <key3>`

10.2.9.2. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture : s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
```

```
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。

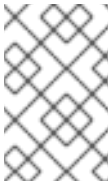
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

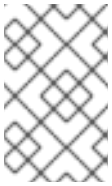
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

10.2.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
```

```
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

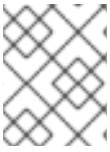
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.2.9.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

10.2.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

10.2.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表10.11 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表10.12 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定

以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表10.13 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
```

```
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定
以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表10.14 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表10.15 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表10.16 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

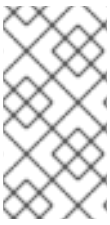
10.2.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

10.2.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform をプロビジョニングする IBM Z インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を Red Hat Enterprise Linux (RHEL) ゲスト仮想マシンとしてインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

事前にパッケージ化された QEMU コピーオンライト (QCOW2) ディスクイメージを使用する RHCOS の高速インストールを実行できます。または、新規の QCOW2 ディスクイメージでフルインストールを実行できます。

10.2.12.1. 事前にパッケージ化された QCOW2 ディスクイメージを使用した高速インストール

Red Hat Enterprise Linux CoreOS (RHCOS) の高速インストールでマシンを作成するには、事前にパッケージ化された Red Hat Enterprise Linux CoreOS (RHCOS)QEMU コピーオンライト (QCOW2) ディスクイメージをインポートします。

前提条件

- KVM を使用して RHEL 8.4 を実行している 1 つ以上の LPAR(この手順では RHEL KVM ホストと呼ばれます)。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- IP アドレスを提供する DHCP サーバー。

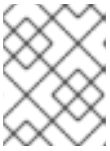
手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから RHEL QEMU コピーオンライト (QCOW2) ディスクイメージファイルを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

2. QCOW2 ディスクイメージおよび Ignition ファイルを RHEL KVM ホストの共通ディレクトリーにダウンロードします。
例: **/var/lib/libvirt/images**



注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. 各 KVM ゲストノードの QCOW2 ディスクイメージバッキングファイルで、新しいディスクイメージを作成します。

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Ignition ファイルと新規ディスクイメージを使用して、新規 KVM ゲストノードを作成します。

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vn_name} \
  --memory {memory} \
  --vcpus {vcpus} \
  --disk {disk} \
  --import \
  --network network={network},mac={mac} \
  --qemu-commandline="-drive \
if=none,id=ignition,format=raw,file={ign_file},readonly=on -device virtio-
blk,serial=ignition,drive=ignition"
```

10.2.12.2. 新規 QCOW2 ディスクイメージへのフルインストール

新規 QEMU copy-on-write (QCOW2) ディスクイメージのフルインストールでマシンを作成するには、以下の手順を実施します。

前提条件

- KVM を使用して RHEL 8.4 を実行している 1 つ以上の LPAR(この手順では RHEL KVM ホストと呼ばれます)。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- HTTP または HTTPS サーバーが設定されている。

手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページ、または RHCOS イメージミラーページから RHEL カーネル、initramfs、および rootfs ファイルを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

2. **virt-install** を起動する前に、ダウンロードした RHEL ライブカーネル、initramfs、および rootfs、および Ignition ファイルを HTTP または HTTPS サーバーに移動します。

注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. RHEL カーネル、initramfs、および Ignition ファイル、新規ディスクイメージ、および調整された parm 引数を使用して、新規 KVM ゲストノードを作成します。

- **--location** には、HTTP サーバーまたは HTTPS サーバーのカーネル/initrd の場所を指定します。
- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
- **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

```
$ virt-install \
  --connect qemu:///system \
  --name {vn_name} \
  --vcpus {vcpus} \
  --memory {memory_mb} \
  --disk {vn_name}.qcow2,size={image_size| default(10,true)} \
  --network network={virt_network_parm} \
  --boot hd \
  --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
  --extra-args "rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=vda
coreos.live.rootfs_url={rhcos_liveos} ip={ip}::{default_gateway}:{subnet_mask_length}:
{vn_name}:enc1:none:{MTU} nameserver={dns} coreos.inst.ignition_url={rhcos_ign}" \
  --noautoconsole \
  --wait
```

10.2.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

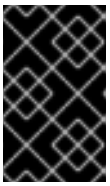
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

10.2.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

10.2.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
master-2 Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

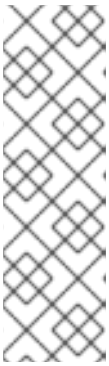
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                 CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

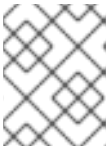
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

10.2.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

10.2.16.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

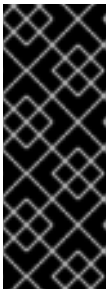
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

10.2.16.1.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。



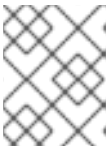
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

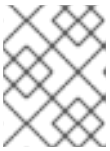
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME           VERSION                      AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                        True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

10.2.16.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

10.2.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
```

```
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

10.2.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

10.2.19. デバッグ情報の収集

IBM Z での OpenShift Container Platform インストールに関する特定の問題のトラブルシューティングおよびデバッグに役立つ可能性のあるデバッグ情報を収集できます。

前提条件

- **oc** CLI ツールをインストールしていること。

手順

1. クラスターにログインします。

```
$ oc login -u <username>
```

2. ハードウェア情報を収集するノードで、デバッグコンテナーを起動します。

```
$ oc debug node/<nodename>
```

3. **/host** ファイルシステムに切り替え、**toolbox** を起動します。

```
$ chroot /host
$ toolbox
```

4. **dbginfo** データを収集します。

```
$ dbginfo.sh
```

5. その後に、**scp** を使用するなどしてデータを取得できます。

関連情報

- How to generate SOSREPORT within OpenShift4 nodes without SSH

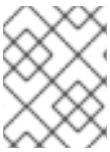
10.2.20. 次のステップ

- クラスターをカスタマイズします。

- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

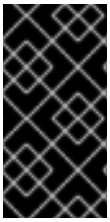
10.3. ネットワークが制限された環境での RHEL KVM のあるクラスターの IBM Z および LINUXONE へのインストール

OpenShift Container Platform バージョン 4.8 では、制限されたネットワークでプロビジョニングする IBM Z および LinuxONE インフラストラクチャーにクラスターをインストールできます。



注記

本書では IBM Z のみについて説明していますが、すべての情報は LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

10.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

重要

インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- クラスター用に NFS を使用して永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、 **Read Write Many** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。
- 論理パーティション (LPAR) でホストされ、RHEL 8.4 以降をベースとする RHEL Kernel Virtual Machine (KVM) システムをプロビジョニングしている。

注記

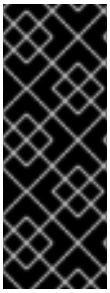
プロキシーを設定する場合は、このサイト一覧も確認してください。

10.3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

10.3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

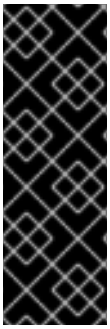
- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

10.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

10.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

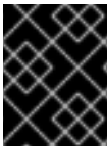
RHEL 8.4 以降をベースとする 1 つ以上の KVM ホストマシン。各 RHEL KVM ホストマシンで libvirt がインストールされ、実行している必要があります。仮想マシンは、各 RHEL KVM ホストマシンでプロビジョニングされます。

10.3.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表10.17 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる RHEL インスタンスにコントロールプレーンマシンを分散します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

10.3.4.2. ネットワーク接続の要件

OpenShift Container Platform インストーラーは、すべての Red Hat Enterprise Linux CoreOS (RHCOS) 仮想マシンに必要な Ignition ファイルを作成します。OpenShift Container Platform の自動インストールはブートストラップマシンで実行されます。これは各ノードで OpenShift Container Platform のインストールを開始し、Kubernetes クラスターを起動してから終了します。このブートストラップ時に、仮想マシンには Dynamic Host Configuration Protocol (DHCP) サーバーまたは静的 IP アドレスでネットワーク接続を確立している必要があります。

10.3.4.3. IBM Z ネットワーク接続の要件

RHEL KVM の IBM Z にインストールするには、以下が必要です。

- OSA または RoCE ネットワークアダプターで設定された RHEL KVM ホスト。
- libvirt または MacVTap のブリッジネットワークを使用してネットワークをゲストに接続するように設定されているいずれかの RHEL KVM ホスト。
  仮想ネットワーク接続の種類 を参照してください。

10.3.4.4. ホストマシンのリソース要件

お使いの環境の RHEL KVM ホストは、OpenShift Container Platform 環境用に計画している仮想マシンをホストするために以下の要件を満たす必要があります。仮想化の使用開始 を参照してください。

OpenShift Container Platform バージョン 4.8 は、以下の IBM ハードウェアにインストールできます。

- IBM z15 (すべてのモデル)、IBM z14 (すべてのモデル)、IBM z13、および IBM z13s
- LinuxONE(すべてのバージョン)

10.3.4.5. 最小の IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当 (これは、各クラスターで、SMT2 が有効になっている)。
- 最低でもネットワーク接続 1 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

注記

専用または共有 IFL を使用して、十分なコンピューティングリソースを割り当てることができます。リソース共有は IBM Z の重要な強みの 1 つです。ただし、各ハイパーバイザーレイヤーで容量を正しく調整し、すべての OpenShift Container Platform クラスターに十分なリソースを確保する必要があります。

重要

クラスターの全体的なパフォーマンスに影響を与える可能性があるため、OpenShift Container Platform クラスターの設定に使用される LPAR には十分なコンピューティング能力が必要です。このコンテキストでは、ハイパーバイザーレベルでの LPAR の加重管理、エンタイトルメント、および CPU 共有が重要なロールを果たします。

オペレーティングシステム要件

- libvirt で管理される、KVM で RHEL 8.4 以降を実行する 1 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

10.3.4.6. 最小リソース要件

それぞれのクラスターの仮想マシンは、以下の最小要件を満たしている必要があります。

仮想マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

10.3.4.7. 推奨される IBM Z システム環境

ハードウェア要件

- 6 つの IFL 相当がそれぞれ割り当てられた LPARS 3 つ (これは、各クラスターで、SMT2 が有効になっている)。
- ネットワーク接続 2 つ。これで、**LoadBalancer** サービスに接続するだけでなく、クラスター外のトラッフィクに関するデータを提供します。

オペレーティングシステム要件

- 高可用性が必要な場合は、libvirt で管理される、KVM で RHEL 8.4 以降を実行する 2 または 3 つの LPAR。

RHEL KVM ホストで、以下を設定します。

- OpenShift Container Platform コンピュートプレーンマシン用に 3 つのゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- OpenShift Container Platform コンピュートマシン用に 6 つ以上のゲスト仮想マシン (RHEL KVM ホストマシン全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

- オーバーコミット環境で必須コンポーネントの可用性を確保するには、**cpu_shares** を使用してコントロールプレーンの優先度を引き上げます。インフラストラクチャーノードが存在する場合は、同じ操作を行います。IBM ドキュメントの schedinfo を参照してください。

10.3.4.8. 優先されるリソース要件

各クラスターの仮想マシンについての優先される要件は以下の通りです。

仮想マシン	オペレーティングシステム	vCPU	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	8	16 GB	120 GB
コンピュート	RHCOS	6	8 GB	120 GB

10.3.4.9. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

関連情報

- IBM Z および LinuxONE 環境に推奨されるホストプラクティス

10.3.4.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

10.3.4.10.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表10.18 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表10.19 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表10.20 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

10.3.4.11. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表10.21 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

10.3.4.11.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例10.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H  ; refresh (3 hours)
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
  IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例10.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

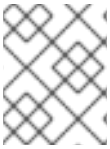
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

10.3.4.12. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

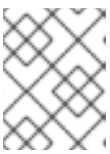
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

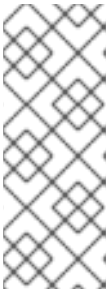
   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表10.22 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

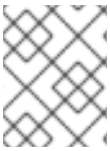
ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表10.23 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

10.3.4.12.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例10.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

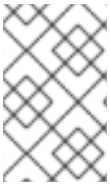
7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

10.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. Red Hat Enterprise Linux CoreOS (RHCOS) の 高速インストールまたは Red Hat Enterprise Linux CoreOS (RHCOS) のフルインストールのいずれかの実行を選択します。フルインストールでは、HTTP または HTTPS サーバーを設定し、Ignition ファイルを提供し、イメージをクラスターノードにインストールする必要があります。高速インストールの場合、HTTP または HTTPS サーバーは必要はありませんが、DHCP サーバーが必要です。高速インストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成およびフルインストール: Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成のセクションを参照してください。
4. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
6. クラスターに必要な DNS インフラストラクチャーを設定します。
   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
7. DNS 設定を検証します。
   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
8. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

10.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

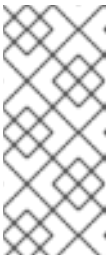
   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1. Kubernetes 内部 API のレコード名を指定します。
2. Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

10.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

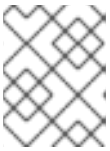
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

10.3.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

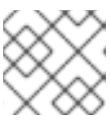
1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



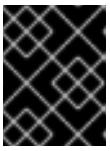
注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

10.3.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

10.3.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表10.24 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

10.3.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表10.25 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

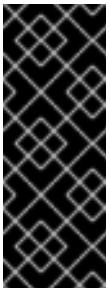
パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

10.3.8.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

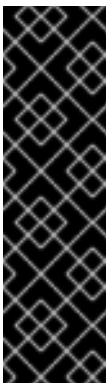
表10.26 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

10.3.8.2. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
  architecture : s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture : s390x
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

```
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。

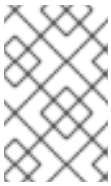
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネット



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

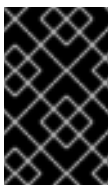
12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

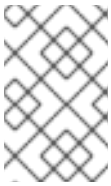


重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

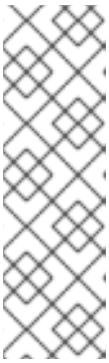
17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

10.3.8.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
```

```
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

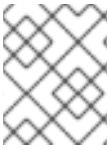
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

10.3.8.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

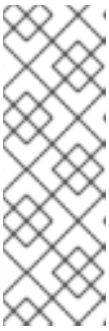
前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

10.3.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

10.3.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表10.27 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表10.28 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表10.29 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表10.30 **ovnKubernetesConfig** object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表10.31 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表10.32 kubeProxyConfig オブジェクト

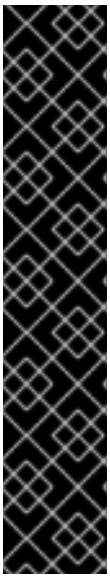
フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

10.3.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
```

```
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

10.3.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform をプロビジョニングする IBM Z インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を Red Hat Enterprise Linux (RHEL) ゲスト仮想マシンとしてインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

事前にパッケージ化された QEMU コピーオンライト (QCOW2) ディスクイメージを使用する RHCOS の高速インストールを実行できます。または、新規の QCOW2 ディスクイメージでフルインストールを実行できます。

10.3.11.1. 事前にパッケージ化された QCOW2 ディスクイメージを使用した高速インストール

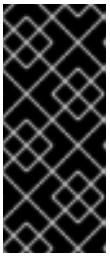
Red Hat Enterprise Linux CoreOS (RHCOS) の高速インストールでマシンを作成するには、事前にパッケージ化された Red Hat Enterprise Linux CoreOS (RHCOS)QEMU コピーオンライト (QCOW2) ディスクイメージをインポートします。

前提条件

- KVM を使用して RHEL 8.4 を実行している 1 つ以上の LPAR(この手順では RHEL KVM ホストと呼ばれます)。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- IP アドレスを提供する DHCP サーバー。

手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから RHEL QEMU コピーオンライト (QCOW2) ディスクイメージファイルを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

2. QCOW2 ディスクイメージおよび Ignition ファイルを RHEL KVM ホストの共通ディレクトリーにダウンロードします。
例: **/var/lib/libvirt/images**

注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. 各 KVM ゲストノードの QCOW2 ディスクイメージバッキングファイルで、新しいディスクイメージを作成します。

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Ignition ファイルと新規ディスクイメージを使用して、新規 KVM ゲストノードを作成します。

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vn_name} \
  --memory {memory} \
  --vcpus {vcpus} \
  --disk {disk} \
  --import \
  --network network={network},mac={mac} \
  --qemu-commandline="-drive \
if=none,id=ignition,format=raw,file={ign_file},readonly=on -device virtio-
blk,serial=ignition,drive=ignition"
```

10.3.11.2. 新規 QCOW2 ディスクイメージへのフルインストール

新規 QEMU copy-on-write (QCOW2) ディスクイメージのフルインストールでマシンを作成するには、以下の手順を実施します。

前提条件

- KVM を使用して RHEL 8.4 を実行している 1 つ以上の LPAR(この手順では RHEL KVM ホストと呼ばれます)。
- KVM/QEMU ハイパーバイザーが RHEL KVM ホストにインストーされている
- ノードのホスト名および逆引き参照を実行できるドメインネームサーバー (DNS)。
- HTTP または HTTPS サーバーが設定されている。

手順

1. Red Hat カスタマーポータルの 製品のダウンロード ページ、または RHCOS イメージミラーページから RHEL カーネル、initramfs、および rootfs ファイルを取得します。

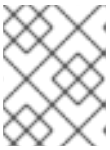
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な RHCOS QCOW2 イメージのみを使用します。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- kernel: **rhcos-<version>-live-kernel-<architecture>**
- initramfs: **rhcos-<version>-live-initramfs.<architecture>.img**
- rootfs: **rhcos-<version>-live-rootfs.<architecture>.img**

2. **virt-install** を起動する前に、ダウンロードした RHEL ライブカーネル、initramfs、および rootfs、および Ignition ファイルを HTTP または HTTPS サーバーに移動します。



注記

Ignition ファイルは OpenShift Container Platform インストーラーによって生成されます。

3. RHEL カーネル、initramfs、および Ignition ファイル、新規ディスクイメージ、および調整された parm 引数を使用して、新規 KVM ゲストノードを作成します。

- **--location** には、HTTP サーバーまたは HTTPS サーバーのカーネル/initrd の場所を指定します。
- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
- **coreos.live.rootfs_url=** の場合、起動しているカーネルおよび initramfs の一致する rootfs アーティファクトを指定します。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

```
$ virt-install \
  --connect qemu:///system \
  --name {vn_name} \
  --vcpus {vcpus} \
  --memory {memory_mb} \
  --disk {vn_name}.qcow2,size={image_size| default(10,true)} \
  --network network={virt_network_parm} \
  --boot hd \
  --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
  --extra-args "rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=vda
coreos.live.rootfs_url={rhcos_liveos} ip={ip}::{default_gateway}:{subnet_mask_length}:
{vn_name}:enc1:none:{MTU} nameserver={dns} coreos.inst.ignition_url={rhcos_ign}" \
  --noautoconsole \
  --wait
```

10.3.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

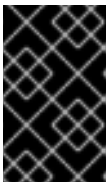
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

10.3.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

10.3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
master-2 Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                 CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

10.3.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

10.3.15.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

10.3.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

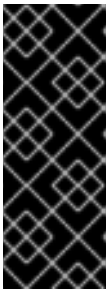
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

10.3.15.2.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                       AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                           True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

   - 以下を実行します。

     ```
     $ oc edit configs.imageregistry/cluster
     ```

     次に、行を変更します。

     ```
     managementState: Removed
     ```

     次のように変更してください。

     ```
     managementState: Managed
     ```

10.3.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

  ```
  $ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
  ```

  警告

  実稼働用以外のクラスターにのみこのオプションを設定します。

  イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

  ```
  Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
  ```

  数分待機した後に、このコマンドを再び実行します。

10.3.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver             apiserver-67b9g                               1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                               1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8     1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

4. Cluster registration ページでクラスターを登録します。

10.3.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

10.3.18. デバッグ情報の収集

IBM Z での OpenShift Container Platform インストールに関する特定の問題のトラブルシューティングおよびデバッグに役立つ可能性のあるデバッグ情報を収集できます。

前提条件

- **oc** CLI ツールをインストールしていること。

手順

1. クラスターにログインします。

```
$ oc login -u <username>
```

2. ハードウェア情報を収集するノードで、デバッグコンテナーを起動します。

```
$ oc debug node/<nodename>
```

3. **/host** ファイルシステムに切り替え、**toolbox** を起動します。

```
$ chroot /host
$ toolbox
```

4. **dbginfo** データを収集します。

```
$ dbginfo.sh
```

5. その後に、**scp** を使用するなどしてデータを取得できます。

関連情報

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH

10.3.19. 次のステップ

- クラスターをカスタマイズします。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。

第11章 IBM POWER SYSTEMS へのインストール

11.1. IBM POWER SYSTEMS へのインストールの準備

11.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

11.1.2. IBM Power Systems に OpenShift Container Platform をインストールする方法の選択

以下の方法のいずれかを使用して、独自にプロビジョニングする IBM Power Systems インフラストラクチャーにクラスターをインストールできます。

- **クラスターの IBM Power Systems へのインストール**: OpenShift Container Platform を独自にプロビジョニングする IBM Power Systems インフラストラクチャーにインストールできます。
- **ネットワークが制限された環境での IBM Power Systems へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、独自にプロビジョニングする IBM Power Systems インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

11.2. クラスターの IBM POWER SYSTEMS へのインストール

OpenShift Container Platform バージョン 4.8 では、プロビジョニングする IBM Power Systems インフラストラクチャーにクラスターをインストールできます。

重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

11.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

- クラスター用に NFS を使用して永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

11.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

11.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

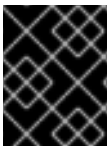
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

11.2.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表11.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

11.2.3.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表11.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	2	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	2	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

11.2.3.3. IBM Power Systems の最小要件

OpenShift Container Platform バージョン 4.8 は、以下の IBM ハードウェアにインストールできます。

- IBM POWER8 または POWER9 プロセッサーベースのシステム

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 IBM Power ベアメタルサーバーまたは 6 LPAR

オペレーティングシステム要件

- IBM POWER8 または POWER9 プロセッサーベースのシステムの 1 つインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 共有イーサネットアダプターを使用して仮想 I/O サーバーによって仮想化される
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 100GB / 16GB
- OpenShift Container Platform コンピュートマシン用に 100 GB / 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 100 GB / 16 GB

11.2.3.4. 推奨される IBM Power システム要件

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 IBM Power ベアメタルサーバーまたは 6 LPAR

オペレーティングシステム要件

- IBM POWER8 または POWER9 プロセッサーベースのシステムの 1 つインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 共有イーサネットアダプターを使用して仮想 I/O サーバーによって仮想化される
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 120 GB / 32 GB
- OpenShift Container Platform コンピュートマシン用に 120 GB / 32 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 120 GB / 16 GB

11.2.3.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

11.2.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

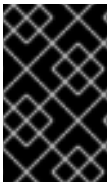
マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

11.2.3.6.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表11.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表11.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表11.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

11.2.3.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

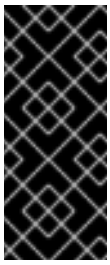
また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表11.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

11.2.3.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例11.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例11.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

11.2.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。

   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表11.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表11.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

11.2.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

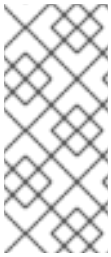
例11.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
```

```
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (6)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (7)
  bind *:80
  mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1. この例では、クラスター名は **ocp4** です。
2. ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。
3. 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。
4. ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。
6. ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。
7. ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

11.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

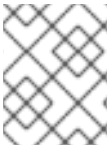
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

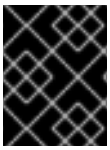


注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

11.2.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

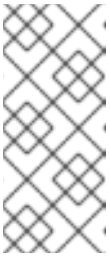
   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

-



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

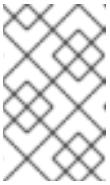
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

11.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

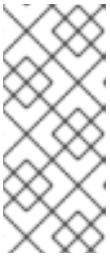
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

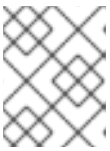
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

11.2.7. インストールプログラムの取得

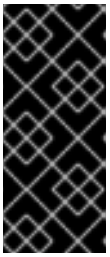
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

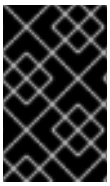
4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

11.2.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

11.2.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

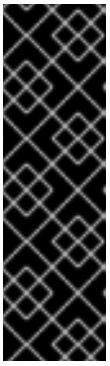
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

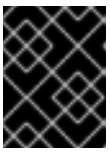
注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

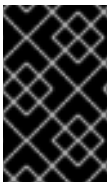
install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

11.2.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

11.2.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表11.9 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値
pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

11.2.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表11.10 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23`

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
	
	複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

11.2.9.1.3. オプションの設定パラメーター

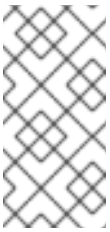
オプションのインストール設定パラメーターは、以下の表で説明されています。

表11.11 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**

パラメーター	説明	値
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**

パラメーター	説明	値
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。 **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。

パラメーター	説明	値
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

11.2.9.2. IBM Z のサンプル install-config.yaml ファイル

11.2.9.3. IBM Power Systems のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture : ppc64le
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture : ppc64le
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
```

```
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

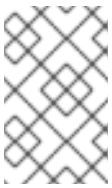
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



注記

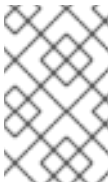
3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラ

フィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。IBM Power Systems インフラストラクチャー

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

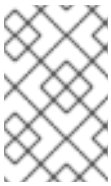
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

11.2.9.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

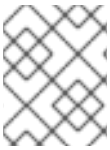
4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

11.2.9.5. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

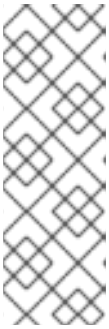
- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

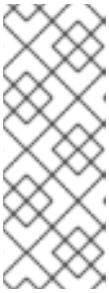
```
compute:
```

```
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。



注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

11.2.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

11.2.10.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表11.12 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。
		
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		
		マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。
		
		`spec:`
		`  serviceNetwork:`
		`  - 172.30.0.0/14`
		
		マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表11.13 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表11.14 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	タイプ	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表11.15 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表11.16 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表11.17 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

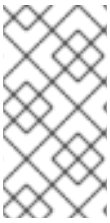
11.2.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

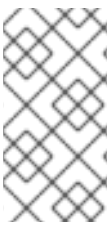
重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは ppc64le でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

11.2.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Power Systems インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

ISO イメージまたはネットワーク PXE ブートを使用する手順を実行して RHCOS をマシンにインストールできます。

11.2.12.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

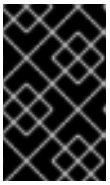
手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージミラー RHCOS イメージミラー ミラーページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-
live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

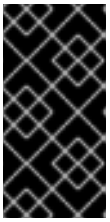
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

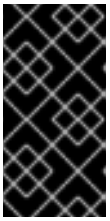


重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

11.2.12.1.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

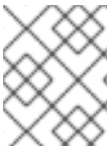
11.2.12.1.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

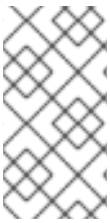
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**

- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

-

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードが使用されている場合の問題を回避するために、常にアクティブバックアップモードでオプション **fail_over_mac=1** を設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

11.2.12.2. PXE ブートを使用した RHCOS のインストール

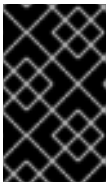
PXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

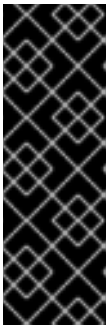
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



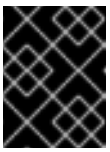
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーと **rootfs** ファイルを HTTP サーバーにアップロードします。
   - iPXE の場合、**kernel**、**initramfs**、および **rootfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージの PXE インストールを設定し、インストールを開始します。
   以下の例で示されるご使用の環境のメニューエントリーを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

3 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. PXE UEFI を使用する場合は、以下の操作を実行します。

   a. システムの起動に必要な **shimx64.efi** and **grubx64.efi** EFI バイナリーと **grub.cfg** ファイルを指定します。

      - ホストに RHCOS ISO をマウントしてから、**images/efiboot.img** ファイルをマウントし、必要な EFI バイナリーを展開します。

        ```
        $ mkdir -p /mnt/iso
        ```

        ```
        $ mkdir -p /mnt/efiboot
        ```

        ```
        $ mount -o loop rhcos-installer.x86_64.iso /mnt/iso
        ```

        ```
        $ mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
        ```

      - **efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

        ```
        $ cp /mnt/efiboot/EFI/redhat/shimx64.efi .
        ```

        ```
        $ cp /mnt/efiboot/EFI/redhat/grubx64.efi .
        ```

        ```
        $ umount /mnt/efiboot
        ```

        ```
        $ umount /mnt/iso
        ```

      - RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

   b. **grub.cfg** ファイルを編集し、以下のような引数を追加します。

      ```
      menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --class gnu --class os {
       linuxefi rhcos-<version>-live-kernel-<architecture> coreos.inst.install_dev=/dev/sda coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
      ```

```
<architecture>.img coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
 initrdefi rhcos-<version>-live-initramfs.<architecture>.img
}
```

詳細は以下のようになります。

rhcos-<version>-live-kernel-<architecture>

TFTP サーバーにアップロードした **kernel** ファイルを指定します。

http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img

HTTP サーバーにアップロードしたライブ rootfs イメージの場所を指定します。

http://<HTTP_server>/bootstrap.ign

HTTP サーバーにアップロードしたブートストラップ Ignition 設定ファイルの場所を指定します。

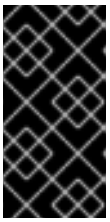
rhcos-<version>-live-initramfs.<architecture>.img

TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

UEFI ブート用に PXE サーバーを設定する方法は、Red Hat ナレッジベースの記事 How to configure/setup a PXE server for Red Hat Enterprise Linux? を参照してください。

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

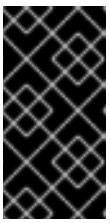


重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

11.2.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

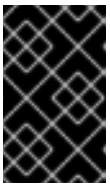
1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

11.2.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

11.2.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

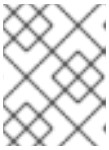
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

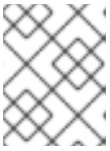
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

11.2.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.8.2     True        False         False     19m
baremetal                                  4.8.2     True        False         False     37m
cloud-credential                            4.8.2     True        False         False     40m
```

```
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

11.2.16.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

11.2.16.1.1. IBM Z の場合のレジストリーストレージの設定

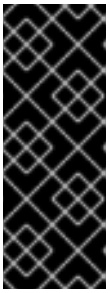
11.2.16.1.2. IBM Power Systems の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

- IBM Power Systems の IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。



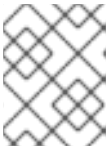
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

   - 以下を実行します。

     ```
     $ oc edit configs.imageregistry/cluster
     ```

     次に、行を変更します。

     ```
     managementState: Removed
     ```

     次のように変更してください。

     ```
     managementState: Managed
     ```

11.2.16.1.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

  ```
  $ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
  {"storage":{"emptyDir":{}}}}'
  ```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

11.2.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                               1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                               1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
Running    0          5m
...
```

–

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. マルチパスを有効にするための追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスのドキュメントの開始**を参照してください。

a. ブート一覧を表示し、システムが通常モードで起動した場合に使用可能なブートデバイスを指定するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o
sda
```

b. 通常モードのブート一覧を更新し、別のデバイス名を追加するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
sdc
sdd
sde
```

元のブートディスクパスがダウンすると、ノードは通常のブートデバイス一覧に登録された別のデバイスから再起動します。

c. すべてのワーカーノードが再起動します。プロセスを監視するには、以下のコマンドを入力します。

```
$ oc get nodes -w
```

注記

インフラストラクチャーノードなどの追加のマシンタイプがある場合は、これらのタイプについてこのプロセスを繰り返します。

4. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

11.2.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、ク

ラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

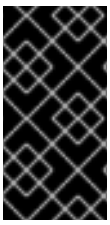
- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

11.2.19. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

11.3. ネットワークが制限された環境での IBM POWER SYSTEMS へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターをネットワークが制限された環境でプロビジョニングする IBM Power Systems インフラストラクチャーにインストールできます。

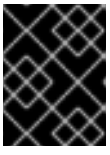


重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

11.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得している。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

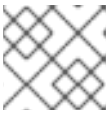
  

  重要

  インストールメディアにアクセスできるマシンでインストール手順が実行されるようにします。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。

- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

11.3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

11.3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

11.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

11.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

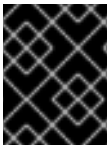
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

11.3.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表11.18 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

11.3.4.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表11.19 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	2	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	2	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

11.3.4.3. IBM Power Systems の最小要件

OpenShift Container Platform バージョン 4.8 は、以下の IBM ハードウェアにインストールできます。

- IBM POWER8 または POWER9 プロセッサーベースのシステム

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 IBM Power ベアメタルサーバーまたは 6 LPAR

オペレーティングシステム要件

- IBM POWER8 または POWER9 プロセッサーベースのシステムの 1 つインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 共有イーサネットアダプターを使用して仮想 I/O サーバーによって仮想化される

- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 100GB / 16GB
- OpenShift Container Platform コンピュートマシン用に 100 GB / 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 100 GB / 16 GB

11.3.4.4. 推奨される IBM Power システム要件

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 IBM Power ベアメタルサーバーまたは 6 LPAR

オペレーティングシステム要件

- IBM POWER8 または POWER9 プロセッサーベースのシステムの 1 つインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 共有イーサネットアダプターを使用して仮想 I/O サーバーによって仮想化される
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 120 GB / 32 GB
- OpenShift Container Platform コンピュートマシン用に 120 GB / 32 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 120 GB / 16 GB

11.3.4.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

11.3.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

11.3.4.6.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

表11.20 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット

プロトコル	ポート	説明
TCP/UDP	**30000-32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表11.21 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表11.22 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

11.3.4.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform ク

ラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表11.23 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コントロールプレーンマシン	**<master><n>. <cluster_name>. <base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

11.3.4.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例11.4 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
```

```
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

-

例11.5 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

11.3.4.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

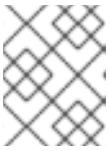
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表11.24 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

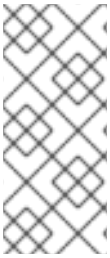
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表11.25 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

11.3.4.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例11.6 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
```

```
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

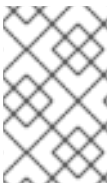
7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

11.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。

OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

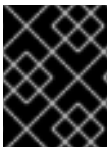
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

11.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1. **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

出力例

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

11.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

11.3.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

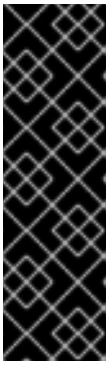
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

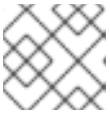
```
$ mkdir <installation_directory>
```



重要

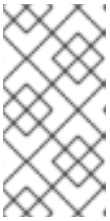
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

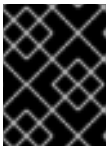
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

11.3.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

11.3.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表11.26 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

11.3.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表11.27 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

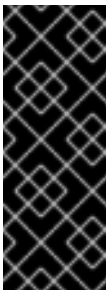
networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
	
	複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

11.3.8.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表11.28 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**

パラメーター	説明	値
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**

パラメーター	説明	値
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。

パラメーター	説明	値
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

11.3.8.2. IBM Z のサンプル install-config.yaml ファイル

11.3.8.3. IBM Power Systems のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
  architecture : ppc64le
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture : ppc64le
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
```

```
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。

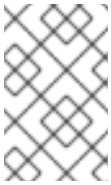
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。IBM Power Systems インフラストラクチャー

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

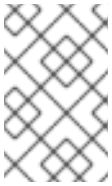
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

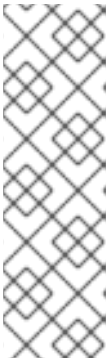
17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

11.3.8.4. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

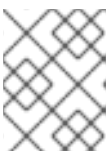
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

11.3.8.5. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

11.3.9. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

11.3.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表11.29 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

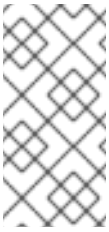
フィールド	タイプ	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表11.30 **defaultNetwork** オブジェクト

フィールド	タイプ	説明

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表11.31 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	タイプ	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表11.32 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表11.33 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表11.34 kubeProxyConfig オブジェクト

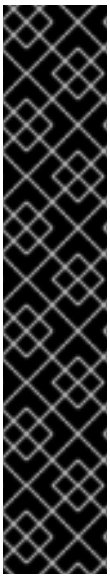
フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

11.3.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

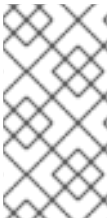
一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



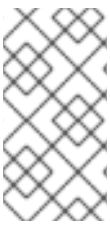
重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは ppc64le でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のフートストラップ、コントロールフレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

11.3.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Power Systems インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

ISO イメージまたはネットワーク PXE ブートを使用する手順を実行して RHCOS をマシンにインストールできます。

11.3.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

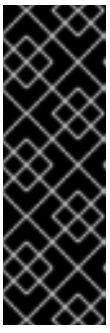
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージミラー RHCOS イメージミラー ミラーページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-
live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。

注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

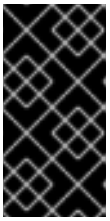
```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

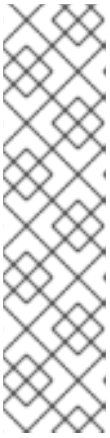
9. RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。
10. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

11.3.11.1.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

11.3.11.1.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

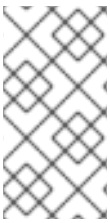
ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**

- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

-

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。**options** はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードが使用されている場合の問題を回避するために、常にアクティブバックアップモードでオプション **fail_over_mac=1** を設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces**は物理 (イーサネット) インターフェイス (**em1**、**em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

11.3.11.2. PXE ブートを使用した RHCOS のインストール

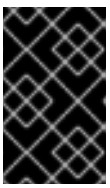
PXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+
(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```

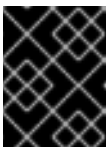
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーと **rootfs** ファイルを HTTP サーバーにアップロードします。
   - iPXE の場合、**kernel**、**initramfs**、および **rootfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージの PXE インストールを設定し、インストールを開始します。
   以下の例で示されるご使用の環境のメニューエントリーを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

3 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. PXE UEFI を使用する場合は、以下の操作を実行します。

   a. システムの起動に必要な **shimx64.efi** and **grubx64.efi** EFI バイナリーと **grub.cfg** ファイルを指定します。

      - ホストに RHCOS ISO をマウントしてから、**images/efiboot.img** ファイルをマウントし、必要な EFI バイナリーを展開します。

        ```
        $ mkdir -p /mnt/iso
        ```

        ```
        $ mkdir -p /mnt/efiboot
        ```

        ```
        $ mount -o loop rhcos-installer.x86_64.iso /mnt/iso
        ```

        ```
        $ mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
        ```

      - **efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

        ```
        $ cp /mnt/efiboot/EFI/redhat/shimx64.efi .
        ```

        ```
        $ cp /mnt/efiboot/EFI/redhat/grubx64.efi .
        ```

        ```
        $ umount /mnt/efiboot
        ```

        ```
        $ umount /mnt/iso
        ```

      - RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

   b. **grub.cfg** ファイルを編集し、以下のような引数を追加します。

      ```
      menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --class gnu --class os {
       linuxefi rhcos-<version>-live-kernel-<architecture> coreos.inst.install_dev=/dev/sda coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
      ```

```
<architecture>.img coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
 initrdefi rhcos-<version>-live-initramfs.<architecture>.img
}
```

詳細は以下のようになります。

rhcos-<version>-live-kernel-<architecture>

TFTP サーバーにアップロードした **kernel** ファイルを指定します。

http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img

HTTP サーバーにアップロードしたライブ rootfs イメージの場所を指定します。

http://<HTTP_server>/bootstrap.ign

HTTP サーバーにアップロードしたブートストラップ Ignition 設定ファイルの場所を指定します。

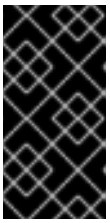
rhcos-<version>-live-initramfs.<architecture>.img

TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

UEFI ブート用に PXE サーバーを設定する方法は、Red Hat ナレッジベースの記事 How to configure/setup a PXE server for Red Hat Enterprise Linux? を参照してください。

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

11.3.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

11.3.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

11.3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

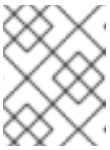
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

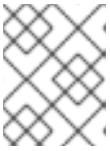
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

11.3.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.8.2     True        False         False     19m
baremetal                                  4.8.2     True        False         False     37m
cloud-credential                            4.8.2     True        False         False     40m
```

```
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

11.3.15.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

11.3.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

11.3.15.2.1. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、イメージレジストリー Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **ManagementState** イメージレジストリー Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

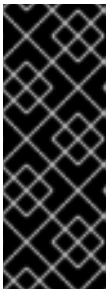
11.3.15.2.2. IBM Z の場合のレジストリーストレージの設定

11.3.15.2.3. IBM Power Systems の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Power Systems の IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。



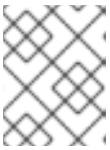
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

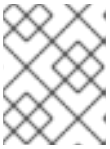
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

11.3.15.2.4. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

11.3.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running    0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. マルチパスを有効にするための追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスのドキュメントの開始**を参照してください。

a. ブート一覧を表示し、システムが通常モードで起動した場合に使用可能なブートデバイスを指定するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o
sda
```

b. 通常モードのブート一覧を更新し、別のデバイス名を追加するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
sdc
sdd
sde
```

元のブートディスクパスがダウンすると、ノードは通常のブートデバイス一覧に登録された別のデバイスから再起動します。

c. すべてのワーカーノードが再起動します。プロセスを監視するには、以下のコマンドを入力します。

```
$ oc get nodes -w
```

注記

インフラストラクチャーノードなどの追加のマシンタイプがある場合は、これらのタイプについてこのプロセスを繰り返します。

4. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

5. Cluster registration ページでクラスターを登録します。

11.3.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

11.3.18. 次のステップ

- クラスターをカスタマイズします。

- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。

第12章 OPENSTACK へのインストール

12.1. OPENSTACK へのインストールの準備

12.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

12.1.2. OpenStack に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

12.1.2.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる Red Hat OpenStack Platform (RHOSP) インフラストラクチャーに、クラスターをインストールできます。

- **カスタマイズによる OpenStack へのクラスターのインストール**: カスタマイズされたクラスターを RHOSP にインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の数多くのカスタマイズオプションは、インストール後 に利用できます。
- **Kuryr を使用した OpenStack へのクラスターのインストール**: Kuryr SDN を使用する RHOSP にカスタマイズされた OpenShift Container Platform クラスターをインストールできます。Kuryr と OpenShift Container Platform の統合は主に、RHOSP の仮想マシンで実行する OpenShift Container Platform クラスター用に設計されました。Kuryr は、OpenShift Container Platform Pod を RHOSP SDN にプラグインしてネットワークのパフォーマンスを強化します。さらに、これは Pod と RHOSP 仮想インスタンス間の接続を可能にします。
- **ネットワークが制限された環境での OpenStack へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを作成して、OpenShift Container Platform をネットワークが制限された環境またはネットワークの非接続環境で RHOSP にインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

12.1.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、独自にプロビジョニングする RHOSP インフラストラクチャーにクラスターをインストールできます。

- **独自のインフラストラクチャーでの OpenStack へのクラスターのインストール**: ユーザーによってプロビジョニングされる RHOSP インフラストラクチャーに OpenShift Container Platform をインストールできます。このインストール方法を使用して、クラスターを既存のインフラストラクチャーおよび変更と統合できます。ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの場合、Nova サーバー、Neutron ポート、セキュリティーグループなどの RHOSP リソースをすべて作成する必要があります。提供される Ansible Playbook を使用してデプロイメントプロセスを支援することができます。

- **Kuryr を使用した独自のインフラストラクチャーの OpenStack へのクラスターのインストール**: Kuryr SDN を使用するユーザーによってプロビジョニングされる RHOSP インフラストラクチャーに OpenShift Container Platform をインストールできます。

- **独自の SR-IOV インフラストラクチャーの OpenStack へのクラスターのインストール**: コンピュートマシンの実行に Single-root Input/Output Virtualization (SR-IOV) ネットワークを使用するユーザーによってプロビジョニングされる RHOSP インフラストラクチャーに、OpenShift Container Platform をインストールできます。

12.2. カスタマイズによる OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、Red Hat OpenStack Platform (RHOSP) にカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に **install-config.yaml** でパラメーターを変更します。

12.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

- OpenShift クラスターでサポートされるプラットフォーム のセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認している。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。

- ブロックストレージ (Cinder) またはオブジェクトストレージ (Swift) などのストレージサービスが RHOSP にインストールされている。オブジェクトストレージは、OpenShift Container Platform レジストリークラスターデプロイメントに推奨されるストレージ技術です。詳細は、ストレージの最適化 を参照してください。

- RHOSP でメタデータサービスが有効化されている。

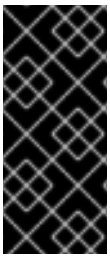
12.2.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表12.1 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

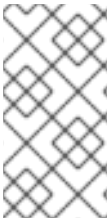
リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

12.2.2.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.2.2.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

12.2.2.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.2.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.2.4. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。

重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

12.2.5. RHOSP で実行されるクラスター上のカスタムストレージを使用したイメージレジストリーの設定

Red Hat OpenStack Platform (RHOSP) にクラスターをインストールした後に、特定のアベイラビリティーゾーンにある Cinder ボリュームをレジストリーストレージとして使用できます。

手順

1. YAML ファイルを作成して、使用するストレージクラスとアベイラビリティーゾーンを指定します。以下に例を示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: custom-csi-storageclass
provisioner: cinder.csi.openstack.org
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  availability: <availability_zone_name>
```

注記

OpenShift Container Platform では、選択したアベイラビリティーゾーンが存在するかどうかは確認されません。設定を適用する前に、アベイラビリティーゾーンの名前を確認してください。

2. コマンドラインから設定を適用します。

```
$ oc apply -f <storage_class_file_name>
```

出力例

```
storageclass.storage.k8s.io/custom-csi-storageclass created
```

3. ストレージクラスと **openshift-image-registry** namespace を使用する永続ボリュームクレーム (PVC) を指定する YAML ファイルを作成します。以下に例を示します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-imageregistry
  namespace: openshift-image-registry (1)
  annotations:
    imageregistry.openshift.io: "true"
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 100Gi (2)
  storageClassName: <your_custom_storage_class> (3)
```

(1) **openshift-image-registry** namespace を入力します。この namespace により、クラスターイメージレジストリーオペレーターは PVC を使用できます。

(2) オプション: ボリュームサイズを調整します。

(3) 作成されるストレージクラスの名前を入力します。

4. コマンドラインから設定を適用します。

```
$ oc apply -f <pvc_file_name>
```

出力例

```
persistentvolumeclaim/csi-pvc-imageregistry created
```

5. イメージレジストリー設定の元の永続ボリューム要求は、新しい要求に置き換えます。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type 'json' -p='[{"op":
"replace", "path": "/spec/storage/pvc/claim", "value": "csi-pvc-imageregistry"}]'
```

出力例

```
config.imageregistry.operator.openshift.io/cluster patched
```

数分すると、設定が更新されます。

検証

レジストリーが定義したリソースを使用していることを確認するには、以下を実行します。

1. PVC クレーム値が PVC 定義で指定した名前と同じであることを確認します。

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

出力例

```
...
status:
    ...
    managementState: Managed
    pvc:
      claim: csi-pvc-imageregistry
...
```

2. PVC のステータスが **Bound** であることを確認します。

```
$ oc get pvc -n openshift-image-registry csi-pvc-imageregistry
```

出力例

```
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS            AGE
csi-pvc-imageregistry   Bound    pvc-72a8f9c9-f462-11e8-b6b6-fa163e18b7b5   100Gi
RWO            custom-csi-storageclass   11m
```

12.2.6. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+---------------+-------------+
| ID                                   | Name          | Router Type |
+--------------------------------------+---------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+---------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

重要

外部ネットワークの CIDR 範囲がデフォルトのネットワーク範囲のいずれかと重複している場合、インストールプロセスを開始する前に、**install-config.yaml** ファイルで一致するネットワーク範囲を変更する必要があります。

デフォルトのネットワーク範囲は以下のとおりです。

ネットワーク	範囲
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

警告

インストールプログラムにより同じ名前を持つ複数のネットワークが見つかる場合、それらのネットワークのいずれかがランダムに設定されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

注記

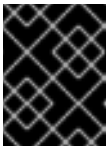
Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

12.2.7. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: 'devuser'
           password: XXX
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。
   a. 認証局ファイルをマシンにコピーします。
   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

     ```
     clouds:
       shiftstack:
         ...
     ```

```
cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

ヒント

カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。
   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値
   b. 現行ディレクトリー
   c. Unix 固有のユーザー設定ディレクトリー (例: ~/**.config/openstack/clouds.yaml**)
   d. Unix 固有のサイト設定ディレクトリー (例: /**etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.2.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.2.9. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

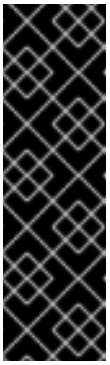
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

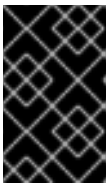
vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

関連情報

使用可能なパラメーターの詳細については、**インストール設定パラメーター セクション** を参照してください。

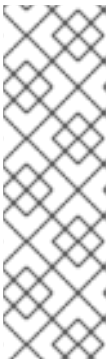
12.2.9.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対す

る呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

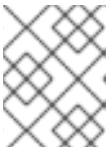
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

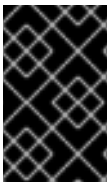
cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

12.2.10. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.2.10.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.2 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.2.10.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.3 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

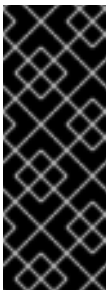
networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
	
	複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

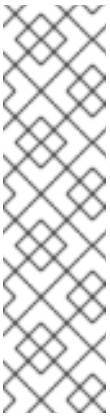
12.2.10.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.4 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

12.2.10.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表12.5 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。

パラメーター	説明	値
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

12.2.10.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表12.6 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
compute.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
compute.platform.openstack.rootVolume.zones	コンピュートマシンの root ボリュームをインストールするアベイラビリティーゾーン。このパラメーターに値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
controlPlane.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.rootVolume.zones	コントロールプレーンマシンの root ボリュームをインストールするアベイラビリティーゾーン。この値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform.openstack.clusterOSImageProperties	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアの一覧。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.ingressFloatingIP	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform.openstack.apiFloatingIP	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

12.2.10.6. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

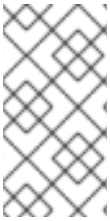
カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。

- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。



注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

12.2.10.7. ベアメタルマシンを使用したクラスターのデプロイ

クラスターでベアメタルマシンを使用する必要がある場合は、**install-config.yaml** ファイルを変更します。クラスターには、ベアメタル上でコントロールプレーンとコンピュートマシンの両方を実行させることも、コンピュートマシンのみを実行させることもできます。

ベアメタルコンピュートマシンは、Kuryr を使用するクラスターではサポートされません。

注記

install-config.yaml ファイルで、ベアメタルワーカーに使用する RHOSP ネットワークが Floating IP アドレスをサポートするかどうかが反映されていることを確認します。

前提条件

- RHOSP の ベアメタルサービス (Ironic) は有効にされており、RHOSP Compute API でアクセスできる。
- ベアメタルは RHOSP フレーバー として利用可能である。
- RHOSP ネットワークは、仮想マシンとベアメタルサーバー接続の両方をサポートする。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。
- マシンを既存のネットワークにデプロイする必要がある場合、RHOSP サブネットがプロビジョニングされる。

- マシンをインストーラーでプロビジョニングされるネットワークに場合、RHOSP ベアメタルサービス (Ironic) はテナントネットワークで実行される Preboot eXecution Environment (PXE) ブートマシンをリッスンし、これと対話できます。
- **install-config.yaml** ファイルを OpenShift Container Platform インストールプログラムの一部として作成している。

手順

1. **install-config.yaml** ファイルで、マシンのフレーバーを編集します。
   a. ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、**controlPlane.platform.openstack.type** の値をベアメタルフレーバーに変更します。
   b. **compute.platform.openstack.type** の値をベアメタルフレーバーに変更します。
   c. 既存のネットワークにマシンをデプロイする場合は、**platform.openstack.machinesSubnet** の値をネットワークの RHOSP サブネット UUID に変更します。コントロールプレーンおよびコンピュートマシンは同じサブネットを使用する必要があります。

   ベアメタルの install-config.yaml のサンプルファイル

```
controlPlane:
    platform:
      openstack:
        type: <bare_metal_control_plane_flavor> 1
...

compute:
  - architecture: amd64
    hyperthreading: Enabled
    name: worker
    platform:
      openstack:
        type: <bare_metal_compute_flavor> 2
    replicas: 3
...

platform:
    openstack:
      machinesSubnet: <subnet_UUID> 3
...
```

   1. ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、この値をベアメタルのフレーバーに変更します。
   2. この値を、コンピュートマシンに使用するベアメタルのフレーバーに変更します。
   3. 既存のネットワークを使用する必要がある場合は、この値を RHOSP サブネットの UUID に変更します。

更新された **install-config.yaml** ファイルを使用してインストールプロセスを完了します。デプロイメント時に作成されるコンピュートマシンは、ファイルに追加したフレーバーを使用します。

注記

インストーラーは、ベアメタルマシンの起動中にタイムアウトする可能性があります。

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

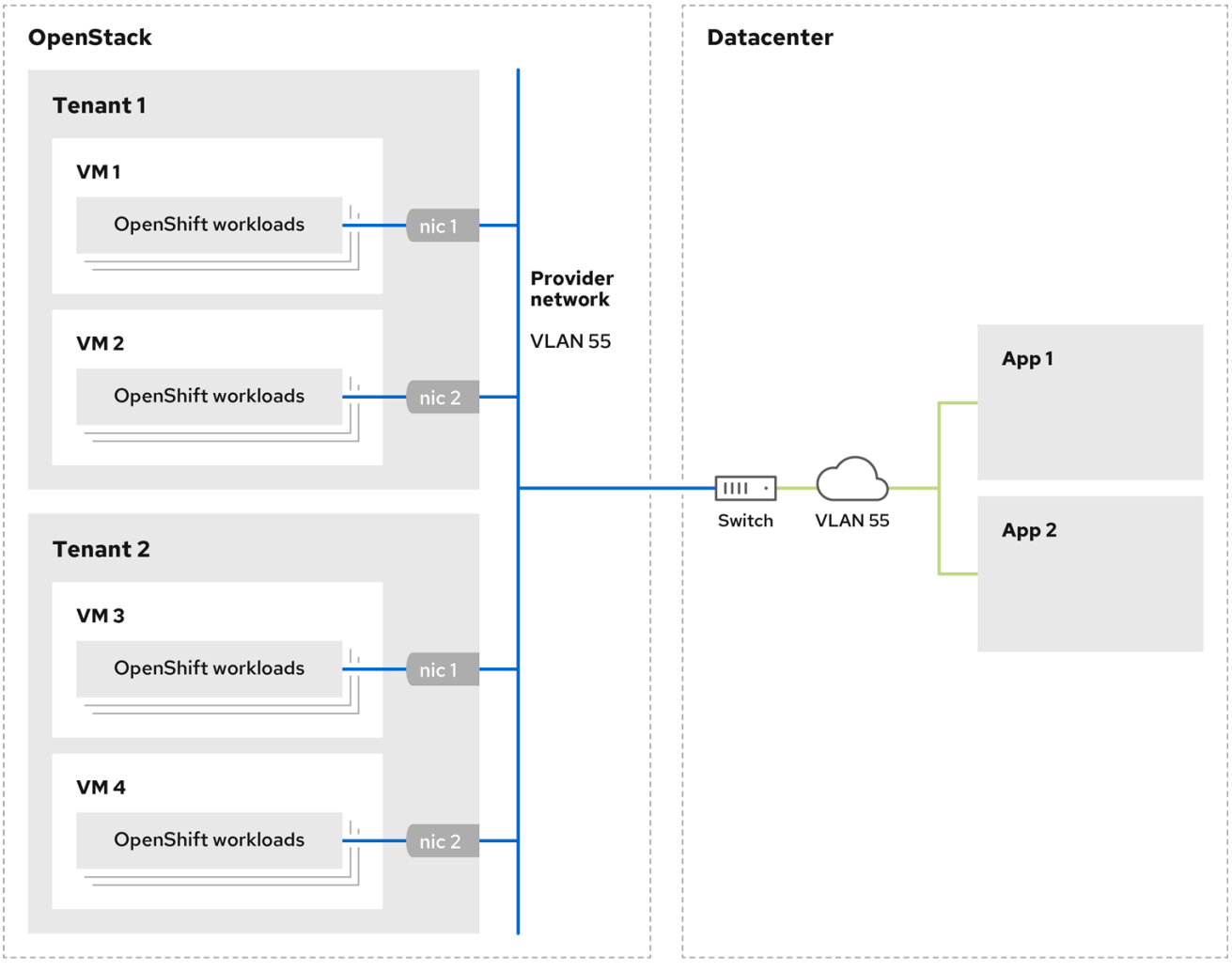
```
./openshift-install wait-for install-complete --log-level debug
```

12.2.10.8. RHOSP プロバイダーネットワーク上のクラスターデプロイメント

プロバイダーネットワーク上のプライマリーネットワークインターフェイスを使用して、OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイできます。プロバイダーネットワークは一般的に、インターネットへの到達に使用可能なパブリックネットワークに、プロジェクトが直接アクセスできるように使用します。ネットワーク作成プロセスの一環として、プロバイダーネットワークをプロジェクト間で共有することもできます。

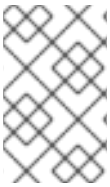
RHOSP プロバイダーネットワークは、データセンター内の既存の物理ネットワークに直接マップします。RHOSP 管理者はこれらを作成する必要があります。

以下の例では、OpenShift Container Platform ワークロードはプロバイダーネットワークを使用してデータセンターに接続されます。



プロバイダーネットワークにインストールされている OpenShift Container Platform クラスターは、テナントネットワークまたは Floating IP アドレスを必要としません。インストーラーは、インストール中にこれらのリソースを作成しません。

プロバイダーネットワークタイプの例には、フラット (タグなし) および VLAN (802.1Q タグ付き) が含まれます。



注記

クラスターは、ネットワークタイプが許可する限り多くのプロバイダーネットワーク接続をサポートできます。たとえば、VLAN ネットワークは、通常最大 4096 の接続をサポートします。

プロバイダーネットワークおよびテナントネットワークの詳細は、RHOSP のドキュメント を参照してください。

12.2.10.8.1. クラスターのインストールにおける RHOSP プロバイダーネットワーク要件

OpenShift Container Platform クラスターをインストールする前に、Red Hat OpenStack Platform (RHOSP) のデプロイメントおよびプロバイダーネットワークは、さまざまな条件を満たす必要があります。

- RHOSP ネットワークサービス (Neutron) が有効化され、RHOSP ネットワーク API 経由でアクセス可能であること。
- RHOSP ネットワークサービスでは、ポートセキュリティーと許可するアドレスペアの機能拡張が有効化 されていること。
- プロバイダーネットワークは他のテナントと共有できます。

  ヒント

  --share フラグを指定して **openstack network create** コマンドを使用して、共有できるネットワークを作成します。

- クラスターのインストールに使用する RHOSP プロジェクトは、プロバイダーネットワークと適切なサブネットを所有する必要があります。

  ヒント

  openshift という名前のプロジェクトのネットワークを作成するには、以下のコマンドを入力します。

  ```
  $ openstack network create --project openshift
  ```

  openshift という名前のプロジェクトのサブネットを作成するには、以下のコマンドを入力します。

  ```
  $ openstack subnet create --project openshift
  ```

  RHOSP でのネットワークの作成に関する詳細は、プロバイダーネットワークに関するドキュメント を参照してください。

クラスターが **admin** ユーザーによって所有されている場合、そのユーザーとしてインストーラーを実行してネットワーク上でポートを作成する必要があります。

重要

プロバイダーネットワークは、クラスターの作成に使用される RHOSP プロジェクトによって所有されている必要があります。所有されていない場合は、RHOSP Compute サービス (Nova) はそのネットワークからポートを要求できません。

- プロバイダーネットワークが、デフォルトで **169.254.169.254** である RHOSP メタデータサービスの IP アドレスに到達できることを確認します。
RHOSP SDN とネットワークサービス設定によっては、サブネットを作成する際に、ルートを提供しなければならない場合があります。以下に例を示します。

```
$ openstack subnet create --dhcp --host-route
destination=169.254.169.254/32,gateway=192.0.2.2 ...
```

- オプション: ネットワークのセキュリティーを保護するには、単一のプロジェクトへのネットワークアクセスを制限する ロールベースのアクセス制御 (RBAC) ルールを作成します。

12.2.10.8.2. プロバイダーネットワークにプライマリーインターフェイスを持つクラスターのデプロイ

Red Hat OpenStack Platform (RHOSP) プロバイダーネットワーク上にプライマリーネットワークインターフェイスを持つ OpenShift Container Platform クラスターをデプロイすることができます。

- クラスターのインストールにおける RHOSP プロバイダーネットワーク要件に記載されているとおりに、お使いの Red Hat OpenStack Platform (RHOSP) のデプロイメントが設定されています。

手順

1. テキストエディターで **install-config.yaml** ファイルを開きます。
2. **platform.openstack.apiVIP** プロパティーの値を API VIP の IP アドレスに設定します。
3. **platform.openstack.ingressVIP** プロパティーの値を Ingress VIP の IP アドレスに設定します。
4. **platform.openstack.machinesSubnet** プロパティーの値をプロバイダーネットワークサブネットの UUID に設定します。
5. **networking.machineNetwork.cidr** プロパティーの値をプロバイダーネットワークサブネットの CIDR ブロックに設定します。

重要

platform.openstack.apiVIP プロパティーおよび **platform.openstack.ingressVIP** プロパティーはいずれも、**networking.machineNetwork.cidr** ブロックから割り当てられていない IP アドレスである必要があります。

RHOSP プロバイダーネットワークに依存するクラスターのインストール設定ファイルのセクション

-

```
    ...
    platform:
      openstack:
        apiVIP: 192.0.2.13
        ingressVIP: 192.0.2.23
        machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
        # ...
    networking:
      machineNetwork:
      - cidr: 192.0.2.0/24
```

警告

プライマリーネットワークインターフェイスにプロバイダーネットワークを使用している間は、**platform.openstack.externalNetwork** パラメーターまたは **platform.openstack.externalDNS** パラメーターを設定することはできません。

クラスターをデプロイする際に、インストーラーは **install-config.yaml** ファイルを使用してプロバイダーネットワークにクラスターをデプロイします。

ヒント

プロバイダーネットワークを含むネットワークを **platform.openstack.additionalNetworkIDs** 一覧に追加できます。

クラスターのデプロイ後に、Pod を追加のネットワークに接続することができます。詳細は、複数ネットワークについて を参照してください。

12.2.10.9. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
```

```
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

12.2.11. コンピュートマシンのアフィニティーの設定

オプションで、インストール時にコンピュートマシンのアフィニティーポリシーを設定できます。インストーラーは、デフォルトでコンピュートマシンのアフィニティーポリシーを選択しません。

インストール後に特定の RHOSP サーバーグループを使用するマシンセットを作成することもできます。

注記

コントロールプレーンマシンは、**soft-anti-affinity** ポリシーで作成されます。

ヒント

RHOSP インスタンスのスケジューリングおよび配置 の詳細は、RHOSP のドキュメントを参照してください。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. RHOSP コマンドラインインターフェイスを使用して、コンピュートマシンのサーバーグループを作成します。以下に例を示します。

```
$ openstack \
    --os-compute-api-version=2.15 \
    server group create \
    --policy anti-affinity \
    my-openshift-worker-group
```

詳細は、 **server group create** コマンドのドキュメント を参照してください。

2. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

ここでは、以下のようになります。

installation_directory

クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

3. **MachineSet** 定義ファイルの **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** を作成します。

4. **spec.template.spec.providerSpec.value** プロパティーの下にある定義に、プロパティー **serverGroupID** を追加します。以下に例を示します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_ID>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (1)
          kind: OpenstackProviderSpec
          networks:
          - filter: {}
            subnets:
```

```
    - filter:
        name: <subnet_name>
        tags: openshiftClusterID=<infrastructure_ID>
  securityGroups:
  - filter: {}
    name: <infrastructure_ID>-<node_role>
  serverMetadata:
    Name: <infrastructure_ID>-<node_role>
    openshiftClusterID: <infrastructure_ID>
  tags:
  - openshiftClusterID=<infrastructure_ID>
  trunk: true
  userDataSecret:
    name: <node_role>-user-data
  availabilityZone: <optional_openstack_availability_zone>
```

1 サーバーグループの UUID をここに追加します。

5. オプション: **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

クラスターのインストール時に、インストーラーは変更した **MachineSet** 定義を使用して RHOSP サーバーグループ内にコンピュートマシンを作成します。

12.2.12. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 /**openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name>
```

1

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.2.13. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.2.13.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API およびクラスターアプリケーションへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

4. FIP を、以下のパラメーターの値として **install-config.yaml** ファイルに追加します。
    - **platform.openstack.ingressFloatingIP**
    - **platform.openstack.apiFloatingIP**

これらの値を使用する場合には、**install-config.yaml** ファイルの **platform.openstack.externalNetwork** パラメーターの値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.2.13.2. Floating IP アドレスなしでのインストールの完了

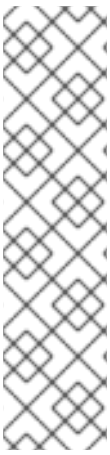
Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

install-config.yaml ファイルで以下のパラメーターを定義しないでください。

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

外部ネットワークを提供できない場合は、**platform.openstack.externalNetwork** を空白のままにすることもできます。**platform.openstack.externalNetwork** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。外部接続を独自に設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムからインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。



注記

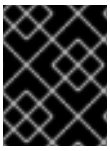
API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.2.14. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

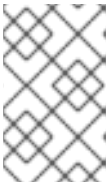
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



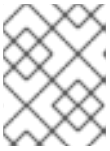
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

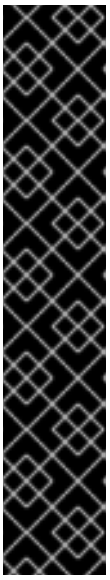
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

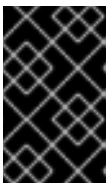
クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

12.2.15. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

12.2.16. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

12.2.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.2.18. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.3. KURYR を使用する OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、Kuryr SDN を使用する Red Hat OpenStack Platform (RHOSP) にカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に **install-config.yaml** でパラメーターを変更します。

12.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- OpenShift クラスターでサポートされるプラットフォーム のセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認している。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ブロックストレージ (Cinder) またはオブジェクトストレージ (Swift) などのストレージサービスが RHOSP にインストールされている。オブジェクトストレージは、OpenShift Container Platform レジストリークラスターデプロイメントに推奨されるストレージ技術です。詳細は、ストレージの最適化 を参照してください。

12.3.2. Kuryr SDN について

Kuryr は、 Neutron および Octavia Red Hat OpenStack Platform (RHOSP) サービスを使用して Pod およびサービスのネットワークを提供する Container Network Interface (CNI) プラグインです。

Kuryr と OpenShift Container Platform の統合は主に、RHOSP の仮想マシンで実行する OpenShift Container Platform クラスター用に設計されました。Kuryr は、OpenShift Container Platform Pod を RHOSP SDN にプラグインしてネットワークのパフォーマンスを強化します。さらに、これは Pod と RHOSP 仮想インスタンス間の接続を可能にします。

Kuryr コンポーネントは **openshift-kuryr** namespace を使用して OpenShift Container Platform の Pod としてインストールされます。

- **kuryr-controller**: **master** ノードにインストールされる単一のサービスインスタンスです。これは、OpenShift Container Platform で **Deployment** としてモデリングされます。
- **kuryr-cni**: 各 OpenShift Container Platform ノードで Kuryr を CNI ドライバーとしてインストールし、設定するコンテナーです。これは、OpenShift Container Platform で **DaemonSet** オブジェクトとしてモデリングされます。

Kuryr コントローラーは OpenShift Container Platform API サーバーで Pod、サービスおよび namespace の作成、更新、および削除イベントについて監視します。これは、OpenShift Container Platform API 呼び出しを Neutron および Octavia の対応するオブジェクトにマップします。そのため、Neutron トランクポート機能を実装するすべてのネットワークソリューションを使用して、Kuryr 経由で OpenShift Container Platform をサポートすることができます。これには、Open vSwitch (OVS) および Open Virtual Network (OVN) などのオープンソースソリューションや Neutron と互換性のある市販の SDN が含まれます。

Kuryr は、カプセル化された RHOSP テナントネットワーク上の OpenShift Container Platform デプロイメントに使用することが推奨されています。これは、 RHOSP ネットワークでカプセル化された OpenShift Container Platform SDN を実行するなど、二重のカプセル化を防ぐために必要です。

プロバイダーネットワークまたはテナント VLAN を使用する場合は、二重のカプセル化を防ぐために Kuryr を使用する必要はありません。パフォーマンス上の利点はそれほど多くありません。ただし、設定によっては、Kuryr を使用して 2 つのオーバーレイが使用されないようにすることには利点がある場合があります。

Kuryr は、以下のすべての基準が true であるデプロイメントでは推奨されません。

- RHOSP のバージョンが 16 よりも前のバージョンである。
- デプロイメントで UDP サービスが使用されているか、または少数のハイパーバイザーで多数の TCP サービスが使用されている。

または、以下を実行します。

- **ovn-octavia** Octavia ドライバーが無効にされている。
- デプロイメントで、少数のハイパーバイザーで多数の TCP サービスが使用されている。

12.3.3. Kuryr を使用して OpenShift Container Platform を RHOSP にインストールするためのリソースのガイドライン

Kuryr SDN を使用する場合、Pod、サービス、namespace およびネットワークポリシーは RHOSP クォータのリソースを使用します。これにより、最小要件が増加します。また、Kuryr にはデフォルトインストールに必要な要件以外の追加要件があります。

以下のクォータを使用してデフォルトのクラスターの最小要件を満たすようにします。

表12.7 Kuryr を使用する RHOSP のデフォルト OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3: LoadBalancer タイプに予想されるサービス数
ポート	1500: Pod ごとに 1 つ必要
ルーター	1
サブネット	250: namespace/プロジェクトごとに 1 つ必要
ネットワーク	250: namespace/プロジェクトごとに 1 つ必要
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	250: サービスおよび NetworkPolicy ごとに 1 つ必要
セキュリティーグループルール	1000
ロードバランサー	100: サービスごとに 1 つ必要
ロードバランサーリスナー	500: サービスで公開されるポートごとに 1 つ必要
ロードバランサーノード	500: サービスで公開されるポートごとに 1 つ必要

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

重要

OVN Octavia ドライバーではなく Amphora ドライバーで Red Hat OpenStack Platform(RHOSP) バージョン 16 を使用している場合、セキュリティーグループはユーザープロジェクトではなくサービスアカウントに関連付けられます。

リソースを設定する際には、以下の点に注意してください。

- 必要なポート数は Pod 数よりも大きくなる。Kuryr はポートプールを使用して、事前に作成済みのポートを Pod で使用できるようにし、Pod の起動時間を短縮します。
- 各ネットワークポリシーは RHOSP セキュリティーグループにマップされ、**NetworkPolicy** 仕様によっては 1 つ以上のルールがセキュリティーグループに追加される。
- 各サービスは RHOSP ロードバランサーにマップされる。クォータに必要なセキュリティーグループの数を見積もる場合には、この要件を考慮してください。
  RHOSP バージョン 15 以前のバージョン、または **ovn-octavia driver** を使用している場合、各ロードバランサーにはユーザープロジェクトと共にセキュリティーグループがあります。
- クォータはロードバランサーのリソース (VM リソースなど) を考慮しませんが、RHOSP デプロイメントのサイズを決定する際にはこれらのリソースを考慮する必要があります。デフォルトのインストールには 50 を超えるロードバランサーが あり、クラスターはそれらのロードバランサーに対応できる必要があります。
  OVN Octavia ドライバーを有効にして RHOSP バージョン 16 を使用している場合は、1 つのロードバランサー仮想マシンのみが生成され、サービスは OVN フロー経由で負荷分散されます。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

Kuryr SDN を有効にするには、使用する環境が以下の要件を満たしている必要があります。

- RHOSP 13+ を実行します。
- オーバークラウドと Octavia を使用します。
- Neutron トランクポートの拡張を使用します。
- ML2/OVS Neutron ドライバーが **ovs-hybrid** の代わりに使用れる場合、**openvswitch** ファイアウォールドライバーを使用します。

12.3.3.1. クォータの拡大

Kuryr SDN を使用する場合、Pod、サービス、namespace、およびネットワークポリシーが使用する Red Hat OpenStack Platform (RHOSP) リソースに対応するためにクォータを引き上げる必要があります。

手順

- 以下のコマンドを実行して、プロジェクトのクォータを増やします。

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets
250 --networks 250 <project>
```

12.3.3.2. Neutron の設定

Kuryr CNI は Neutron トランクの拡張を使用してコンテナーを Red Hat OpenStack Platform (RHOSP) SDN にプラグインします。したがって、Kuryr が適切に機能するには **trunks** 拡張を使用する必要があります。

さらにデフォルトの ML2/OVS Neutron ドライバーを使用する場合には、セキュリティーグループがトランクサブポートで実行され、Kuryr がネットワークポリシーを適切に処理できるように、**ovs_hybrid** ではなく **openvswitch** に設定される必要があります。

12.3.3.3. Octavia の設定

Kuryr SDN は Red Hat OpenStack Platform (RHOSP) の Octavia LBaaS を使用して OpenShift Container Platform サービスを実装します。したがって、Kuryr SDN を使用するように RHOSP に Octavia コンポーネントをインストールし、設定する必要があります。

Octavia を有効にするには、Octavia サービスを RHOSP オーバークラウドのインストール時に組み込むか、またはオーバークラウドがすでに存在する場合は Octavia サービスをアップグレードする必要があります。Octavia を有効にする以下の手順は、オーバークラウドのクリーンインストールまたはオーバークラウドの更新の両方に適用されます。

注記

以下の手順では、Octavia を使用する場合に RHOSP のデプロイメント 時に必要となる主な手順のみを説明します。また、レジストリーメソッド が変更されることにも留意してください。

以下の例では、ローカルレジストリーの方法を使用しています。

手順

1. ローカルレジストリーを使用している場合、イメージをレジストリーにアップロードするためのテンプレートを作成します。以下に例を示します。

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{product-version} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. **local_registry_images.yaml** ファイルに Octavia イメージが含まれることを確認します。以下に例を示します。

```
...
```

```
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-
45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```

注記

Octavia コンテナーのバージョンは、インストールされている特定の RHOSP リリースによって異なります。

3. コンテナーイメージを **registry.redhat.io** からアンダークラウドノードにプルします。

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

これには、ネットワークおよびアンダークラウドディスクの速度に応じて多少の時間がかかる可能性があります。

4. Octavia ロードバランサーは OpenShift Container Platform API にアクセスするために使用されるため、それらのリスナーの接続のデフォルトタイムアウトを増やす必要があります。デフォルトのタイムアウトは 50 秒です。以下のファイルをオーバークラウドのデプロイコマンドに渡し、タイムアウトを 20 分に増やします。

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```

注記

これは RHOSP 13.0.13+ では不要です。

5. Octavia を使用してオーバークラウドをインストールまたは更新します。

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

注記

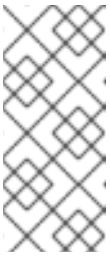
このコマンドには、Octavia に関連付けられたファイルのみが含まれます。これは、RHOSP の特定のインストールによって異なります。詳細は RHOSP のドキュメントを参照してください。Octavia インストールのカスタマイズについての詳細は、Octavia デプロイメントのプランニング を参照してください。

注記

Kuryr SDN を利用する際には、オーバークラウドのインストールに Neutron の **trunk** 拡張機能が必要です。これは、Director デプロイメントでデフォルトで有効にされます。Neutron バックエンドが ML2/OVS の場合、デフォルトの **ovs-hybrid** の代わりに **openvswitch** ファイアウォールを使用します。バックエンドが ML2/OVN の場合には変更の必要がありません。

6. RHOSP の 13.0.13 よりも前のバージョンでは、プロジェクトの作成後にプロジェクト ID を **octavia.conf** 設定ファイルに追加します。

- トラフィックが Octavia ロードバランサーを通過する場合など、複数のサービス全体でネットワークポリシーを実行するには、Octavia がユーザープロジェクトで Amphora 仮想マシンセキュリティーグループを作成するようにする必要があります。
この変更により、必要なロードバランサーのセキュリティーグループがそのプロジェクトに属し、それらをサービスの分離を実行するように更新できます。



注記

RHOSP の 13.0.13 よりも後のバージョンでは、このタスクは必要ありません。

Octavia は、ロードバランサー VIP へのアクセスを制限する新しい ACL API を実装します。

a. プロジェクト ID を取得します。

```
$ openstack project show <project>
```

出力例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | PROJECT_ID                       |
| is_domain   | False                            |
| name        | *<project>*                      |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

b. プロジェクト ID をコントローラーの **octavia.conf** に追加します。

i. **stackrc** ファイルを取得します。

```
$ source stackrc  # Undercloud credentials
```

ii. オーバークラウドコントローラーを一覧表示します。

```
$ openstack server list
```

出力例

```
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| ID                                   | Name        | Status | Networks
| Image         | Flavor     |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0    | ACTIVE |
ctlplane=192.168.24.6 | overcloud-full | compute    |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
```

iii. コントローラーに対して SSH を実行します。

```
$ ssh heat-admin@192.168.24.8
```

iv. **octavia.conf** ファイルを編集して、プロジェクトを Amphora セキュリティーグループがユーザーのアカウントに設定されているプロジェクトの一覧に追加します。

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

c. 新しい設定が読み込まれるように Octavia ワーカーを再起動します。

```
controller-0$ sudo docker restart octavia_worker
```

注記

RHOSP 環境によっては、Octavia は UDP リスナーをサポートしない可能性があります。RHOSP の 13.0.13 よりも前のバージョンで Kuryr SDN を使用する場合、UDP サービスはサポートされません。RHOSP バージョン 16 以降は UDP をサポートします。

12.3.3.3.1. Octavia OVN ドライバー

Octavia は Octavia API を使用して複数のプロバイダードライバーをサポートします。

利用可能なすべての Octavia プロバイダードライバーをコマンドラインで表示するには、以下を入力します。

```
$ openstack loadbalancer provider list
```

出力例

```
+---------+--------------------------------------------+
| name    | description                                |
+---------+--------------------------------------------+
| amphora | The Octavia Amphora driver.                |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn     | Octavia OVN driver.                        |
+---------+--------------------------------------------+
```

RHOSP バージョン 16 以降、Octavia OVN プロバイダードライバー (**ovn**) は RHOSP デプロイメントの OpenShift Container Platform でサポートされます。

ovn は、Octavia および OVN が提供する負荷分散用の統合ドライバーです。これは基本的な負荷分散機能をサポートし、OpenFlow ルールに基づいています。このドライバーは、OVN Neutron ML2 を使用するデプロイメント上の director により Octavia で自動的に有効にされます。

Amphora プロバイダードライバーがデフォルトのドライバーです。ただし、**ovn** が有効にされる場合には、Kuryr がこれを使用します。

Kuryr が Amphora の代わりに **ovn** を使用する場合は、以下の利点があります。

- リソース要件が減少します。Kuryr は、各サービスにロードバランサーの仮想マシンを必要としません。
- ネットワークレイテンシーが短縮されます。
- サービスごとに仮想マシンを使用する代わりに、OpenFlow ルールを使用することで、サービスの作成速度が上がります。
- Amphora 仮想マシンで集中管理されるのではなく、すべてのノードに分散負荷分散アクションが分散されます。

RHOSP クラウドがバージョン 13 から 16 にアップグレードした後に、クラスターを Octavia OVN ドライバーを使用するように設定 できます。

12.3.3.4. Kuryr を使用したインストールについての既知の制限

OpenShift Container Platform を Kuryr SDN で使用する場合、いくつかの既知の制限があります。

RHOSP の一般的な制限
OpenShift Container Platform を Kuryr SDN と共に使用する場合は、すべてのバージョンおよび環境に適用されるいくつかの制限があります。

- **NodePort** タイプの **Service** オブジェクトはサポートされません。
- OVN Octavia プロバイダーを使用するクラスターは、**Service** オブジェクトをサポートします。このオブジェクトについて、**.spec.selector** プロパティーは、**Endpoints** オブジェクトの **.subsets.addresses** プロパティーにノードまたは Pod のサブネットが含まれる場合は指定されません。
- マシンが作成されるサブネットがルーターに接続されていない場合や、サブネットが接続されていても、ルーターに外部ゲートウェイが設定されていない場合、Kuryr はタイプが **LoadBalancer** の **Service** オブジェクトの Floating IP を作成できません。
- **Service** オブジェクトで **sessionAffinity=ClientIP** プロパティーを設定しても効果はありません。Kuryr はこの設定をサポートしていません。

RHOSP バージョンの制限
OpenShift Container Platform を Kuryr SDN で使用する場合は、RHOSP バージョンに依存するいくつかの制限があります。

- RHOSP の 16 よりも前のバージョンでは、デフォルトの Octavia ロードバランサードライバー (Amphora) を使用します。このドライバーでは、OpenShift Container Platform サービスごとに 1 つの Amphora ロードバランサー仮想マシンをデプロイする必要があります。サービス数が多すぎると、リソースが不足する可能性があります。
  OVN Octavia ドライバーが無効にされている以降のバージョンの RHOSP のデプロイメントでも Amphora ドライバーを使用します。この場合も、RHOSP の以前のバージョンと同じリソースに関する懸念事項を考慮する必要があります。

- バージョン 13.0.13 よりも前の Octavia RHOSP バージョンは UDP リスナーをサポートしません。そのため、OpenShift Container Platform UDP サービスはサポートされません。

- 13.0.13 よりも前の Octavia RHOSP バージョンは、同じポートで複数のプロトコルをリッスンできません。TCP や UDP など、同じポートを異なるプロトコルに公開するサービスはサポートされません。

- Kuryr SDN は、サービスによる自動解凍をサポートしていません。

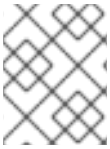
RHOSP 環境の制限
Kuryr SDN を使用する場合に、デプロイメント環境に依存する制限事項があります。

Octavia には UDP プロトコルおよび複数のリスナーのサポートがないため、RHOCP バージョンが 13.0.13 よりも前のバージョンの場合、Kuryr は Pod が DNS 解決に TCP を使用するように強制します。

Go バージョン 1.12 以前では、CGO サポートが無効にされた状態でコンパイルされたアプリケーションは UDP のみを使用します。この場合、ネイティブの Go リゾルバーは、TCP が DNS 解決に強制的に実行されるかどうかを制御する、**resolv.conf** の **use-vc** オプションを認識しません。その結果、UDP は引き続き DNS 解決に使用されますが、これは失敗します。

TCP の強制を許可するには、環境変数 **CGO_ENABLED** を **1** に設定 (例: **CGO_ENABLED=1**) されている状態でアプリケーションをコンパイルするか、または変数がないことを確認します。

Go バージョン 1.13 以降では、UDP を使用した DNS 解決が失敗する場合に TCP が自動的に使用されます。



注記

Alpine ベースのコンテナーを含む musl ベースのコンテナーは **use-vc** オプションをサポートしません。

RHOSP のアップグレードの制限
RHOSP のアップグレードプロセスにより、Octavia API が変更され、ロードバランサーに使用される Amphora イメージへのアップグレードが必要になる可能性があります。

API の変更に個別に対応できます。

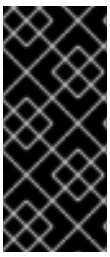
Amphora イメージがアップグレードされると、RHOSP Operator は既存のロードバランサー仮想マシンを 2 つの方法で処理できます。

- ロードバランサーのフェイルオーバー をトリガーしてそれぞれの仮想マシンをアップグレードします。

- ユーザーが仮想マシンのアップグレードを行う必要があります。

Operator が最初のオプションを選択する場合、フェイルオーバー時に短い時間のダウンタイムが生じる可能性があります。

Operator が 2 つ目のオプションを選択する場合、既存のロードバランサーは UDP リスナーなどのアップグレードされた Octavia API 機能をサポートしません。この場合、ユーザーはこれらの機能を使用するためにサービスを再作成する必要があります。



重要

OpenShift Container Platform が UDP の負荷分散をサポートする新規の Octavia バージョンを検出する場合、これは DNS サービスを自動的に再作成します。サービスの再作成により、サービスのデフォルトが UDP の負荷分散をサポートするようになります。

再作成により、DNS サービスに約 1 分間のダウンタイムが発生します。

12.3.3.5. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.3.3.6. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

12.3.3.7. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

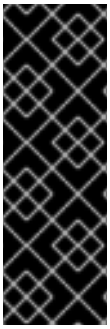
- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.3.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.3.5. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。

重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。

- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

12.3.6. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

重要

外部ネットワークの CIDR 範囲がデフォルトのネットワーク範囲のいずれかと重複している場合、インストールプロセスを開始する前に、**install-config.yaml** ファイルで一致するネットワーク範囲を変更する必要があります。

デフォルトのネットワーク範囲は以下のとおりです。

ネットワーク	範囲
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

警告

インストールプログラムにより同じ名前を持つ複数のネットワークが見つかる場合、それらのネットワークのいずれかがランダムに設定されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

12.3.7. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.3.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

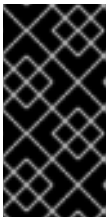
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.3.9. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

      iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

      iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

      v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

      vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

      vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

      viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

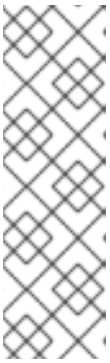
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

12.3.9.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を

4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

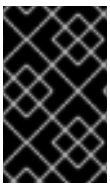
cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

12.3.10. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.3.10.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.8 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値
pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

12.3.10.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.9 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 `networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23`

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
	
	複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

12.3.10.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.10 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。
	
	注記
	
	インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。
		1 つ以上のキー。以下に例を示します。
		
		`sshKey:`
		`  <key1>`
		`  <key2>`
		`  <key3>`

12.3.10.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表12.11 追加の RHOSP パラメーター

パラメーター	説明	値

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

12.3.10.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表12.12 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
compute.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
compute.platform.openstack.rootVolume.zones	コンピュートマシンの root ボリュームをインストールするアベイラビリティーゾーン。このパラメーターに値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

パラメーター	説明	値
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.rootVolume.zones	コントロールプレーンマシンの root ボリュームをインストールするアベイラビリティーゾーン。この値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform.openstack.clusterOSImageProperties	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアの一覧。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.ingressFloatingIP	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform.openstack.apiFloatingIP	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

12.3.10.6. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。

- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。

注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

12.3.10.7. Kuryr を使用した OpenStack のカスタマイズされた install-config.yaml ファイルのサンプル

デフォルトの OpenShift SDN ではなく Kuryr SDN を使用してデプロイするには、**install-config.yaml** ファイルを変更して **Kuryr** を必要な **networking.networkType** として追加してから、デフォルトの OpenShift Container Platform SDN インストール手順に進む必要があります。このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16 1
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
    trunkSupport: true 2
    octaviaSupport: true 3
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

1 Amphora Octavia ドライバーは、ロードバランサーごとに 2 つのポートを作成します。そのため、インストーラーが作成するサービスサブネットは、**serviceNetwork** プロパティーの値として指定される CIDR のサイズは 2 倍になります。IP アドレスの競合を防ぐには、範囲をより広くする必要があります。

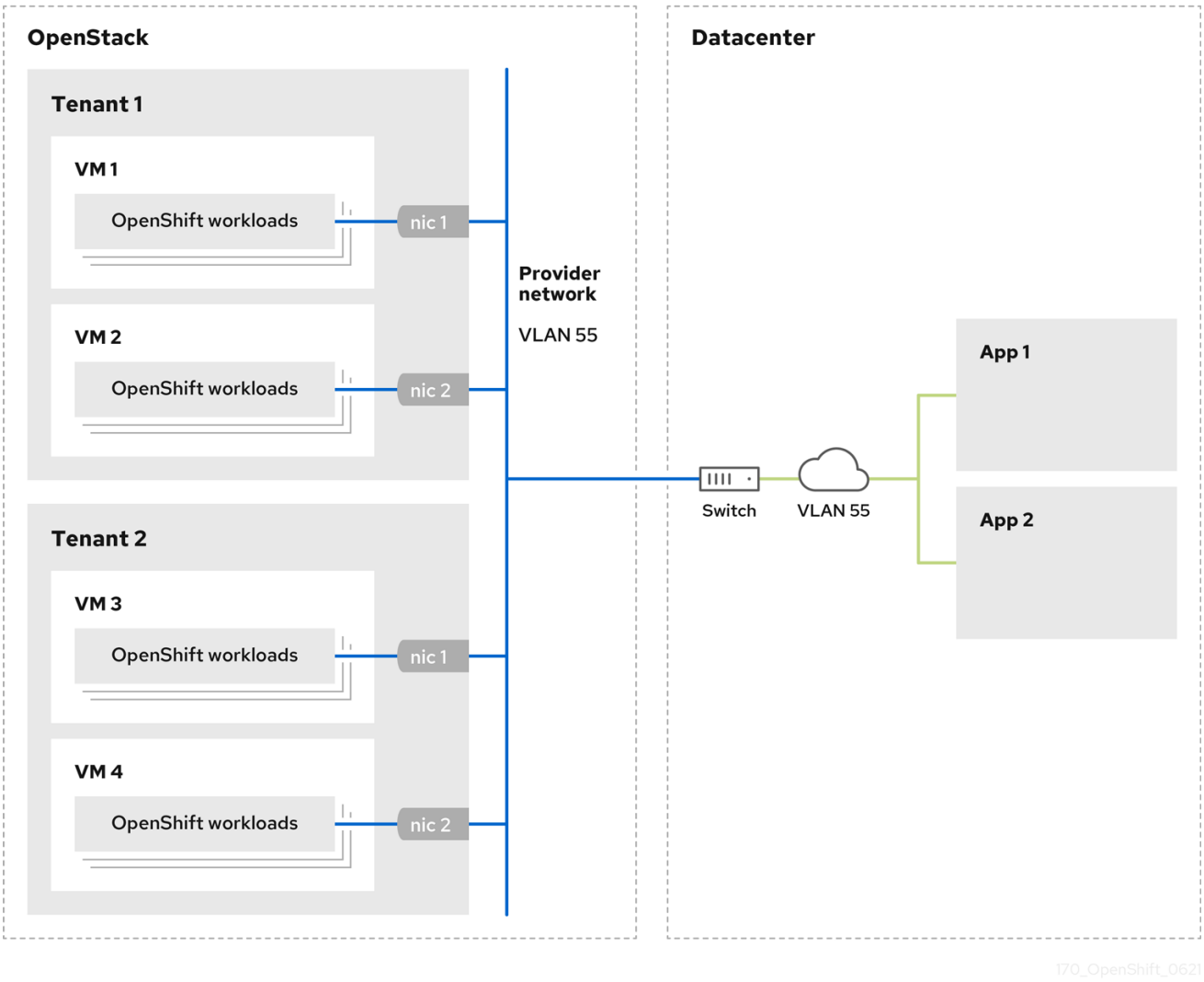
2 3 **trunkSupport** と **octaviaSupport** の両方はインストーラーによって自動的に検出されるため、それらを設定する必要はありません。ただし、ご使用の環境がこれらの両方の要件を満たさないと、Kuryr SDN は適切に機能しません。トランクは Pod を RHOSP ネットワークに接続するために必要であり、Octavia は OpenShift Container Platform サービスを作成するために必要です。

12.3.10.8. RHOSP プロバイダーネットワーク上のクラスターデプロイメント

プロバイダーネットワーク上のプライマリーネットワークインターフェイスを使用して、OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイできます。プロバイダーネットワークは一般的に、インターネットへの到達に使用可能なパブリックネットワークに、プロジェクトが直接アクセスできるように使用します。ネットワーク作成プロセスの一環として、プロバイダーネットワークをプロジェクト間で共有することもできます。

RHOSP プロバイダーネットワークは、データセンター内の既存の物理ネットワークに直接マップします。RHOSP 管理者はこれらを作成する必要があります。

以下の例では、OpenShift Container Platform ワークロードはプロバイダーネットワークを使用してデータセンターに接続されます。



プロバイダーネットワークにインストールされている OpenShift Container Platform クラスターは、テナントネットワークまたは Floating IP アドレスを必要としません。インストーラーは、インストール中にこれらのリソースを作成しません。

プロバイダーネットワークタイプの例には、フラット (タグなし) および VLAN (802.1Q タグ付き) が含まれます。

注記

クラスターは、ネットワークタイプが許可する限り多くのプロバイダーネットワーク接続をサポートできます。たとえば、VLAN ネットワークは、通常最大 4096 の接続をサポートします。

プロバイダーネットワークおよびテナントネットワークの詳細は、RHOSP のドキュメント を参照してください。

12.3.10.8.1. クラスターのインストールにおける RHOSP プロバイダーネットワーク要件

OpenShift Container Platform クラスターをインストールする前に、Red Hat OpenStack Platform (RHOSP) のデプロイメントおよびプロバイダーネットワークは、さまざまな条件を満たす必要があります。

- RHOSP ネットワークサービス (Neutron) が有効化され、RHOSP ネットワーク API 経由でアクセス可能であること。
- RHOSP ネットワークサービスでは、ポートセキュリティーと許可するアドレスペアの機能拡張が有効化 されていること。
- プロバイダーネットワークは他のテナントと共有できます。

  ヒント

  --share フラグを指定して **openstack network create** コマンドを使用して、共有できるネットワークを作成します。

- クラスターのインストールに使用する RHOSP プロジェクトは、プロバイダーネットワークと適切なサブネットを所有する必要があります。

  ヒント

  openshift という名前のプロジェクトのネットワークを作成するには、以下のコマンドを入力します。

  ```
  $ openstack network create --project openshift
  ```

  openshift という名前のプロジェクトのサブネットを作成するには、以下のコマンドを入力します。

  ```
  $ openstack subnet create --project openshift
  ```

  RHOSP でのネットワークの作成に関する詳細は、プロバイダーネットワークに関するドキュメント を参照してください。

  クラスターが **admin** ユーザーによって所有されている場合、そのユーザーとしてインストーラーを実行してネットワーク上でポートを作成する必要があります。

  重要

  プロバイダーネットワークは、クラスターの作成に使用される RHOSP プロジェクトによって所有されている必要があります。所有されていない場合は、RHOSP Compute サービス (Nova) はそのネットワークからポートを要求できません。

- プロバイダーネットワークが、デフォルトで **169.254.169.254** である RHOSP メタデータサービスの IP アドレスに到達できることを確認します。
  RHOSP SDN とネットワークサービス設定によっては、サブネットを作成する際に、ルートを提供しなければならない場合があります。以下に例を示します。

  ```
  $ openstack subnet create --dhcp --host-route
  destination=169.254.169.254/32,gateway=192.0.2.2 ...
  ```

- オプション: ネットワークのセキュリティーを保護するには、単一のプロジェクトへのネットワークアクセスを制限する ロールベースのアクセス制御 (RBAC) ルールを作成します。

12.3.10.8.2. プロバイダーネットワークにプライマリーインターフェイスを持つクラスターのデプロイ

Red Hat OpenStack Platform (RHOSP) プロバイダーネットワーク上にプライマリーネットワークインターフェイスを持つ OpenShift Container Platform クラスターをデプロイすることができます。

- クラスターのインストールにおける RHOSP プロバイダーネットワーク要件に記載されているとおりに、お使いの Red Hat OpenStack Platform (RHOSP) のデプロイメントが設定されています。

手順

1. テキストエディターで **install-config.yaml** ファイルを開きます。
2. **platform.openstack.apiVIP** プロパティーの値を API VIP の IP アドレスに設定します。
3. **platform.openstack.ingressVIP** プロパティーの値を Ingress VIP の IP アドレスに設定します。
4. **platform.openstack.machinesSubnet** プロパティーの値をプロバイダーネットワークサブネットの UUID に設定します。
5. **networking.machineNetwork.cidr** プロパティーの値をプロバイダーネットワークサブネットの CIDR ブロックに設定します。

重要

platform.openstack.apiVIP プロパティーおよび **platform.openstack.ingressVIP** プロパティーはいずれも、**networking.machineNetwork.cidr** ブロックから割り当てられていない IP アドレスである必要があります。

RHOSP プロバイダーネットワークに依存するクラスターのインストール設定ファイルのセクション

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

警告

プライマリーネットワークインターフェイスにプロバイダーネットワークを使用している間は、**platform.openstack.externalNetwork** パラメーターまたは **platform.openstack.externalDNS** パラメーターを設定することはできません。

クラスターをデプロイする際に、インストーラーは **install-config.yaml** ファイルを使用してプロバイダーネットワークにクラスターをデプロイします。

ヒント

プロバイダーネットワークを含むネットワークを **platform.openstack.additionalNetworkIDs** 一覧に追加できます。

クラスターのデプロイ後に、Pod を追加のネットワークに接続することができます。詳細は、複数ネットワークについて を参照してください。

12.3.10.9. Kuryr ポートプール

Kuryr ポートプールでは、Pod 作成のスタンバイ状態の多数のポートを維持します。

ポートをスタンバイ状態に維持すると、Pod の作成時間が必要最小限に抑えることができます。ポートプールを使用しない場合には、Kuryr は Pod が作成または削除されるたびにポートの作成または削除を明示的に要求する必要があります。

Kuryr が使用する Neutron ポートは、namespace に関連付けられるサブネットに作成されます。これらの Pod ポートは、OpenShift Container Platform クラスターノードのプライマリーポートにサブポートとして追加されます。

Kuryr は namespace をそれぞれ、別のサブネットに保存するため、namespace-worker ペアごとに別個のポートプールが維持されます。

クラスターをインストールする前に、**cluster-network-03-config.yml** マニフェストファイルに以下のパラメーターを設定して、ポートプールの動作を設定できます。

- **enablePortPoolsPrepopulation** パラメーターで、プールの事前生成が制御されるので、Kuryr は新規ホストが追加される時や新規 namespace の作成時などの作成時に、Kuryr によりプールにポートが強制的に追加されるようにします。デフォルト値は **false** です。
- **poolMinPorts** パラメーターは、プールに保持する空きポートの最小数です。デフォルト値は **1** です。
- **poolMaxPorts** パラメーターは、プールに保持する空きポートの最大数です。値が **0** の場合は、上限が無効になります。これはデフォルト設定です。
  OpenStack ポートのクォータが低い場合や、Pod ネットワークで IP アドレスの数が限定されている場合には、このオプションを設定して、不要なポートが削除されるようにします。
- **poolBatchPorts** パラメーターは、一度に作成可能な Neutron ポートの最大数を定義します。デフォルト値は **3** です。

12.3.10.10. インストール時の Kuryr ポートプールの調整

インストール時に、Pod 作成の速度や効率性を制御するために Kuryr で Red Hat OpenStack Platform (RHOSP) Neutron ポートを管理する方法を設定できます。

前提条件

- **install-config.yaml** ファイルを作成して変更しておく。

手順

1. コマンドラインからマニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Cluster Network Operator 設定を記述するカスタムリソース (CR) を入力します。

```
$ oc edit networks.operator.openshift.io cluster
```

4. 要件に合わせて設定を編集します。以下のファイルをサンプルとして紹介しています。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: Kuryr
    kuryrConfig:
      enablePortPoolsPrepopulation: false 1
      poolMinPorts: 1 2
      poolBatchPorts: 3 3
      poolMaxPorts: 5 4
      openstackServiceNetwork: 172.30.0.0/15 5
```

1. **enablePortPoolsPrepopulation** の値を **true** に設定し、namespace の作成時、または新規ノードがクラスターに追加された後に Kuryr が新規 Neutron ポートを作成するようにします。この設定により、Neutron ポートのクォータが引き上げられますが、Pod の起動に必要となる時間を短縮できます。デフォルト値は **false** です。

2. Kuryr は、対象のプール内にある空きポートの数が **poolMinPorts** の値よりも少ない場合には、プールに新規ポートを作成します。デフォルト値は **1** です。

3. **poolBatchPorts** は、空きポートの数が **poolMinPorts** の値よりも少ない場合に作成される新規ポートの数を制御します。デフォルト値は **3** です。

4. プール内の空きポートの数が **poolMaxPorts** の値よりも多い場合に、Kuryr はその値と同じ数になるまでポートを削除します。この値を **0** に設定すると、この上限は無効になり、プールが縮小できないようにします。デフォルト値は **0** です。

5. **openStackServiceNetwork** パラメーターは、RHOSP Octavia の LoadBalancer に割り当てられるネットワークの CIDR 範囲を定義します。

   このパラメーターを Amphora ドライバーと併用する場合には、Octavia は、ロードバランサーごとに、このネットワークから IP アドレスを 2 つ (OpenShift 用に 1 つ、VRRP 接続用に 1 つ) 取得します。これらの IP アドレスは OpenShift Container Platform と Neutron でそれぞれ管理されるため、異なるプールから取得する必要があります。したがって、**openStackServiceNetwork serviceNetwork** の値の 2 倍になる必要があり、**serviceNetwork** の値は、**openStackServiceNetwork** で定義された範囲と完全に重複する必要があります。

   CNO は、このパラメーターの定義範囲から取得した VRRP IP アドレスが **serviceNetwork** パラメーターの定義範囲と重複しないことを検証します。

   このパラメーターが設定されていない場合には、CNO は **serviceNetwork** の拡張値を使用します。この値は、プリフィックスのサイズを 1 つずつ減らして決定します。

5. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

6. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

12.3.11. コンピュートマシンのアフィニティーの設定

オプションで、インストール時にコンピュートマシンのアフィニティーポリシーを設定できます。インストーラーは、デフォルトでコンピュートマシンのアフィニティーポリシーを選択しません。

インストール後に特定の RHOSP サーバーグループを使用するマシンセットを作成することもできます。

注記

コントロールプレーンマシンは、**soft-anti-affinity** ポリシーで作成されます。

ヒント

RHOSP インスタンスのスケジューリングおよび配置 の詳細は、RHOSP のドキュメントを参照してください。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. RHOSP コマンドラインインターフェイスを使用して、コンピュートマシンのサーバーグループを作成します。以下に例を示します。

```
$ openstack \
    --os-compute-api-version=2.15 \
    server group create \
    --policy anti-affinity \
    my-openshift-worker-group
```

詳細は、 **server group create** コマンドのドキュメント を参照してください。

2. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

ここでは、以下のようになります。

installation_directory

クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

3. **MachineSet** 定義ファイルの **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** を作成します。

4. **spec.template.spec.providerSpec.value** プロパティーの下にある定義に、プロパティー **serverGroupID** を追加します。以下に例を示します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_ID>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
    spec:
      providerSpec:
```

```
      value:
        apiVersion: openstackproviderconfig.openshift.io/v1alpha1
        cloudName: openstack
        cloudsSecret:
          name: openstack-cloud-credentials
          namespace: openshift-machine-api
        flavor: <nova_flavor>
        image: <glance_image_name_or_location>
        serverGroupID: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (1)
        kind: OpenstackProviderSpec
        networks:
        - filter: {}
          subnets:
          - filter:
              name: <subnet_name>
              tags: openshiftClusterID=<infrastructure_ID>
        securityGroups:
        - filter: {}
          name: <infrastructure_ID>-<node_role>
        serverMetadata:
          Name: <infrastructure_ID>-<node_role>
          openshiftClusterID: <infrastructure_ID>
        tags:
        - openshiftClusterID=<infrastructure_ID>
        trunk: true
        userDataSecret:
          name: <node_role>-user-data
        availabilityZone: <optional_openstack_availability_zone>
```

(1) サーバーグループの UUID をここに追加します。

5. オプション: **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

クラスターのインストール時に、インストーラーは変更した **MachineSet** 定義を使用して RHOSP サーバーグループ内にコンピュートマシンを作成します。

12.3.12. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.3.13. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.3.13.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API およびクラスターアプリケーションへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

4. FIP を、以下のパラメーターの値として **install-config.yaml** ファイルに追加します。
   - **platform.openstack.ingressFloatingIP**
   - **platform.openstack.apiFloatingIP**

これらの値を使用する場合には、**install-config.yaml** ファイルの **platform.openstack.externalNetwork** パラメーターの値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.3.13.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

install-config.yaml ファイルで以下のパラメーターを定義しないでください。

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

外部ネットワークを提供できない場合は、**platform.openstack.externalNetwork** を空白のままにすることもできます。**platform.openstack.externalNetwork** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。外部接続を独自に設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムからインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。

注記

API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.3.14. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

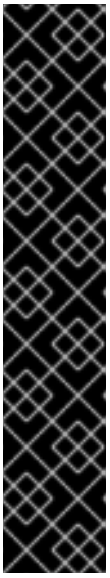
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

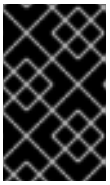
注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

12.3.15. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

12.3.16. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

12.3.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.3.18. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.4. SR-IOV で接続されたコンピュートマシンをサポートする OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、Single Root I/O Virtualization (SR-IOV) テクノロジーと共にコンピュートマシンを使用できる Red Hat OpenStack Platform (RHOSP) にクラスターをインストールできます。

12.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。

    - OpenShift クラスターでサポートされるプラットフォームのセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認します。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。

- ネットワーク設定がプロバイダーのネットワークに依存しないことを確認します。プロバイダーネットワークはサポートされません。

- ブロックストレージ (Cinder) またはオブジェクトストレージ (Swift) などのストレージサービスが RHOSP にインストールされている必要があります。オブジェクトストレージは、OpenShift Container Platform レジストリークラスターデプロイメントに推奨されるストレージ技術です。詳細は、ストレージの最適化 を参照してください。

- RHOSP でメタデータサービスが有効化されています。

12.4.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表12.13 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

12.4.2.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.4.2.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

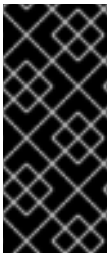
それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

また、Single-root Input/Output Virtualization (SR-IOV) を使用するクラスターの場合、RHOSP コンピュートノードには Huge Page をサポートするフレーバーが必要です。



重要

SR-IOV デプロイメントでは、多くの場合、専用の CPU や分離された CPU などのパフォーマンスの最適化が駆使されます。パフォーマンスを最大化するには、基礎となる RHOSP デプロイメントをこれらの最適化機能を使用するように設定してから、OpenShift Container Platform コンピュートマシンを最適化されたインフラストラクチャーで実行するように設定します。

関連情報

- パフォーマンスの良い RHOSP コンピュートノードの設定についての詳細は、パフォーマンスを向上させるためのコンピュートノードの設定 について参照してください。

12.4.2.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.4.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

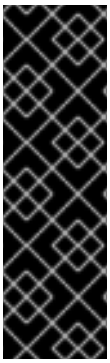
- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.4.4. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。



重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

12.4.5. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
```

```
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

注記

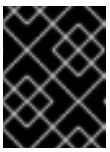
Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

12.4.6. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: 'devuser'
           password: XXX
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

   ```
   clouds:
     shiftstack:
       ...
       cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
   ```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

   ```
   $ oc edit configmap -n openshift-config cloud-provider-config
   ```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.4.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

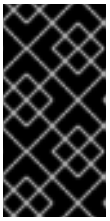
1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。

2. インフラストラクチャープロバイダーを選択します。

3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.4.8. インストール設定ファイルの作成

インストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

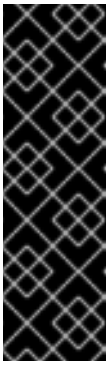
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

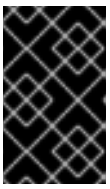
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. クラスターの記述名を入力します。

iii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

12.4.8.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

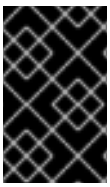
cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

12.4.9. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.4.9.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.14 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.4.9.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.15 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

12.4.9.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.16 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

12.4.9.4. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。
- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。

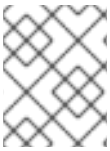
注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

12.4.9.5. ベアメタルマシンを使用したクラスターのデプロイ

クラスターがベアメタルマシンを使用する必要がある場合は、**inventory.yaml** ファイルを変更します。クラスターには、ベアメタル上でコントロールプレーンとコンピュートマシンの両方を実行させることも、コンピュートマシンのみを実行させることもできます。

ベアメタルコンピュートマシンは、Kuryr を使用するクラスターではサポートされません。



注記

install-config.yaml ファイルで、ベアメタルワーカーに使用する RHOSP ネットワークが Floating IP アドレスをサポートするかどうかが反映されていることを確認します。

前提条件

- RHOSP の ベアメタルサービス (Ironic) は有効にされており、RHOSP Compute API でアクセスできる。
- ベアメタルは RHOSP フレーバー として利用可能である。
- RHOSP ネットワークは、仮想マシンとベアメタルサーバー接続の両方をサポートする。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。

- マシンを既存のネットワークにデプロイする必要がある場合、RHOSP サブネットがプロビジョニングされる。
- マシンをインストーラーでプロビジョニングされるネットワークに場合、RHOSP ベアメタルサービス (Ironic) はテナントネットワークで実行される Preboot eXecution Environment (PXE) ブートマシンをリッスンし、これと対話できます。
- **inventory.yaml** ファイルを OpenShift Container Platform インストールプロセスの一部として作成している。

手順

1. **inventory.yaml** ファイルで、マシンのフレーバーを編集します。
   a. ベアメタルコントロールプレーンマシンを使用する場合は、**os_flavor_master** の値をベアメタルフレーバーに変更します。
   b. **os_flavor_worker** の値をベアメタルフレーバーに変更します。

   ベアメタルの inventory.yaml のサンプルファイル

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
...
```

1 ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、この値をベアメタルのフレーバーに変更します。

2 この値を、コンピュートマシンに使用するベアメタルのフレーバーに変更します。

更新された **inventory.yaml** ファイルを使用してインストールプロセスを完了します。デプロイメント時に作成されるマシンは、ファイルに追加したフレーバーを使用します。

注記

インストーラーは、ベアメタルマシンの起動中にタイムアウトする可能性があります。

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
./openshift-install wait-for install-complete --log-level debug
```

12.4.9.6. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

12.4.10. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.4.11. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.4.11.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API およびクラスターアプリケーションへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

4. FIP を、以下のパラメーターの値として **install-config.yaml** ファイルに追加します。
   - **platform.openstack.ingressFloatingIP**
   - **platform.openstack.apiFloatingIP**

これらの値を使用する場合には、**install-config.yaml** ファイルの **platform.openstack.externalNetwork** パラメーターの値として外部ネットワークを入力する必要もあります。

ヒント

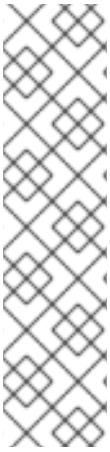
Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.4.11.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

ファイルで以下を定義しないでください。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムからインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。



注記

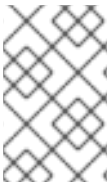
API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.4.12. コンピュートマシン用の SR-IOV ネットワークの作成

Red Hat OpenStack Platform (RHOSP) デプロイメントで Single Root I/O Virtualization (SR-IOV) をサポートする場合、コンピュートマシンを実行する SR-IOV ネットワークをプロビジョニングすることができます。



注記

以下の手順では、コンピュートマシンへの接続が可能な外部のフラットネットワークおよび外部の VLAN ベースのネットワークを作成します。RHOSP のデプロイメントによっては、ネットワークの他のタイプが必要になる場合があります。

前提条件

- クラスターは SR-IOV をサポートしている。

  注記

  クラスターがサポートするかどうかが不明な場合は、OpenShift Container Platform SR-IOV ハードウェアネットワークについてのドキュメントを参照してください。

- RHOSP デプロイメントの一部として、無線とアップリンクのプロバイダーネットワークを作成している。これらのネットワークを表すために **radio** および **uplink** の名前がすべてのコマンド例で使用されています。

手順

1. コマンドラインで、無線の RHOSP ネットワークを作成します。

   ```
   $ openstack network create radio --provider-physical-network radio --provider-network-type
   flat --external
   ```

2. アップリンクの RHOSP ネットワークを作成します。

```
$ openstack network create uplink --provider-physical-network uplink --provider-network-type
vlan --external
```

3. 無線ネットワーク用のサブネットを作成します。

```
$ openstack subnet create --network radio --subnet-range <radio_network_subnet_range>
radio
```

4. アップリンクネットワーク用のサブネットを作成します。

```
$ openstack subnet create --network uplink --subnet-range <uplink_network_subnet_range>
uplink
```

12.4.13. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

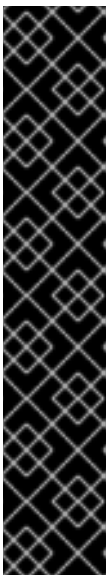
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

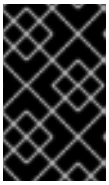
注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

12.4.14. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

12.4.15. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

クラスターが機能しています。ただし、SR-IOV コンピュートマシンを追加する前に、追加のタスクを実行する必要があります。

12.4.16. SR-IOV 向けに RHOSP で実行されるクラスターの準備

Red Hat OpenStack Platform (RHOSP) で実行されるクラスターで single root I/O virtualization (SR-IOV) を使用する前に、RHOSP メタデータをドライブとしてマウント可能にし、仮想機能 I/O (VFIO) ドライバーの非 IOMMU Operator を有効にします。

12.4.16.1. RHOSP メタデータサービスをマウント可能なドライブとして有効化

マシン設定をマシンプールに適用することで、Red Hat OpenStack Platform (RHOSP) メタデータサービスをマウント可能なドライブとして利用可能にすることができます。

以下のマシン設定により、SR-IOV ネットワーク Operator 内から RHOSP ネットワーク UUID を表示できます。この設定により、SR-IOV リソースのクラスター SR-IOV リソースへの関連付けが単純化されます。

手順

1. 以下のテンプレートからマシン設定ファイルを作成します。

マウント可能なメタデータサービスマシン設定ファイル

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 20-mount-config (1)
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: create-mountpoint-var-config.service
          enabled: true
          contents: |
            [Unit]
            Description=Create mountpoint /var/config
            Before=kubelet.service

            [Service]
            ExecStart=/bin/mkdir -p /var/config

            [Install]
            WantedBy=var-config.mount

        - name: var-config.mount
          enabled: true
          contents: |
            [Unit]
            Before=local-fs.target
            [Mount]
```

```
        Where=/var/config
        What=/dev/disk/by-label/config-2
        [Install]
        WantedBy=local-fs.target
```

1 選択する名前に置き換えることができます。

2. コマンドラインからマシン設定を適用します。

```
$ oc apply -f <machine_config_file_name>.yaml
```

12.4.16.2. RHOSP VFIO ドライバーの非 IOMMU 機能の有効化

マシン設定をマシンプールに適用することで、Red Hat OpenStack Platform (RHOSP) 仮想機能 I/O (VFIO) ドライバーの非 IOMMU 機能を有効にすることができます。RHOSP vfio-pci ドライバーではこの機能が必要です。

手順

1. 以下のテンプレートからマシン設定ファイルを作成します。

非 IOMMU VFIO マシン設定ファイル

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 99-vfio-noiommu 1
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - path: /etc/modprobe.d/vfio-noiommu.conf
        mode: 0644
        contents:
          source:
data:;base64,b3B0aW9ucyB2ZmlvIGVuYWJsZV91bnNhZmVfbm9pb21tdV9tb2RlPTEK
```

1 選択する名前に置き換えることができます。

2. コマンドラインからマシン設定を適用します。

```
$ oc apply -f <machine_config_file_name>.yaml
```

クラスターがインストールされ、SR-IOV 設定用に準備されます。次のステップのセクションに記載されているインストール後の SR-IOV タスクを完了します。

12.4.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.4.18. 次のステップ

- クラスターの SR-IOV 設定を完了するには、以下を実行します。
  - Performance Addon Operator をインストール します。
  - Huge Page のサポートのある Performance Addon Operator を設定 します。
  - SR-IOV Operator をインストール します。
  - SR-IOV ネットワークデバイスを設定 します。
  - SR-IOV コンピュートマシンセットを追加 します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.5. 独自のインフラストラクチャーを使用した OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、ユーザーによってプロビジョニングされたインフラストラクチャーを実行するクラスターを Red Hat OpenStack Platform (RHOSP) にインストールできます。

独自のインフラストラクチャーを使用することで、クラスターを既存のインフラストラクチャーおよび変更と統合できます。このプロセスでは、インストーラーでプロビジョニングされるインストールの場合よりも多くの手作業が必要になります。Nova サーバー、Neutron ポート、セキュリティーグループなどのすべての RHOSP リソースを作成する必要があるためです。ただし、Red Hat では、デプロイメントプロセスを支援する Ansible Playbook を提供しています。

12.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

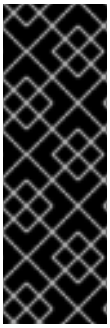
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- OpenShift クラスターでサポートされるプラットフォーム のセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認している。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- OpenShift Container Platform のインストール先に RHOSP アカウントがある。
- インストールプログラムを実行するマシンには、以下が含まれる。
  - インストールプロセス時に作成したファイルを保持できる単一ディレクトリー
  - Python 3

12.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.5.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表12.17 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

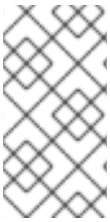
リソース	値
Floating IP アドレス	3

リソース	値
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

12.5.3.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス

- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.5.3.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

12.5.3.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.5.4. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでのインストールプロセスを単純化する Ansible Playbook には、複数の Python モジュールが必要です。インストーラーを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされている。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

12.5.5. インストール Playbook のダウンロード

OpenShift Container Platform を独自の Red Hat OpenStack Platform (RHOSP) インフラストラクチャーにインストールするために使用できる Ansible Playbook をダウンロードします。

前提条件

- curl コマンドラインツールがマシンで利用できる。

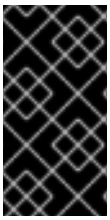
手順

- Playbook を作業ディレクトリーにダウンロードするには、コマンドラインから以下のスクリプトを実行します。

```
$ xargs -n 1 curl -O <<< '
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/bootstrap.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/common.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/compute-nodes.yaml
      https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/control-
plane.yaml
```

```
    https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/security-
groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
load-balancers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-
containers.yaml'
```

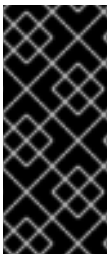
Playbook はマシンにダウンロードされます。



重要

インストールプロセス時に、Playbook を変更してデプロイメントを設定できます。

クラスターの有効期間中に、すべての Playbook を保持します。OpenShift Container Platform クラスターを RHOSP から削除するには Playbook が必要です。



重要

bootstrap.yaml、**compute-nodes.yaml**、**control-plane.yaml**、**network.yaml**、および **security-groups.yaml** ファイルに加えた編集内容は、**down-** の接頭辞が付けられた対応する Playbook に一致している必要があります。たとえば、**bootstrap.yaml** ファイルへの編集は、**down-bootstrap.yaml** ファイルにも反映される必要があります。両方のファイルを編集しない場合、サポートされるクラスターの削除プロセスは失敗します。

12.5.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

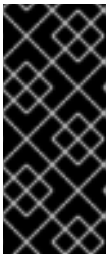
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。

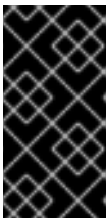
2. インフラストラクチャープロバイダーを選択します。

3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.5.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定しま

す。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.5.8. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成

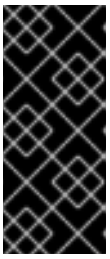
OpenShift Container Platform インストールプログラムでは、Red Hat Enterprise Linux CoreOS (RHCOS) イメージが Red Hat OpenStack Platform (RHOSP) クラスターに存在する必要があります。最新の RHCOS イメージを取得した後、RHOSP CLI を使用してこれをアップロードします。

前提条件

- RHOSP CLI がインストールされています。

手順

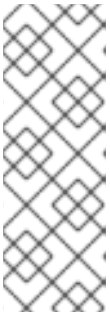
1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、Red Hat Enterprise Linux (RHEL) 8 用の OpenShift Container Platform 4.8 の最新リリースを選択します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**をダウンロードします。
4. イメージを展開します。



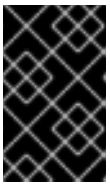
注記

クラスターが使用する前に RHOSP イメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. ダウンロードしたイメージから、RHOSP CLI を使用して **rhcos** という名前のイメージをクラスターに作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```



重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

RHOSP にイメージをアップロードした後は、インストールプログラムでイメージを利用できます。

12.5.9. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

12.5.10. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.5.10.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API、クラスターアプリケーション、およびブートストラッププロセスへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. Red Hat OpenStack Platform (RHOSP) CLI を使用して、ブートストラップ FIP を作成します。

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

5. FIP を以下の変数の値として **inventory.yaml** ファイルに追加します。
   - **os_api_fip**
   - **os_bootstrap_fip**
   - **os_ingress_fip**

これらの値を使用する場合には、**inventory.yaml** ファイルの **os_external_network** 変数の値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.5.10.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

inventory.yaml ファイルで、以下の変数を定義しないでください。

- **os_api_fip**
- **os_bootstrap_fip**

- **os_ingress_fip**

外部ネットワークを提供できない場合は、**os_external_network** を空白のままにすることもできます。**os_external_network** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。インストールプロセスで、ネットワークリソースを作成する際に、独自の外部接続を設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムから **wait-for** コマンドでインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。

注記

API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

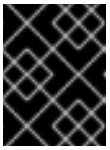
DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.5.11. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
     ```

```
    user_domain_name: Default
    project_domain_name: Default
  dev-env:
   region_name: RegionOne
   auth:
    username: 'devuser'
    password: XXX
    project_name: 'devonly'
    auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.5.12. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

      iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

      iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

      v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

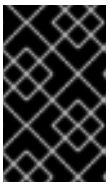
      vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

      vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

      viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



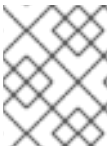
重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

これで、指定したディレクトリーに **install-config.yaml** ファイルが作成されます。

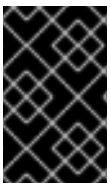
12.5.13. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.5.13.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.18 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.5.13.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.19 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

12.5.13.3. オプションの設定パラメーター

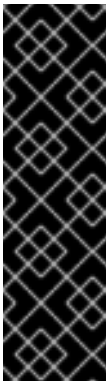
オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.20 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

12.5.13.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表12.21 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。

パラメーター	説明	値
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

12.5.13.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表12.22 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
compute.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
compute.platform.openstack.rootVolume.zones	コンピュートマシンの root ボリュームをインストールするアベイラビリティーゾーン。このパラメーターに値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

パラメーター	説明	値
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.rootVolume.zones	コントロールプレーンマシンの root ボリュームをインストールするアベイラビリティーゾーン。この値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform.openstack.clusterOSImageProperties	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアの一覧。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.ingressFloatingIP	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform.openstack.apiFloatingIP	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

12.5.13.6. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

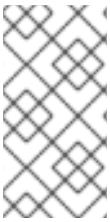
カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。

- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。

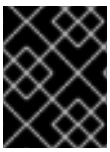


注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

12.5.13.7. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。



重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

12.5.13.8. マシンのカスタムサブネットの設定

インストールプログラムがデフォルトで使用する IP 範囲は、OpenShift Container Platform のインストール時に作成する Neutron サブネットと一致しない可能性があります。必要な場合は、インストール設定ファイルを編集して、新規マシンの CIDR 値を更新します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; (1)
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

(1) 必要な Neutron サブネットに一致する値 (例: **192.0.2.0/24**) を挿入します。

   - 値を手動で設定するには、ファイルを開き、**networking.machineCIDR** の値を必要な Neutron サブネットに一致する値に設定します。

12.5.13.9. コンピュートマシンプールを空にする

独自のインフラストラクチャーを使用するインストールを実行するには、インストール設定ファイルのコンピュートマシンの数をゼロに設定します。その後、これらのマシンを手動で作成します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

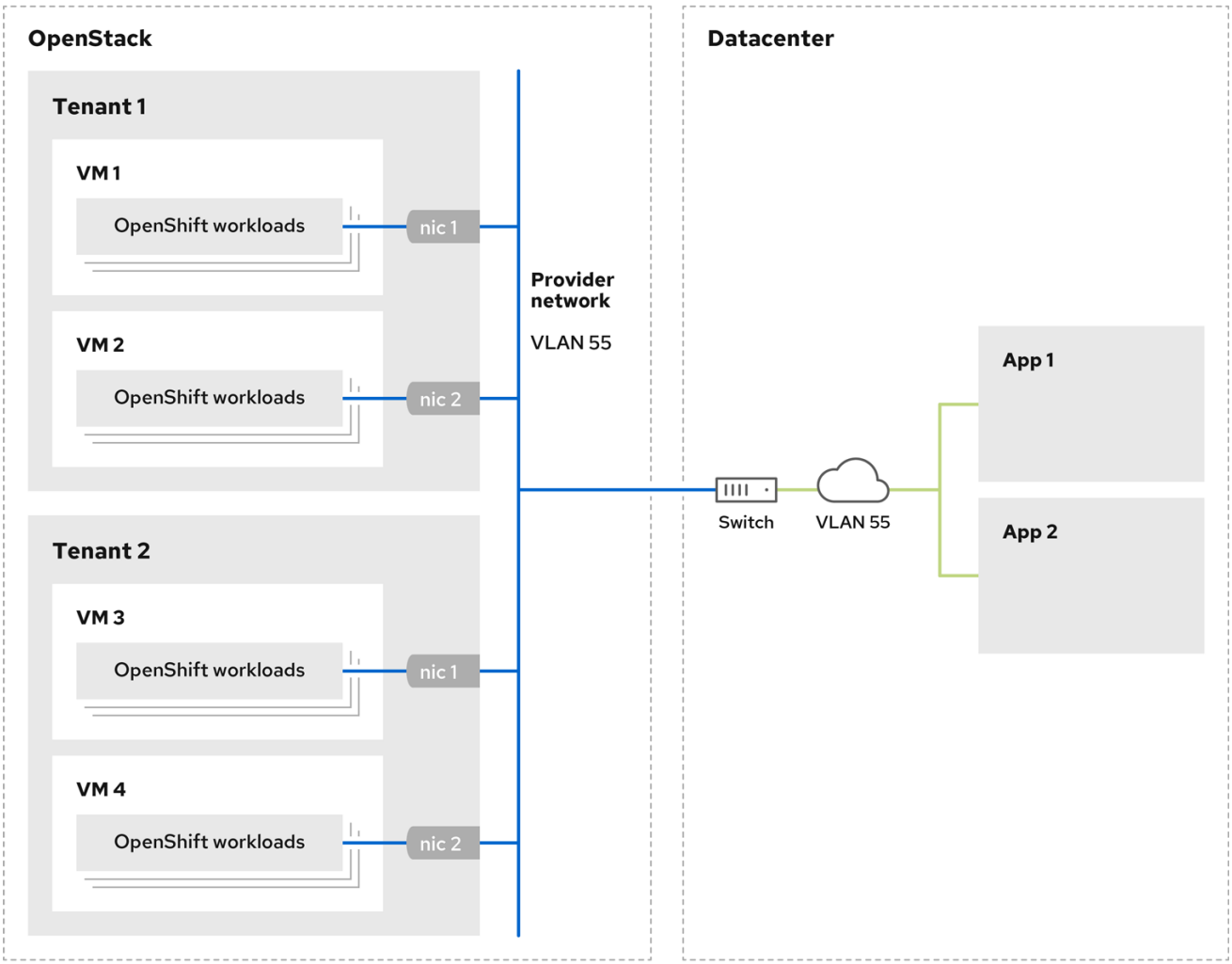
   - 値を手動で設定するには、ファイルを開き、**compute.<first entry>.replicas** の値を **0** に設定します。

12.5.13.10. RHOSP プロバイダーネットワーク上のクラスターデプロイメント

プロバイダーネットワーク上のプライマリーネットワークインターフェイスを使用して、OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイできます。プロバイダーネットワークは一般的に、インターネットへの到達に使用可能なパブリックネットワークに、プロジェクトが直接アクセスできるように使用します。ネットワーク作成プロセスの一環として、プロバイダーネットワークをプロジェクト間で共有することもできます。

RHOSP プロバイダーネットワークは、データセンター内の既存の物理ネットワークに直接マップします。RHOSP 管理者はこれらを作成する必要があります。

以下の例では、OpenShift Container Platform ワークロードはプロバイダーネットワークを使用してデータセンターに接続されます。



プロバイダーネットワークにインストールされている OpenShift Container Platform クラスターは、テナントネットワークまたは Floating IP アドレスを必要としません。インストーラーは、インストール中にこれらのリソースを作成しません。

プロバイダーネットワークタイプの例には、フラット (タグなし) および VLAN (802.1Q タグ付き) が含まれます。

注記

クラスターは、ネットワークタイプが許可する限り多くのプロバイダーネットワーク接続をサポートできます。たとえば、VLAN ネットワークは、通常最大 4096 の接続をサポートします。

プロバイダーネットワークおよびテナントネットワークの詳細は、RHOSP のドキュメント を参照してください。

12.5.13.10.1. クラスターのインストールにおける RHOSP プロバイダーネットワーク要件

OpenShift Container Platform クラスターをインストールする前に、Red Hat OpenStack Platform (RHOSP) のデプロイメントおよびプロバイダーネットワークは、さまざまな条件を満たす必要があります。

- RHOSP ネットワークサービス (Neutron) が有効化され、RHOSP ネットワーク API 経由でアクセス可能であること。
- RHOSP ネットワークサービスでは、ポートセキュリティーと許可するアドレスペアの機能拡張が有効化 されていること。
- プロバイダーネットワークは他のテナントと共有できます。

  ヒント

  --share フラグを指定して **openstack network create** コマンドを使用して、共有できるネットワークを作成します。

- クラスターのインストールに使用する RHOSP プロジェクトは、プロバイダーネットワークと適切なサブネットを所有する必要があります。

  ヒント

  openshift という名前のプロジェクトのネットワークを作成するには、以下のコマンドを入力します。

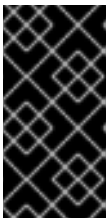
  ```
  $ openstack network create --project openshift
  ```

  openshift という名前のプロジェクトのサブネットを作成するには、以下のコマンドを入力します。

  ```
  $ openstack subnet create --project openshift
  ```

  RHOSP でのネットワークの作成に関する詳細は、プロバイダーネットワークに関するドキュメント を参照してください。

  クラスターが **admin** ユーザーによって所有されている場合、そのユーザーとしてインストーラーを実行してネットワーク上でポートを作成する必要があります。

  重要

  プロバイダーネットワークは、クラスターの作成に使用される RHOSP プロジェクトによって所有されている必要があります。所有されていない場合は、RHOSP Compute サービス (Nova) はそのネットワークからポートを要求できません。

- プロバイダーネットワークが、デフォルトで **169.254.169.254** である RHOSP メタデータサービスの IP アドレスに到達できることを確認します。
  RHOSP SDN とネットワークサービス設定によっては、サブネットを作成する際に、ルートを提供しなければならない場合があります。以下に例を示します。

  ```
  $ openstack subnet create --dhcp --host-route
  destination=169.254.169.254/32,gateway=192.0.2.2 ...
  ```

- オプション: ネットワークのセキュリティーを保護するには、単一のプロジェクトへのネットワークアクセスを制限する ロールベースのアクセス制御 (RBAC) ルールを作成します。

12.5.13.10.2. プロバイダーネットワークにプライマリーインターフェイスを持つクラスターのデプロイ

Red Hat OpenStack Platform (RHOSP) プロバイダーネットワーク上にプライマリーネットワークインターフェイスを持つ OpenShift Container Platform クラスターをデプロイすることができます。

- クラスターのインストールにおける RHOSP プロバイダーネットワーク要件に記載されているとおりに、お使いの Red Hat OpenStack Platform (RHOSP) のデプロイメントが設定されています。

手順

1. テキストエディターで **install-config.yaml** ファイルを開きます。
2. **platform.openstack.apiVIP** プロパティーの値を API VIP の IP アドレスに設定します。
3. **platform.openstack.ingressVIP** プロパティーの値を Ingress VIP の IP アドレスに設定します。
4. **platform.openstack.machinesSubnet** プロパティーの値をプロバイダーネットワークサブネットの UUID に設定します。
5. **networking.machineNetwork.cidr** プロパティーの値をプロバイダーネットワークサブネットの CIDR ブロックに設定します。

重要

platform.openstack.apiVIP プロパティーおよび **platform.openstack.ingressVIP** プロパティーはいずれも、**networking.machineNetwork.cidr** ブロックから割り当てられていない IP アドレスである必要があります。

RHOSP プロバイダーネットワークに依存するクラスターのインストール設定ファイルのセクション

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

警告

プライマリーネットワークインターフェイスにプロバイダーネットワークを使用している間は、**platform.openstack.externalNetwork** パラメーターまたは **platform.openstack.externalDNS** パラメーターを設定することはできません。

クラスターをデプロイする際に、インストーラーは **install-config.yaml** ファイルを使用してプロバイダーネットワークにクラスターをデプロイします。

ヒント

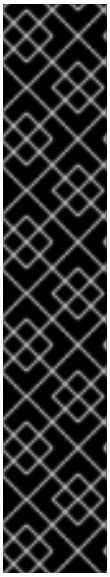
プロバイダーネットワークを含むネットワークを **platform.openstack.additionalNetworkIDs** 一覧に追加できます。

クラスターのデプロイ後に、Pod を追加のネットワークに接続することができます。詳細は、複数ネットワークについて を参照してください。

12.5.14. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

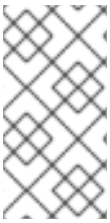
一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。
   -

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

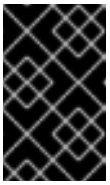
```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. メタデータファイルの **infraID** キーを環境変数としてエクスポートします。

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

ヒント

metadata.json から **infraID** キーを抽出し、作成するすべての RHOSP リソースの接頭辞として使用します。これを実行することで、同じプロジェクトで複数のデプロイメントを実行する際に名前の競合が発生しないようにします。

12.5.15. ブートストラップ Ignition ファイルの準備

OpenShift Container Platform インストールプロセスは、ブートストラップ Ignition 設定ファイルから作成されるブートストラップマシンに依存します。

ファイルを編集し、アップロードします。次に、Red Hat OpenStack Platform (RHOSP) がプライマリーファイルをダウンロードする際に使用するセカンダリーブートストラップ Ignition 設定ファイルを作成します。

前提条件

- インストーラープログラムが生成するブートストラップ Ignition ファイル **bootstrap.ign** があります。
- インストーラーのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成** を参照してください。
- HTTP(S) でアクセス可能な方法でブートストラップ Ignition ファイルを保存できます。
  - 記載された手順では RHOSP イメージサービス (Glance) を使用しますが、RHOSP ストレージサービス (Swift)、Amazon S3、内部 HTTP サーバー、またはアドホックの Nova サーバーを使用することもできます。

手順

1. 以下の Python スクリプトを実行します。スクリプトはブートストラップ Ignition ファイルを変更して、ホスト名および利用可能な場合は、実行時の CA 証明書ファイルを設定します。

```
import base64
```

```
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

2. RHOSP CLI を使用して、ブートストラップ Ignition ファイルを使用するイメージを作成します。

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. イメージの詳細を取得します。

```
$ openstack image show <image_name>
```

file 値をメモします。これは **v2/images/<image_ID>/file** パターンをベースとしています。

注記

作成したイメージがアクティブであることを確認します。

4. イメージサービスのパブリックアドレスを取得します。

```
$ openstack catalog show image
```

5. パブリックアドレスとイメージ **file** 値を組み合わせ、結果を保存場所として保存します。この場所は、**<image_service_public_URL>/v2/images/<image_ID>/file** パターンをベースとしています。

6. 認証トークンを生成し、トークン ID を保存します。

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** というファイルに以下のコンテンツを挿入し、独自の値に一致するようにプレースホルダーを編集します。

```
{
  "ignition": {
    "config": {
      "merge": [{
        "source": "<storage_url>", 1
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>" 4
        }]
      }
    },
    "version": "3.2.0"
  }
}
```

1. **ignition.config.append.source** の値をブートストラップ Ignition ファイルのストレージ URL に置き換えます。
2. **httpHeaders** の **name** を **"X-Auth-Token"** に設定します。
3. **httpHeaders** の **value** をトークンの ID に設定します。
4. ブートストラップ Ignition ファイルサーバーが自己署名証明書を使用する場合は、base64 でエンコードされた証明書を含めます。

8. セカンダリー Ignition 設定ファイルを保存します。

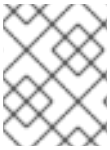
ブートストラップ Ignition データはインストール時に RHOSP に渡されます。

警告

ブートストラップ Ignition ファイルには、**clouds.yaml** 認証情報などの機密情報が含まれます。これを安全な場所に保存し、インストールプロセスの完了後に削除します。

12.5.16. RHOSP でのコントロールプレーンの Ignition 設定ファイルの作成

独自のインフラストラクチャーを使用して OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールするには、コントロールプレーンの Ignition 設定ファイルが必要です。複数の設定ファイルを作成する必要があります。



注記

ブートストラップ Ignition 設定と同様に、各コントロールプレーンマシンのホスト名を明示的に定義する必要があります。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。

手順

- コマンドラインで、以下の Python スクリプトを実行します。

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
storage['files'] = files;
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

以下の 3 つのコントロールプレーン Ignition ファイルが作成されます。**<INFRA_ID>-master-0-ignition.json**、**<INFRA_ID>-master-1-ignition.json**、および **<INFRA_ID>-master-2-ignition.json**。

12.5.17. RHOSP でのネットワークリソースの作成

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを作成します。時間を節約するには、セキュリティーグループ、ネットワーク、サブネット、ルーター、およびポートを生成する指定された Ansible Playbook を実行します。

前提条件

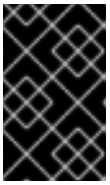
- Python 3 がマシンにインストールされている。
- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。

手順

1. オプション: 外部ネットワークの値を **inventory.yaml** Playbook に追加します。

   inventory.yaml Ansible Playbook の外部ネットワーク値の例

   ```
   ...
       # The public network providing connectivity to the cluster. If not
       # provided, the cluster external connectivity must be provided in another
       # way.

       # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
       os_external_network: 'external'
   ...
   ```

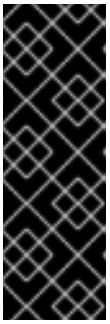
   重要

   inventory.yaml ファイルの **os_external_network** の値を指定しなかった場合は、仮想マシンが Glance および外部接続にアクセスできるようにする必要があります。

2. オプション: 外部ネットワークおよび Floating IP (FIP) アドレスの値を **inventory.yaml** Playbook に追加します。

   inventory.yaml Ansible Playbook の FIP 値の例

   ```
   ...
       # OpenShift API floating IP address. If this value is non-empty, the
       # corresponding floating IP will be attached to the Control Plane to
       # serve the OpenShift API.
       os_api_fip: '203.0.113.23'

       # OpenShift Ingress floating IP address. If this value is non-empty, the
       # corresponding floating IP will be attached to the worker nodes to serve
       # the applications.
       os_ingress_fip: '203.0.113.19'

       # If this value is non-empty, the corresponding floating IP will be
       # attached to the bootstrap machine. This is needed for collecting logs
       # in case of install failure.
       os_bootstrap_fip: '203.0.113.20'
   ```



重要

os_api_fip および **os_ingress_fip** の値を定義しない場合、インストール後のネットワーク設定を実行する必要があります。

os_bootstrap_fip の値を定義しない場合、インストーラーは失敗したインストールからデバッグ情報をダウンロードできません。

詳細は、環境へのアクセスの有効化を参照してください。

3. コマンドラインで、**security-groups.yaml** Playbook を実行してセキュリティーグループを作成します。

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. コマンドラインで、**network.yaml** Playbook を実行して、ネットワーク、サブネット、およびルーターを作成します。

```
$ ansible-playbook -i inventory.yaml network.yaml
```

5. オプション: Nova サーバーが使用するデフォルトのリゾルバーを制御する必要がある場合は、RHOSP CLI コマンドを実行します。

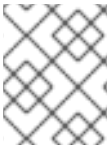
```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

オプションで、作成した **inventory.yaml** ファイルを使用してインストールをカスタマイズできます。たとえば、ベアメタルマシンを使用するクラスターをデプロイすることができます。

12.5.17.1. ベアメタルマシンを使用したクラスターのデプロイ

クラスターがベアメタルマシンを使用する必要がある場合は、**inventory.yaml** ファイルを変更します。クラスターには、ベアメタル上でコントロールプレーンとコンピュートマシンの両方を実行させることも、コンピュートマシンのみを実行させることもできます。

ベアメタルコンピュートマシンは、Kuryr を使用するクラスターではサポートされません。



注記

install-config.yaml ファイルで、ベアメタルワーカーに使用する RHOSP ネットワークが Floating IP アドレスをサポートするかどうかが反映されていることを確認します。

前提条件

- RHOSP の ベアメタルサービス (Ironic) は有効にされており、RHOSP Compute API でアクセスできる。
- ベアメタルは RHOSP フレーバー として利用可能である。
- RHOSP ネットワークは、仮想マシンとベアメタルサーバー接続の両方をサポートする。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。

- マシンを既存のネットワークにデプロイする必要がある場合、RHOSP サブネットがプロビジョニングされる。
- マシンをインストーラーでプロビジョニングされるネットワークに場合、RHOSP ベアメタルサービス (Ironic) はテナントネットワークで実行される Preboot eXecution Environment (PXE) ブートマシンをリッスンし、これと対話できます。
- **inventory.yaml** ファイルを OpenShift Container Platform インストールプロセスの一部として作成している。

手順

1. **inventory.yaml** ファイルで、マシンのフレーバーを編集します。
   a. ベアメタルコントロールプレーンマシンを使用する場合は、**os_flavor_master** の値をベアメタルフレーバーに変更します。
   b. **os_flavor_worker** の値をベアメタルフレーバーに変更します。

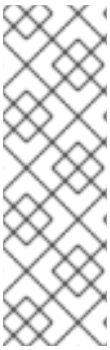
   ベアメタルの inventory.yaml のサンプルファイル

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
...
```

1 ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、この値をベアメタルのフレーバーに変更します。

2 この値を、コンピュートマシンに使用するベアメタルのフレーバーに変更します。

更新された **inventory.yaml** ファイルを使用してインストールプロセスを完了します。デプロイメント時に作成されるマシンは、ファイルに追加したフレーバーを使用します。



注記

インストーラーは、ベアメタルマシンの起動中にタイムアウトする可能性があります。

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
./openshift-install wait-for install-complete --log-level debug
```

12.5.18. RHOSP でのブートストラップマシンの作成

ブートストラップマシンを作成し、これに Red Hat OpenStack Platform (RHOSP) で実行するために必要なネットワークアクセスを付与します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **bootstrap.yaml** Ansible Playbook は共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. ブートストラップサーバーがアクティブになった後に、ログを表示し、Ignition ファイルが受信されたことを確認します。

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

12.5.19. RHOSP でのコントロールプレーンの作成

生成した Ignition 設定ファイルを使用して 3 つのコントロールプレーンマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
- **inventory.yaml**、 **common.yaml**、および **control-plane.yaml** Ansible Playbook は共通ディレクトリーにあります。
- コントロールプレーンの Ignition 設定ファイルの作成で作成された 3 つの Ignition ファイルがある。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。

2. コントロールプレーン Ignition 設定ファイルが作業ディレクトリーにない場合、それらをここにコピーします。

3. コマンドラインで、**control-plane.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

4. 以下のコマンドを実行してブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete
```

コントロールプレーンマシンが実行され、クラスターに参加していることを確認できるメッセージが表示されます。

```
INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

12.5.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.5.21. RHOSP からのブートストラップリソースの削除

不要になったブートストラップリソースを削除します。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、**common.yaml**、および **down-bootstrap.yaml** Ansible Playbook が共通ディレクトリーにある。
- コントロールプレーンマシンが実行中である。
  - マシンのステータスが不明な場合は、クラスターステータスの確認を参照してください。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**down-bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

ブートストラップポート、サーバー、および Floating IP アドレスが削除されます。

警告

ブートストラップ Ignition ファイル URL をまだ無効にしていない場合は、無効にしてください。

12.5.22. RHOSP でのコンピュートマシンの作成

コントロールプレーンの起動後、コンピュートマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **compute-nodes.yaml** Ansible Playbook が共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。
- コントロールプレーンが有効である。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

次のステップ

- マシンの証明書署名要求を承認します。

12.5.23. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

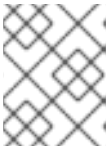
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

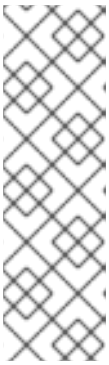
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

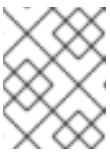
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                    CONDITION
```

```
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

12.5.24. インストールの正常な実行の確認

OpenShift Container Platform のインストールが完了していることを確認します。

前提条件

- インストールプログラム (**openshift-install**) があります。

手順

- コマンドラインで、以下を入力します。

```
$ openshift-install --log-level debug wait-for install-complete
```

プログラムはコンソール URL と管理者のログイン情報を出力します。

12.5.25. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.5.26. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラストラフィックを設定 します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.6. 独自のインフラストラクチャーでの KURYR を使用する OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、ユーザーによってプロビジョニングされたインフラストラクチャーを実行するクラスターを Red Hat OpenStack Platform (RHOSP) にインストールできます。

独自のインフラストラクチャーを使用することで、クラスターを既存のインフラストラクチャーおよび変更と統合できます。このプロセスでは、インストーラーでプロビジョニングされるインストールの場合よりも多くの手作業が必要になります。Nova サーバー、Neutron ポート、セキュリティーグループなどのすべての RHOSP リソースを作成する必要があるためです。ただし、Red Hat では、デプロイメントプロセスを支援する Ansible Playbook を提供しています。

12.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- OpenShift クラスターでサポートされるプラットフォーム のセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認している。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- OpenShift Container Platform のインストール先に RHOSP アカウントがある。
- インストールプログラムを実行するマシンには、以下が含まれる。
  - インストールプロセス時に作成したファイルを保持できる単一ディレクトリー
  - Python 3

12.6.2. Kuryr SDN について

Kuryr は、Neutron および Octavia Red Hat OpenStack Platform (RHOSP) サービスを使用して Pod およびサービスのネットワークを提供する Container Network Interface (CNI) プラグインです。

Kuryr と OpenShift Container Platform の統合は主に、RHOSP の仮想マシンで実行する OpenShift Container Platform クラスター用に設計されました。Kuryr は、OpenShift Container Platform Pod を RHOSP SDN にプラグインしてネットワークのパフォーマンスを強化します。さらに、これは Pod と RHOSP 仮想インスタンス間の接続を可能にします。

Kuryr コンポーネントは **openshift-kuryr** namespace を使用して OpenShift Container Platform の Pod としてインストールされます。

- **kuryr-controller**: **master** ノードにインストールされる単一のサービスインスタンスです。これは、OpenShift Container Platform で **Deployment** としてモデリングされます。
- **kuryr-cni**: 各 OpenShift Container Platform ノードで Kuryr を CNI ドライバーとしてインストールし、設定するコンテナーです。これは、OpenShift Container Platform で **DaemonSet** オブジェクトとしてモデリングされます。

Kuryr コントローラーは OpenShift Container Platform API サーバーで Pod、サービスおよび namespace の作成、更新、および削除イベントについて監視します。これは、OpenShift Container Platform API 呼び出しを Neutron および Octavia の対応するオブジェクトにマップします。そのため、Neutron トランクポート機能を実装するすべてのネットワークソリューションを使用して、Kuryr 経由で OpenShift Container Platform をサポートすることができます。これには、Open vSwitch (OVS) および Open Virtual Network (OVN) などのオープンソースソリューションや Neutron と互換性のある市販の SDN が含まれます。

Kuryr は、カプセル化された RHOSP テナントネットワーク上の OpenShift Container Platform デプロイメントに使用することが推奨されています。これは、 RHOSP ネットワークでカプセル化された OpenShift Container Platform SDN を実行するなど、二重のカプセル化を防ぐために必要です。

プロバイダーネットワークまたはテナント VLAN を使用する場合は、二重のカプセル化を防ぐために Kuryr を使用する必要はありません。パフォーマンス上の利点はそれほど多くありません。ただし、設定によっては、Kuryr を使用して 2 つのオーバーレイが使用されないようにすることには利点がある場合があります。

Kuryr は、以下のすべての基準が true であるデプロイメントでは推奨されません。

- RHOSP のバージョンが 16 よりも前のバージョンである。

- デプロイメントで UDP サービスが使用されているか、または少数のハイパーバイザーで多数の TCP サービスが使用されている。

または、以下を実行します。

- **ovn-octavia** Octavia ドライバーが無効にされている。
- デプロイメントで、少数のハイパーバイザーで多数の TCP サービスが使用されている。

12.6.3. Kuryr を使用して OpenShift Container Platform を RHOSP にインストールするためのリソースのガイドライン

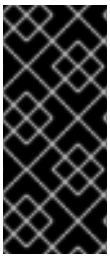
Kuryr SDN を使用する場合、Pod、サービス、namespace およびネットワークポリシーは RHOSP クォータのリソースを使用します。これにより、最小要件が増加します。また、Kuryr にはデフォルトインストールに必要な要件以外の追加要件があります。

以下のクォータを使用してデフォルトのクラスターの最小要件を満たすようにします。

表12.23 Kuryr を使用する RHOSP のデフォルト OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3: LoadBalancer タイプに予想されるサービス数
ポート	1500: Pod ごとに 1 つ必要
ルーター	1
サブネット	250: namespace/プロジェクトごとに 1 つ必要
ネットワーク	250: namespace/プロジェクトごとに 1 つ必要
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	250: サービスおよび NetworkPolicy ごとに 1 つ必要
セキュリティーグループルール	1000
ロードバランサー	100: サービスごとに 1 つ必要
ロードバランサーリスナー	500: サービスで公開されるポートごとに 1 つ必要
ロードバランサーノード	500: サービスで公開されるポートごとに 1 つ必要

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

重要

OVN Octavia ドライバーではなく Amphora ドライバーで Red Hat OpenStack Platform(RHOSP) バージョン 16 を使用している場合、セキュリティーグループはユーザープロジェクトではなくサービスアカウントに関連付けられます。

リソースを設定する際には、以下の点に注意してください。

- 必要なポート数は Pod 数よりも大きくなる。Kuryr はポートプールを使用して、事前に作成済みのポートを Pod で使用できるようにし、Pod の起動時間を短縮します。
- 各ネットワークポリシーは RHOSP セキュリティーグループにマップされ、**NetworkPolicy** 仕様によっては 1 つ以上のルールがセキュリティーグループに追加される。
- 各サービスは RHOSP ロードバランサーにマップされる。クォータに必要なセキュリティーグループの数を見積もる場合には、この要件を考慮してください。
  RHOSP バージョン 15 以前のバージョン、または **ovn-octavia driver** を使用している場合、各ロードバランサーにはユーザープロジェクトと共にセキュリティーグループがあります。
- クォータはロードバランサーのリソース (VM リソースなど) を考慮しませんが、RHOSP デプロイメントのサイズを決定する際にはこれらのリソースを考慮する必要があります。デフォルトのインストールには 50 を超えるロードバランサーが あり、クラスターはそれらのロードバランサーに対応できる必要があります。
  OVN Octavia ドライバーを有効にして RHOSP バージョン 16 を使用している場合は、1 つのロードバランサー仮想マシンのみが生成され、サービスは OVN フロー経由で負荷分散されます。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

Kuryr SDN を有効にするには、使用する環境が以下の要件を満たしている必要があります。

- RHOSP 13+ を実行します。
- オーバークラウドと Octavia を使用します。
- Neutron トランクポートの拡張を使用します。
- ML2/OVS Neutron ドライバーが **ovs-hybrid** の代わりに使用れる場合、**openvswitch** ファイアウォールドライバーを使用します。

12.6.3.1. クォータの拡大

Kuryr SDN を使用する場合、Pod、サービス、namespace、およびネットワークポリシーが使用する Red Hat OpenStack Platform (RHOSP) リソースに対応するためにクォータを引き上げる必要があります。

手順

- 以下のコマンドを実行して、プロジェクトのクォータを増やします。

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets 250 --networks 250 <project>
```

12.6.3.2. Neutron の設定

Kuryr CNI は Neutron トランクの拡張を使用してコンテナーを Red Hat OpenStack Platform (RHOSP) SDN にプラグインします。したがって、Kuryr が適切に機能するには **trunks** 拡張を使用する必要があります。

さらにデフォルトの ML2/OVS Neutron ドライバーを使用する場合には、セキュリティーグループがトランクサブポートで実行され、Kuryr がネットワークポリシーを適切に処理できるように、**ovs_hybrid** ではなく **openvswitch** に設定される必要があります。

12.6.3.3. Octavia の設定

Kuryr SDN は Red Hat OpenStack Platform (RHOSP) の Octavia LBaaS を使用して OpenShift Container Platform サービスを実装します。したがって、Kuryr SDN を使用するように RHOSP に Octavia コンポーネントをインストールし、設定する必要があります。

Octavia を有効にするには、Octavia サービスを RHOSP オーバークラウドのインストール時に組み込むか、またはオーバークラウドがすでに存在する場合は Octavia サービスをアップグレードする必要があります。Octavia を有効にする以下の手順は、オーバークラウドのクリーンインストールまたはオーバークラウドの更新の両方に適用されます。

注記

以下の手順では、Octavia を使用する場合に RHOSP のデプロイメント 時に必要となる主な手順のみを説明します。また、レジストリーメソッド が変更されることにも留意してください。

以下の例では、ローカルレジストリーの方法を使用しています。

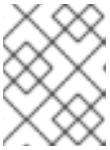
手順

1. ローカルレジストリーを使用している場合、イメージをレジストリーにアップロードするためのテンプレートを作成します。以下に例を示します。

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{product-version} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. **local_registry_images.yaml** ファイルに Octavia イメージが含まれることを確認します。以下に例を示します。

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```



注記

Octavia コンテナーのバージョンは、インストールされている特定の RHOSP リリースによって異なります。

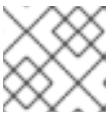
3. コンテナーイメージを **registry.redhat.io** からアンダークラウドノードにプルします。

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

これには、ネットワークおよびアンダークラウドディスクの速度に応じて多少の時間がかかる可能性があります。

4. Octavia ロードバランサーは OpenShift Container Platform API にアクセスするために使用されるため、それらのリスナーの接続のデフォルトタイムアウトを増やす必要があります。デフォルトのタイムアウトは 50 秒です。以下のファイルをオーバークラウドのデプロイコマンドに渡し、タイムアウトを 20 分に増やします。

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```



注記

これは RHOSP 13.0.13+ では不要です。

5. Octavia を使用してオーバークラウドをインストールまたは更新します。

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

注記

このコマンドには、Octavia に関連付けられたファイルのみが含まれます。これは、RHOSP の特定のインストールによって異なります。詳細は RHOSP のドキュメントを参照してください。Octavia インストールのカスタマイズについての詳細は、Octavia デプロイメントのプランニング を参照してください。

注記

Kuryr SDN を利用する際には、オーバークラウドのインストールに Neutron の **trunk** 拡張機能が必要です。これは、Director デプロイメントでデフォルトで有効にされます。Neutron バックエンドが ML2/OVS の場合、デフォルトの **ovs-hybrid** の代わりに **openvswitch** ファイアウォールを使用します。バックエンドが ML2/OVN の場合には変更の必要がありません。

6. RHOSP の 13.0.13 よりも前のバージョンでは、プロジェクトの作成後にプロジェクト ID を **octavia.conf** 設定ファイルに追加します。

- トラフィックが Octavia ロードバランサーを通過する場合など、複数のサービス全体でネットワークポリシーを実行するには、Octavia がユーザープロジェクトで Amphora 仮想マシンセキュリティーグループを作成するようにする必要があります。
この変更により、必要なロードバランサーのセキュリティーグループがそのプロジェクトに属し、それらをサービスの分離を実行するように更新できます。

注記

RHOSP の 13.0.13 よりも後のバージョンでは、このタスクは必要ありません。

Octavia は、ロードバランサー VIP へのアクセスを制限する新しい ACL API を実装します。

a. プロジェクト ID を取得します。

```
$ openstack project show <project>
```

出力例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | PROJECT_ID                       |
| is_domain   | False                            |
| name        | *<project>*                      |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

b. プロジェクト ID をコントローラーの **octavia.conf** に追加します。

i. **stackrc** ファイルを取得します。

```
$ source stackrc  # Undercloud credentials
```

ii. オーバークラウドコントローラーを一覧表示します。

```
$ openstack server list
```

出力例

```
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| ID                                   | Name        | Status | Networks
| Image         | Flavor     |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0    | ACTIVE |
ctlplane=192.168.24.6 | overcloud-full | compute    |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
```

iii. コントローラーに対して SSH を実行します。

```
$ ssh heat-admin@192.168.24.8
```

iv. **octavia.conf** ファイルを編集して、プロジェクトを Amphora セキュリティーグループがユーザーのアカウントに設定されているプロジェクトの一覧に追加します。

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

c. 新しい設定が読み込まれるように Octavia ワーカーを再起動します。

```
controller-0$ sudo docker restart octavia_worker
```

注記

RHOSP 環境によっては、Octavia は UDP リスナーをサポートしない可能性があります。RHOSP の 13.0.13 よりも前のバージョンで Kuryr SDN を使用する場合、UDP サービスはサポートされません。RHOSP バージョン 16 以降は UDP をサポートします。

12.6.3.3.1. Octavia OVN ドライバー

Octavia は Octavia API を使用して複数のプロバイダードライバーをサポートします。

利用可能なすべての Octavia プロバイダードライバーをコマンドラインで表示するには、以下を入力します。

```
$ openstack loadbalancer provider list
```

出力例

```
+---------+-------------------------------------------------+
| name    | description                                     |
+---------+-------------------------------------------------+
| amphora | The Octavia Amphora driver.                     |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn     | Octavia OVN driver.                             |
+---------+-------------------------------------------------+
```

RHOSP バージョン 16 以降、Octavia OVN プロバイダードライバー (**ovn**) は RHOSP デプロイメントの OpenShift Container Platform でサポートされます。

ovn は、Octavia および OVN が提供する負荷分散用の統合ドライバーです。これは基本的な負荷分散機能をサポートし、OpenFlow ルールに基づいています。このドライバーは、OVN Neutron ML2 を使用するデプロイメント上の director により Octavia で自動的に有効にされます。

Amphora プロバイダードライバーがデフォルトのドライバーです。ただし、**ovn** が有効にされる場合には、Kuryr がこれを使用します。

Kuryr が Amphora の代わりに **ovn** を使用する場合は、以下の利点があります。

- リソース要件が減少します。Kuryr は、各サービスにロードバランサーの仮想マシンを必要としません。
- ネットワークレイテンシーが短縮されます。
- サービスごとに仮想マシンを使用する代わりに、OpenFlow ルールを使用することで、サービスの作成速度が上がります。
- Amphora 仮想マシンで集中管理されるのではなく、すべてのノードに分散負荷分散アクションが分散されます。

12.6.3.4. Kuryr を使用したインストールについての既知の制限

OpenShift Container Platform を Kuryr SDN で使用する場合、いくつかの既知の制限があります。

RHOSP の一般的な制限
OpenShift Container Platform を Kuryr SDN と共に使用する場合は、すべてのバージョンおよび環境に適用されるいくつかの制限があります。

- **NodePort** タイプの **Service** オブジェクトはサポートされません。
- OVN Octavia プロバイダーを使用するクラスターは、**Service** オブジェクトをサポートします。このオブジェクトについて、**.spec.selector** プロパティーは、**Endpoints** オブジェクトの **.subsets.addresses** プロパティーにノードまたは Pod のサブネットが含まれる場合は指定されません。

- マシンが作成されるサブネットがルーターに接続されていない場合や、サブネットが接続されていても、ルーターに外部ゲートウェイが設定されていない場合、Kuryr はタイプが **LoadBalancer** の **Service** オブジェクトの Floating IP を作成できません。

- **Service** オブジェクトで **sessionAffinity=ClientIP** プロパティーを設定しても効果はありません。Kuryr はこの設定をサポートしていません。

RHOSP バージョンの制限
OpenShift Container Platform を Kuryr SDN で使用する場合は、RHOSP バージョンに依存するいくつかの制限があります。

- RHOSP の 16 よりも前のバージョンでは、デフォルトの Octavia ロードバランサードライバー (Amphora) を使用します。このドライバーでは、OpenShift Container Platform サービスごとに 1 つの Amphora ロードバランサー仮想マシンをデプロイする必要があります。サービス数が多すぎると、リソースが不足する可能性があります。
  OVN Octavia ドライバーが無効にされている以降のバージョンの RHOSP のデプロイメントでも Amphora ドライバーを使用します。この場合も、RHOSP の以前のバージョンと同じリソースに関する懸念事項を考慮する必要があります。

- バージョン 13.0.13 よりも前の Octavia RHOSP バージョンは UDP リスナーをサポートしません。そのため、OpenShift Container Platform UDP サービスはサポートされません。

- 13.0.13 よりも前の Octavia RHOSP バージョンは、同じポートで複数のプロトコルをリッスンできません。TCP や UDP など、同じポートを異なるプロトコルに公開するサービスはサポートされません。

- Kuryr SDN は、サービスによる自動解凍をサポートしていません。

RHOSP 環境の制限
Kuryr SDN を使用する場合に、デプロイメント環境に依存する制限事項があります。

Octavia には UDP プロトコルおよび複数のリスナーのサポートがないため、RHOCP バージョンが 13.0.13 よりも前のバージョンの場合、Kuryr は Pod が DNS 解決に TCP を使用するように強制します。

Go バージョン 1.12 以前では、CGO サポートが無効にされた状態でコンパイルされたアプリケーションは UDP のみを使用します。この場合、ネイティブの Go リゾルバーは、TCP が DNS 解決に強制的に実行されるかどうかを制御する、**resolv.conf** の **use-vc** オプションを認識しません。その結果、UDP は引き続き DNS 解決に使用されますが、これは失敗します。

TCP の強制を許可するには、環境変数 **CGO_ENABLED** を **1** に設定 (例: **CGO_ENABLED=1**) されている状態でアプリケーションをコンパイルするか、または変数がないことを確認します。

Go バージョン 1.13 以降では、UDP を使用した DNS 解決が失敗する場合に TCP が自動的に使用されます。

注記

Alpine ベースのコンテナーを含む musl ベースのコンテナーは **use-vc** オプションをサポートしません。

RHOSP のアップグレードの制限
RHOSP のアップグレードプロセスにより、Octavia API が変更され、ロードバランサーに使用される Amphora イメージへのアップグレードが必要になる可能性があります。

API の変更に個別に対応できます。

Amphora イメージがアップグレードされると、RHOSP Operator は既存のロードバランサー仮想マシンを 2 つの方法で処理できます。

- ロードバランサーのフェイルオーバー をトリガーしてそれぞれの仮想マシンをアップグレードします。
- ユーザーが仮想マシンのアップグレードを行う必要があります。

Operator が最初のオプションを選択する場合、フェイルオーバー時に短い時間のダウンタイムが生じる可能性があります。

Operator が 2 つ目のオプションを選択する場合、既存のロードバランサーは UDP リスナーなどのアップグレードされた Octavia API 機能をサポートしません。この場合、ユーザーはこれらの機能を使用するためにサービスを再作成する必要があります。

重要

OpenShift Container Platform が UDP の負荷分散をサポートする新規の Octavia バージョンを検出する場合、これは DNS サービスを自動的に再作成します。サービスの再作成により、サービスのデフォルトが UDP の負荷分散をサポートするようになります。

再作成により、DNS サービスに約 1 分間のダウンタイムが発生します。

12.6.3.5. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.6.3.6. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

12.6.3.7. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

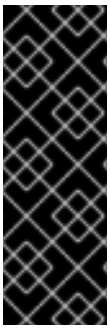
- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.6.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

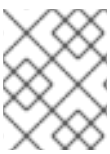


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.6.5. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでのインストールプロセスを単純化する Ansible Playbook には、複数の Python モジュールが必要です。インストーラーを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。



注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされている。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

12.6.6. インストール Playbook のダウンロード

OpenShift Container Platform を独自の Red Hat OpenStack Platform (RHOSP) インフラストラクチャーにインストールするために使用できる Ansible Playbook をダウンロードします。

前提条件

- curl コマンドラインツールがマシンで利用できる。

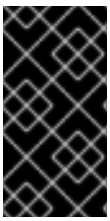
手順

- Playbook を作業ディレクトリーにダウンロードするには、コマンドラインから以下のスクリプトを実行します。

```
$ xargs -n 1 curl -O <<< '
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/bootstrap.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/common.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/compute-nodes.yaml
```

```
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-load-balancers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-containers.yaml'
```

Playbook はマシンにダウンロードされます。



重要

インストールプロセス時に、Playbook を変更してデプロイメントを設定できます。

クラスターの有効期間中に、すべての Playbook を保持します。OpenShift Container Platform クラスターを RHOSP から削除するには Playbook が必要です。

重要

bootstrap.yaml、**compute-nodes.yaml**、**control-plane.yaml**、**network.yaml**、および **security-groups.yaml** ファイルに加えた編集内容は、**down-** の接頭辞が付けられた対応する Playbook に一致している必要があります。たとえば、**bootstrap.yaml** ファイルへの編集は、**down-bootstrap.yaml** ファイルにも反映される必要があります。両方のファイルを編集しない場合、サポートされるクラスターの削除プロセスは失敗します。

12.6.7. インストールプログラムの取得

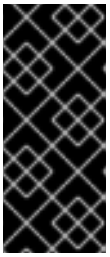
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

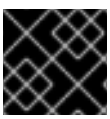
5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.6.8. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.6.9. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成

OpenShift Container Platform インストールプログラムでは、Red Hat Enterprise Linux CoreOS (RHCOS) イメージが Red Hat OpenStack Platform (RHOSP) クラスターに存在する必要があります。最新の RHCOS イメージを取得した後、RHOSP CLI を使用してこれをアップロードします。

前提条件

- RHOSP CLI がインストールされています。

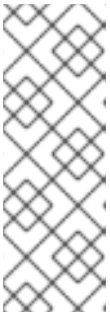
手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、Red Hat Enterprise Linux (RHEL) 8 用の OpenShift Container Platform 4.8 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**をダウンロードします。
4. イメージを展開します。



注記

クラスターが使用する前に RHOSP イメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. ダウンロードしたイメージから、RHOSP CLI を使用して **rhcos** という名前のイメージをクラスターに作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-${RHCOS_VERSION}-openstack.qcow2 rhcos
```

重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

RHOSP にイメージをアップロードした後は、インストールプログラムでイメージを利用できます。

12.6.10. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

12.6.11. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.6.11.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API、クラスターアプリケーション、およびブートストラッププロセスへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. Red Hat OpenStack Platform (RHOSP) CLI を使用して、ブートストラップ FIP を作成します。

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

5. FIP を以下の変数の値として **inventory.yaml** ファイルに追加します。
   - **os_api_fip**
   - **os_bootstrap_fip**
   - **os_ingress_fip**

これらの値を使用する場合には、**inventory.yaml** ファイルの **os_external_network** 変数の値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.6.11.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

inventory.yaml ファイルで、以下の変数を定義しないでください。

- **os_api_fip**
- **os_bootstrap_fip**

- **os_ingress_fip**

外部ネットワークを提供できない場合は、**os_external_network** を空白のままにすることもできます。**os_external_network** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。インストールプロセスで、ネットワークリソースを作成する際に、独自の外部接続を設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムから **wait-for** コマンドでインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。

注記

API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>. IN A <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN A <ingress_port_IP>
```

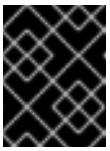
DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.6.12. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
     ```

```
    user_domain_name: Default
    project_domain_name: Default
  dev-env:
   region_name: RegionOne
   auth:
    username: 'devuser'
    password: XXX
    project_name: 'devonly'
    auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.6.13. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

      iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

      iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

      v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

      vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

      vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

      viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

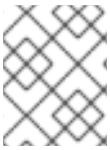
重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

これで、指定したディレクトリーに **install-config.yaml** ファイルが作成されます。

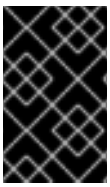
12.6.14. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.6.14.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.24 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.6.14.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.25 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

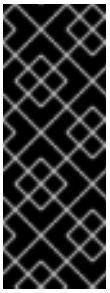
12.6.14.3. オプションの設定パラメーター

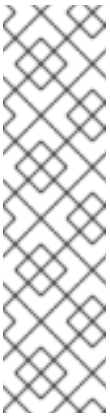
オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.26 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

12.6.14.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表12.27 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。

パラメーター	説明	値
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

12.6.14.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表12.28 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
compute.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
compute.platform.openstack.rootVolume.zones	コンピュートマシンの root ボリュームをインストールするアベイラビリティーゾーン。このパラメーターに値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

パラメーター	説明	値
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.rootVolume.zones	コントロールプレーンマシンの root ボリュームをインストールするアベイラビリティーゾーン。この値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform.openstack.clusterOSImageProperties	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアの一覧。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.ingressFloatingIP	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform.openstack.apiFloatingIP	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

12.6.14.6. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。

- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。
- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。
- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。

注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

12.6.14.7. Kuryr を使用した OpenStack のカスタマイズされた install-config.yaml ファイルのサンプル

デフォルトの OpenShift SDN ではなく Kuryr SDN を使用してデプロイするには、**install-config.yaml** ファイルを変更して **Kuryr** を必要な **networking.networkType** として追加してから、デフォルトの OpenShift Container Platform SDN インストール手順に進む必要があります。このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16 1
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
    trunkSupport: true 2
    octaviaSupport: true 3
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

1 Amphora Octavia ドライバーは、ロードバランサーごとに 2 つのポートを作成します。そのため、インストーラーが作成するサービスサブネットは、**serviceNetwork** プロパティーの値として指定される CIDR のサイズは 2 倍になります。IP アドレスの競合を防ぐには、範囲をより広くする必要があります。

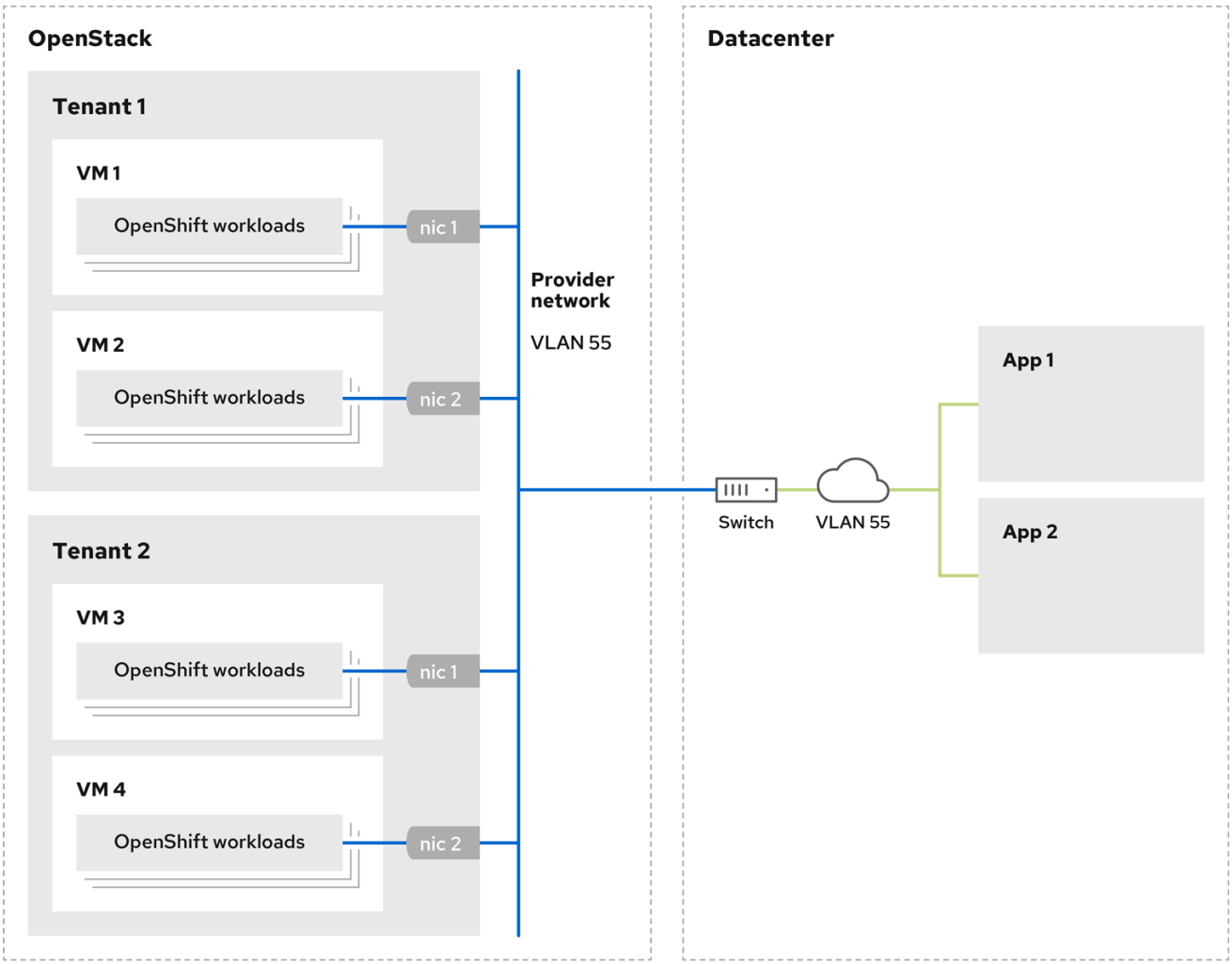
2 3 **trunkSupport** と **octaviaSupport** の両方はインストーラーによって自動的に検出されるため、それらを設定する必要はありません。ただし、ご使用の環境がこれらの両方の要件を満たさないと、Kuryr SDN は適切に機能しません。トランクは Pod を RHOSP ネットワークに接続するために必要であり、Octavia は OpenShift Container Platform サービスを作成するために必要です。

12.6.14.8. RHOSP プロバイダーネットワーク上のクラスターデプロイメント

プロバイダーネットワーク上のプライマリーネットワークインターフェイスを使用して、OpenShift Container Platform クラスターを Red Hat OpenStack Platform (RHOSP) にデプロイできます。プロバイダーネットワークは一般的に、インターネットへの到達に使用可能なパブリックネットワークに、プロジェクトが直接アクセスできるように使用します。ネットワーク作成プロセスの一環として、プロバイダーネットワークをプロジェクト間で共有することもできます。

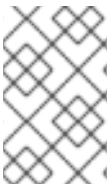
RHOSP プロバイダーネットワークは、データセンター内の既存の物理ネットワークに直接マップします。RHOSP 管理者はこれらを作成する必要があります。

以下の例では、OpenShift Container Platform ワークロードはプロバイダーネットワークを使用してデータセンターに接続されます。



プロバイダーネットワークにインストールされている OpenShift Container Platform クラスターは、テナントネットワークまたは Floating IP アドレスを必要としません。インストーラーは、インストール中にこれらのリソースを作成しません。

プロバイダーネットワークタイプの例には、フラット (タグなし) および VLAN (802.1Q タグ付き) が含まれます。



注記

クラスターは、ネットワークタイプが許可する限り多くのプロバイダーネットワーク接続をサポートできます。たとえば、VLAN ネットワークは、通常最大 4096 の接続をサポートします。

プロバイダーネットワークおよびテナントネットワークの詳細は、RHOSP のドキュメント を参照してください。

12.6.14.8.1. クラスターのインストールにおける RHOSP プロバイダーネットワーク要件

OpenShift Container Platform クラスターをインストールする前に、Red Hat OpenStack Platform (RHOSP) のデプロイメントおよびプロバイダーネットワークは、さまざまな条件を満たす必要があります。

- RHOSP ネットワークサービス (Neutron) が有効化され、RHOSP ネットワーク API 経由でアクセス可能であること。
- RHOSP ネットワークサービスでは、ポートセキュリティーと許可するアドレスペアの機能拡張が有効化 されていること。
- プロバイダーネットワークは他のテナントと共有できます。

  ヒント

  --share フラグを指定して **openstack network create** コマンドを使用して、共有できるネットワークを作成します。

- クラスターのインストールに使用する RHOSP プロジェクトは、プロバイダーネットワークと適切なサブネットを所有する必要があります。

  ヒント

  openshift という名前のプロジェクトのネットワークを作成するには、以下のコマンドを入力します。

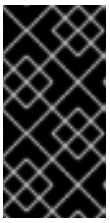
  ```
  $ openstack network create --project openshift
  ```

  openshift という名前のプロジェクトのサブネットを作成するには、以下のコマンドを入力します。

  ```
  $ openstack subnet create --project openshift
  ```

  RHOSP でのネットワークの作成に関する詳細は、プロバイダーネットワークに関するドキュメント を参照してください。

  クラスターが **admin** ユーザーによって所有されている場合、そのユーザーとしてインストーラーを実行してネットワーク上でポートを作成する必要があります。

  重要

  プロバイダーネットワークは、クラスターの作成に使用される RHOSP プロジェクトによって所有されている必要があります。所有されていない場合は、RHOSP Compute サービス (Nova) はそのネットワークからポートを要求できません。

- プロバイダーネットワークが、デフォルトで **169.254.169.254** である RHOSP メタデータサービスの IP アドレスに到達できることを確認します。
  RHOSP SDN とネットワークサービス設定によっては、サブネットを作成する際に、ルートを提供しなければならない場合があります。以下に例を示します。

  ```
  $ openstack subnet create --dhcp --host-route
  destination=169.254.169.254/32,gateway=192.0.2.2 ...
  ```

- オプション: ネットワークのセキュリティーを保護するには、単一のプロジェクトへのネットワークアクセスを制限する ロールベースのアクセス制御 (RBAC) ルールを作成します。

12.6.14.8.2. プロバイダーネットワークにプライマリーインターフェイスを持つクラスターのデプロイ

Red Hat OpenStack Platform (RHOSP) プロバイダーネットワーク上にプライマリーネットワークインターフェイスを持つ OpenShift Container Platform クラスターをデプロイすることができます。

- クラスターのインストールにおける RHOSP プロバイダーネットワーク要件に記載されているとおりに、お使いの Red Hat OpenStack Platform (RHOSP) のデプロイメントが設定されています。

手順

1. テキストエディターで **install-config.yaml** ファイルを開きます。
2. **platform.openstack.apiVIP** プロパティーの値を API VIP の IP アドレスに設定します。
3. **platform.openstack.ingressVIP** プロパティーの値を Ingress VIP の IP アドレスに設定します。
4. **platform.openstack.machinesSubnet** プロパティーの値をプロバイダーネットワークサブネットの UUID に設定します。
5. **networking.machineNetwork.cidr** プロパティーの値をプロバイダーネットワークサブネットの CIDR ブロックに設定します。

重要

platform.openstack.apiVIP プロパティーおよび **platform.openstack.ingressVIP** プロパティーはいずれも、**networking.machineNetwork.cidr** ブロックから割り当てられていない IP アドレスである必要があります。

RHOSP プロバイダーネットワークに依存するクラスターのインストール設定ファイルのセクション

```
...
platform:
  openstack:
    apiVIP: 192.0.2.13
    ingressVIP: 192.0.2.23
    machinesSubnet: fa806b2f-ac49-4bce-b9db-124bc64209bf
    # ...
networking:
  machineNetwork:
  - cidr: 192.0.2.0/24
```

警告

プライマリーネットワークインターフェイスにプロバイダーネットワークを使用している間は、**platform.openstack.externalNetwork** パラメーターまたは **platform.openstack.externalDNS** パラメーターを設定することはできません。

クラスターをデプロイする際に、インストーラーは **install-config.yaml** ファイルを使用してプロバイダーネットワークにクラスターをデプロイします。

ヒント

プロバイダーネットワークを含むネットワークを **platform.openstack.additionalNetworkIDs** 一覧に追加できます。

クラスターのデプロイ後に、Pod を追加のネットワークに接続することができます。詳細は、複数ネットワークについて を参照してください。

12.6.14.9. Kuryr ポートプール

Kuryr ポートプールでは、Pod 作成のスタンバイ状態の多数のポートを維持します。

ポートをスタンバイ状態に維持すると、Pod の作成時間が必要最小限に抑えることができます。ポートプールを使用しない場合には、Kuryr は Pod が作成または削除されるたびにポートの作成または削除を明示的に要求する必要があります。

Kuryr が使用する Neutron ポートは、namespace に関連付けられるサブネットに作成されます。これらの Pod ポートは、OpenShift Container Platform クラスターノードのプライマリーポートにサブポートとして追加されます。

Kuryr は namespace をそれぞれ、別のサブネットに保存するため、namespace-worker ペアごとに別個のポートプールが維持されます。

クラスターをインストールする前に、**cluster-network-03-config.yml** マニフェストファイルに以下のパラメーターを設定して、ポートプールの動作を設定できます。

- **enablePortPoolsPrepopulation** パラメーターで、プールの事前生成が制御されるので、Kuryr は新規ホストが追加される時や新規 namespace の作成時などの作成時に、Kuryr によりプールにポートが強制的に追加されるようにします。デフォルト値は **false** です。
- **poolMinPorts** パラメーターは、プールに保持する空きポートの最小数です。デフォルト値は **1** です。
- **poolMaxPorts** パラメーターは、プールに保持する空きポートの最大数です。値が **0** の場合は、上限が無効になります。これはデフォルト設定です。
  OpenStack ポートのクォータが低い場合や、Pod ネットワークで IP アドレスの数が限定されている場合には、このオプションを設定して、不要なポートが削除されるようにします。
- **poolBatchPorts** パラメーターは、一度に作成可能な Neutron ポートの最大数を定義します。デフォルト値は **3** です。

12.6.14.10. インストール時の Kuryr ポートプールの調整

インストール時に、Pod 作成の速度や効率性を制御するために Kuryr で Red Hat OpenStack Platform (RHOSP) Neutron ポートを管理する方法を設定できます。

前提条件

- **install-config.yaml** ファイルを作成して変更しておく。

手順

1. コマンドラインからマニフェストファイルを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Cluster Network Operator 設定を記述するカスタムリソース (CR) を入力します。

```
$ oc edit networks.operator.openshift.io cluster
```

4. 要件に合わせて設定を編集します。以下のファイルをサンプルとして紹介しています。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: Kuryr
    kuryrConfig:
      enablePortPoolsPrepopulation: false 1
      poolMinPorts: 1 2
      poolBatchPorts: 3 3
      poolMaxPorts: 5 4
      openstackServiceNetwork: 172.30.0.0/15 5
```

1. **enablePortPoolsPrepopulation** の値を **true** に設定し、namespace の作成時、または新規ノードがクラスターに追加された後に Kuryr が新規 Neutron ポートを作成するようにします。この設定により、Neutron ポートのクォータが引き上げられますが、Pod の起動に必要となる時間を短縮できます。デフォルト値は **false** です。

2. Kuryr は、対象のプール内にある空きポートの数が **poolMinPorts** の値よりも少ない場合には、プールに新規ポートを作成します。デフォルト値は **1** です。

3. **poolBatchPorts** は、空きポートの数が **poolMinPorts** の値よりも少ない場合に作成される新規ポートの数を制御します。デフォルト値は **3** です。

4. プール内の空きポートの数が **poolMaxPorts** の値よりも多い場合に、Kuryr はその値と同じ数になるまでポートを削除します。この値を **0** に設定すると、この上限は無効になり、プールが縮小できないようにします。デフォルト値は **0** です。

5. **openStackServiceNetwork** パラメーターは、RHOSP Octavia の LoadBalancer に割り当てられるネットワークの CIDR 範囲を定義します。

   このパラメーターを Amphora ドライバーと併用する場合には、Octavia は、ロードバランサーごとに、このネットワークから IP アドレスを 2 つ (OpenShift 用に 1 つ、VRRP 接続用に 1 つ) 取得します。これらの IP アドレスは OpenShift Container Platform と Neutron でそれぞれ管理されるため、異なるプールから取得する必要があります。したがって、**openStackServiceNetwork serviceNetwork** の値の 2 倍になる必要があり、**serviceNetwork** の値は、**openStackServiceNetwork** で定義された範囲と完全に重複する必要があります。

   CNO は、このパラメーターの定義範囲から取得した VRRP IP アドレスが **serviceNetwork** パラメーターの定義範囲と重複しないことを検証します。

   このパラメーターが設定されていない場合には、CNO は **serviceNetwork** の拡張値を使用します。この値は、プリフィックスのサイズを 1 つずつ減らして決定します。

5. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

6. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

12.6.14.11. マシンのカスタムサブネットの設定

インストールプログラムがデフォルトで使用する IP 範囲は、OpenShift Container Platform のインストール時に作成する Neutron サブネットと一致しない可能性があります。必要な場合は、インストール設定ファイルを編集して、新規マシンの CIDR 値を更新します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。

2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。

   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

1 必要な Neutron サブネットに一致する値 (例: **192.0.2.0/24**) を挿入します。

- 値を手動で設定するには、ファイルを開き、**networking.machineCIDR** の値を必要な Neutron サブネットに一致する値に設定します。

12.6.14.12. コンピュートマシンプールを空にする

独自のインフラストラクチャーを使用するインストールを実行するには、インストール設定ファイルのコンピュートマシンの数をゼロに設定します。その後、これらのマシンを手動で作成します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   - 値を手動で設定するには、ファイルを開き、**compute.<first entry>.replicas** の値を **0** に設定します。

12.6.14.13. ネットワークタイプの変更

デフォルトで、インストールプログラムは **OpenShiftSDN** ネットワークタイプを選択します。代わりに Kuryr を使用するには、プログラムが生成したインストール設定ファイルの値を変更します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドプロンプトで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

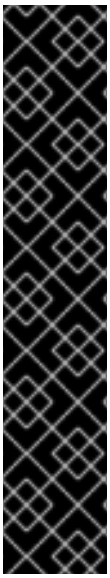
```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["networkType"] = "Kuryr";
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   - 値を手動で設定するには、ファイルを開き、**networking.networkType** を **"Kuryr"** に設定します。

12.6.15. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

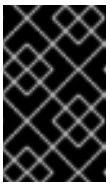
```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. メタデータファイルの **infraID** キーを環境変数としてエクスポートします。

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

ヒント

metadata.json から **infraID** キーを抽出し、作成するすべての RHOSP リソースの接頭辞として使用します。これを実行することで、同じプロジェクトで複数のデプロイメントを実行する際に名前の競合が発生しないようにします。

12.6.16. ブートストラップ Ignition ファイルの準備

OpenShift Container Platform インストールプロセスは、ブートストラップ Ignition 設定ファイルから作成されるブートストラップマシンに依存します。

ファイルを編集し、アップロードします。次に、Red Hat OpenStack Platform (RHOSP) がプライマリーファイルをダウンロードする際に使用するセカンダリーブートストラップ Ignition 設定ファイルを作成します。

前提条件

- インストーラープログラムが生成するブートストラップ Ignition ファイル **bootstrap.ign** があります。
- インストーラーのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成** を参照してください。
- HTTP(S) でアクセス可能な方法でブートストラップ Ignition ファイルを保存できます。
  - 記載された手順では RHOSP イメージサービス (Glance) を使用しますが、RHOSP ストレージサービス (Swift)、Amazon S3、内部 HTTP サーバー、またはアドホックの Nova サーバーを使用することもできます。

手順

1. 以下の Python スクリプトを実行します。スクリプトはブートストラップ Ignition ファイルを変更して、ホスト名および利用可能な場合は、実行時の CA 証明書ファイルを設定します。

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

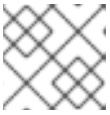
2. RHOSP CLI を使用して、ブートストラップ Ignition ファイルを使用するイメージを作成します。

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. イメージの詳細を取得します。

```
$ openstack image show <image_name>
```

file 値をメモします。これは **v2/images/<image_ID>/file** パターンをベースとしています。



注記

作成したイメージがアクティブであることを確認します。

4. イメージサービスのパブリックアドレスを取得します。

```
$ openstack catalog show image
```

5. パブリックアドレスとイメージ **file** 値を組み合わせ、結果を保存場所として保存します。この場所は、**<image_service_public_URL>/v2/images/<image_ID>/file** パターンをベースとしています。

6. 認証トークンを生成し、トークン ID を保存します。

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** というファイルに以下のコンテンツを挿入し、独自の値に一致するようにプレースホルダーを編集します。

```
{
  "ignition": {
    "config": {
      "merge": [{
        "source": "<storage_url>", 1
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>" 4
        }]
      }
    },
    "version": "3.2.0"
  }
}
```

1. **ignition.config.append.source** の値をブートストラップ Ignition ファイルのストレージ URL に置き換えます。
2. **httpHeaders** の **name** を **"X-Auth-Token"** に設定します。
3. **httpHeaders** の **value** をトークンの ID に設定します。
4. ブートストラップ Ignition ファイルサーバーが自己署名証明書を使用する場合は、base64 でエンコードされた証明書を含めます。

8. セカンダリー Ignition 設定ファイルを保存します。

ブートストラップ Ignition データはインストール時に RHOSP に渡されます。

警告

ブートストラップ Ignition ファイルには、**clouds.yaml** 認証情報などの機密情報が含まれます。これを安全な場所に保存し、インストールプロセスの完了後に削除します。

12.6.17. RHOSP でのコントロールプレーンの Ignition 設定ファイルの作成

独自のインフラストラクチャーを使用して OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールするには、コントロールプレーンの Ignition 設定ファイルが必要です。複数の設定ファイルを作成する必要があります。

注記

ブートストラップ Ignition 設定と同様に、各コントロールプレーンマシンのホスト名を明示的に定義する必要があります。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。

手順

- コマンドラインで、以下の Python スクリプトを実行します。

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
storage['files'] = files;
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

以下の 3 つのコントロールプレーン Ignition ファイルが作成されます。**<INFRA_ID>-master-0-ignition.json**、**<INFRA_ID>-master-1-ignition.json**、および **<INFRA_ID>-master-2-ignition.json**。

12.6.18. RHOSP でのネットワークリソースの作成

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを作成します。時間を節約するには、セキュリティーグループ、ネットワーク、サブネット、ルーター、およびポートを生成する指定された Ansible Playbook を実行します。

前提条件

- Python 3 がマシンにインストールされている。
- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。

手順

1. オプション: 外部ネットワークの値を **inventory.yaml** Playbook に追加します。

   inventory.yaml Ansible Playbook の外部ネットワーク値の例

   ```
   ...
       # The public network providing connectivity to the cluster. If not
       # provided, the cluster external connectivity must be provided in another
       # way.

       # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
       os_external_network: 'external'
   ...
   ```

   重要

   inventory.yaml ファイルの **os_external_network** の値を指定しなかった場合は、仮想マシンが Glance および外部接続にアクセスできるようにする必要があります。

2. オプション: 外部ネットワークおよび Floating IP (FIP) アドレスの値を **inventory.yaml** Playbook に追加します。

   inventory.yaml Ansible Playbook の FIP 値の例

   ```
   ...
       # OpenShift API floating IP address. If this value is non-empty, the
       # corresponding floating IP will be attached to the Control Plane to
       # serve the OpenShift API.
       os_api_fip: '203.0.113.23'

       # OpenShift Ingress floating IP address. If this value is non-empty, the
       # corresponding floating IP will be attached to the worker nodes to serve
       # the applications.
       os_ingress_fip: '203.0.113.19'

       # If this value is non-empty, the corresponding floating IP will be
       # attached to the bootstrap machine. This is needed for collecting logs
       # in case of install failure.
       os_bootstrap_fip: '203.0.113.20'
   ```

重要

os_api_fip および **os_ingress_fip** の値を定義しない場合、インストール後のネットワーク設定を実行する必要があります。

os_bootstrap_fip の値を定義しない場合、インストーラーは失敗したインストールからデバッグ情報をダウンロードできません。

詳細は、環境へのアクセスの有効化を参照してください。

3. コマンドラインで、**security-groups.yaml** Playbook を実行してセキュリティーグループを作成します。

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. コマンドラインで、**network.yaml** Playbook を実行して、ネットワーク、サブネット、およびルーターを作成します。

```
$ ansible-playbook -i inventory.yaml network.yaml
```

5. オプション: Nova サーバーが使用するデフォルトのリゾルバーを制御する必要がある場合は、RHOSP CLI コマンドを実行します。

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

12.6.19. RHOSP でのブートストラップマシンの作成

ブートストラップマシンを作成し、これに Red Hat OpenStack Platform (RHOSP) で実行するために必要なネットワークアクセスを付与します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **bootstrap.yaml** Ansible Playbook は共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. ブートストラップサーバーがアクティブになった後に、ログを表示し、Ignition ファイルが受信されたことを確認します。

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

12.6.20. RHOSP でのコントロールプレーンの作成

生成した Ignition 設定ファイルを使用して 3 つのコントロールプレーンマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
- **inventory.yaml**、 **common.yaml**、および **control-plane.yaml** Ansible Playbook は共通ディレクトリーにあります。
- コントロールプレーンの Ignition 設定ファイルの作成で作成された 3 つの Ignition ファイルがある。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コントロールプレーン Ignition 設定ファイルが作業ディレクトリーにない場合、それらをここにコピーします。
3. コマンドラインで、**control-plane.yaml** Playbook を実行します。

   ```
   $ ansible-playbook -i inventory.yaml control-plane.yaml
   ```

4. 以下のコマンドを実行してブートストラッププロセスをモニターします。

   ```
   $ openshift-install wait-for bootstrap-complete
   ```

   コントロールプレーンマシンが実行され、クラスターに参加していることを確認できるメッセージが表示されます。

   ```
   INFO API v1.14.6+f9b5405 up
   INFO Waiting up to 30m0s for bootstrapping to complete...
   ...
   INFO It is now safe to remove the bootstrap resources
   ```

12.6.21. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.6.22. RHOSP からのブートストラップリソースの削除

不要になったブートストラップリソースを削除します。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、**common.yaml**、および **down-bootstrap.yaml** Ansible Playbook が共通ディレクトリーにある。
- コントロールプレーンマシンが実行中である。
  - マシンのステータスが不明な場合は、クラスターステータスの確認を参照してください。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**down-bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

ブートストラップポート、サーバー、および Floating IP アドレスが削除されます。

警告

ブートストラップ Ignition ファイル URL をまだ無効にしていない場合は、無効にしてください。

12.6.23. RHOSP でのコンピュートマシンの作成

コントロールプレーンの起動後、コンピュートマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **compute-nodes.yaml** Ansible Playbook が共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。
- コントロールプレーンが有効である。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

次のステップ

- マシンの証明書署名要求を承認します。

12.6.24. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.21.0
master-1  Ready    master  63m  v1.21.0
master-2  Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                 CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

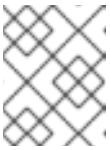
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

12.6.25. インストールの正常な実行の確認

OpenShift Container Platform のインストールが完了していることを確認します。

前提条件

- インストールプログラム (**openshift-install**) があります。

手順

- コマンドラインで、以下を入力します。

```
$ openshift-install --log-level debug wait-for install-complete
```

プログラムはコンソール URL と管理者のログイン情報を出力します。

12.6.26. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.6.27. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.7. 独自の SR-IOV インフラストラクチャーを使用した OPENSTACK へのクラスターのインストール

OpenShift Container Platform 4.8 では、ユーザーによってプロビジョニングされたインフラストラクチャーで実行され、Single-root Input/Output Virtualization (SR-IOV) ネットワークを使用してコンピュートマシンを実行するクラスターを Red Hat OpenStack Platform (RHOSP) にインストールできます。

独自のインフラストラクチャーを使用することで、クラスターを既存のインフラストラクチャーおよび変更と統合できます。このプロセスでは、インストーラーでプロビジョニングされるインストールの場合よりも多くの手作業が必要になります。Nova サーバー、Neutron ポート、セキュリティーグループなどのすべての RHOSP リソースを作成する必要があるためです。ただし、Red Hat では、デプロイメントプロセスを支援する Ansible Playbook を提供しています。

12.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- OpenShift クラスターでサポートされるプラットフォーム のセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認している。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。
- OpenShift Container Platform のインストール先に RHOSP アカウントがある。
- インストールプログラムを実行するマシンには、以下が含まれます。
  - インストールプロセス時に作成したファイルを保持できる単一ディレクトリー

- Python 3

12.7.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.7.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

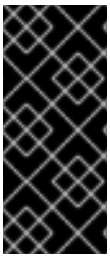
OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表12.29 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB

リソース	値
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

12.7.3.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.7.3.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス

- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

また、Single-root Input/Output Virtualization (SR-IOV) を使用するクラスターの場合、RHOSP コンピュートノードには Huge Page をサポートするフレーバーが必要です。

重要

SR-IOV デプロイメントでは、多くの場合、専用の CPU や分離された CPU などのパフォーマンスの最適化が駆使されます。パフォーマンスを最大化するには、基礎となる RHOSP デプロイメントをこれらの最適化機能を使用するように設定してから、OpenShift Container Platform コンピュートマシンを最適化されたインフラストラクチャーで実行するように設定します。

関連情報

- パフォーマンスの良い RHOSP コンピュートノードの設定についての詳細は、パフォーマンスを向上させるためのコンピュートノードの設定 について参照してください。

12.7.3.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.7.4. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでのインストールプロセスを単純化する Ansible Playbook には、複数の Python モジュールが必要です。インストーラーを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされている。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

12.7.5. インストール Playbook のダウンロード

OpenShift Container Platform を独自の Red Hat OpenStack Platform (RHOSP) インフラストラクチャーにインストールするために使用できる Ansible Playbook をダウンロードします。

前提条件

- curl コマンドラインツールがマシンで利用できる。

手順

- Playbook を作業ディレクトリーにダウンロードするには、コマンドラインから以下のスクリプトを実行します。

```
$ xargs -n 1 curl -O <<< '
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/bootstrap.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/common.yaml
      https://raw.githubusercontent.com/openshift/installer/release-
4.8/upi/openstack/compute-nodes.yaml
```

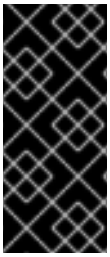
```
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/inventory.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-bootstrap.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-compute-nodes.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-control-plane.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-load-balancers.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-network.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-security-groups.yaml
    https://raw.githubusercontent.com/openshift/installer/release-4.8/upi/openstack/down-containers.yaml'
```

Playbook はマシンにダウンロードされます。

重要

インストールプロセス時に、Playbook を変更してデプロイメントを設定できます。

クラスターの有効期間中に、すべての Playbook を保持します。OpenShift Container Platform クラスターを RHOSP から削除するには Playbook が必要です。



重要

bootstrap.yaml、**compute-nodes.yaml**、**control-plane.yaml**、**network.yaml**、および **security-groups.yaml** ファイルに加えた編集内容は、**down-** の接頭辞が付けられた対応する Playbook に一致している必要があります。たとえば、**bootstrap.yaml** ファイルへの編集は、**down-bootstrap.yaml** ファイルにも反映される必要があります。両方のファイルを編集しない場合、サポートされるクラスターの削除プロセスは失敗します。

12.7.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

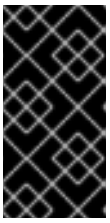
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.7.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

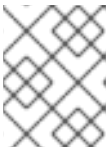
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

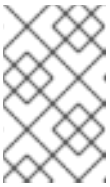
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.7.8. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成

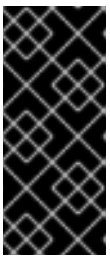
OpenShift Container Platform インストールプログラムでは、Red Hat Enterprise Linux CoreOS (RHCOS) イメージが Red Hat OpenStack Platform (RHOSP) クラスターに存在する必要があります。最新の RHCOS イメージを取得した後、RHOSP CLI を使用してこれをアップロードします。

前提条件

- RHOSP CLI がインストールされています。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、Red Hat Enterprise Linux (RHEL) 8 用の OpenShift Container Platform 4.8 の最新リリースを選択します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**をダウンロードします。
4. イメージを展開します。

注記

クラスターが使用する前に RHOSP イメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. ダウンロードしたイメージから、RHOSP CLI を使用して **rhcos** という名前のイメージをクラスターに作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-${RHCOS_VERSION}-openstack.qcow2 rhcos
```

重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

RHOSP にイメージをアップロードした後は、インストールプログラムでイメージを利用できます。

12.7.9. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

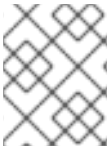
1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+---------------+-------------+
| ID                                   | Name          | Router Type |
+--------------------------------------+---------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+---------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。



注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

12.7.10. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.7.10.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API、クラスターアプリケーション、およびブートストラッププロセスへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. Red Hat OpenStack Platform (RHOSP) CLI を使用して、ブートストラップ FIP を作成します。

```
$ openstack floating ip create --description "bootstrap machine" <external_network>
```

4. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

5. FIP を以下の変数の値として **inventory.yaml** ファイルに追加します。
   - **os_api_fip**
   - **os_bootstrap_fip**
   - **os_ingress_fip**

これらの値を使用する場合には、**inventory.yaml** ファイルの **os_external_network** 変数の値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.7.10.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

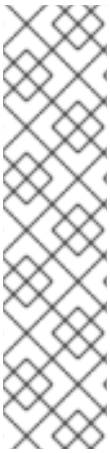
inventory.yaml ファイルで、以下の変数を定義しないでください。

- **os_api_fip**
- **os_bootstrap_fip**

- **os_ingress_fip**

外部ネットワークを提供できない場合は、**os_external_network** を空白のままにすることもできます。**os_external_network** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。インストールプロセスで、ネットワークリソースを作成する際に、独自の外部接続を設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムから **wait-for** コマンドでインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。



注記

API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

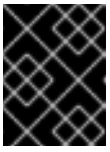
DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.7.11. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
     ```

```
    user_domain_name: Default
    project_domain_name: Default
  dev-env:
   region_name: RegionOne
   auth:
    username: 'devuser'
    password: XXX
    project_name: 'devonly'
    auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.7.12. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

      iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

      iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

      v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

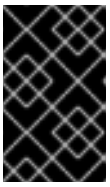
      vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

      vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

      viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

これで、指定したディレクトリーに **install-config.yaml** ファイルが作成されます。

12.7.13. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.7.13.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.30 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.7.13.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.31 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

12.7.13.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.32 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

12.7.13.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表12.33 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。

パラメーター	説明	値
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

12.7.13.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表12.34 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
compute.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
compute.platform.openstack.rootVolume.zones	コンピュートマシンの root ボリュームをインストールするアベイラビリティーゾーン。このパラメーターに値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

パラメーター	説明	値
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.rootVolume.zones	コントロールプレーンマシンの root ボリュームをインストールするアベイラビリティーゾーン。この値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform.openstack.clusterOSImageProperties	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアの一覧。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**

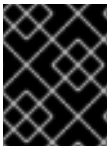
platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.ingressFloatingIP	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform.openstack.apiFloatingIP	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

12.7.13.6. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。



重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
```

```
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

12.7.13.7. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。デフォルトで、ノードおよびポートはこの上に作成されます。**platform.openstack.machinesSubnet** プロパティーの値をサブネットの UUID に設定すると、異なる RHOSP サブネットにノードおよびポートを作成することができます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、設定が以下の要件を満たしていることを確認してください。

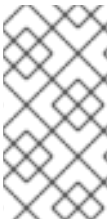
- **platform.openstack.machinesSubnet** で使用されるサブネットで DHCP が有効にされている。
- **platform.openstack.machinesSubnet** の CIDR は **networking.machineNetwork** の CIDR に一致する。
- インストールプログラムのユーザーには、固定 IP アドレスを持つポートなど、このネットワークでポートを作成するパーミッションがある。

カスタムサブネットを使用するクラスターには、以下の制限があります。

- Floating IP アドレスを使用するクラスターをインストールする予定の場合には、**platform.openstack.machinesSubnet** サブネットを **externalNetwork** ネットワークに接続されているルーターに接続する必要があります。

- **platform.openstack.machinesSubnet** の値が **install-config.yaml** ファイルに設定されている場合、インストールプログラムは RHOSP マシンのプライベートネットワークまたはサブネットを作成しません。

- **platform.openstack.externalDNS** プロパティーは、カスタムサブネットと同時に使用することはできません。カスタムサブネットを使用するクラスターに DNS を追加するには、RHOSP ネットワークで DNS を設定します。



注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

12.7.13.8. マシンのカスタムサブネットの設定

インストールプログラムがデフォルトで使用する IP 範囲は、OpenShift Container Platform のインストール時に作成する Neutron サブネットと一致しない可能性があります。必要な場合は、インストール設定ファイルを編集して、新規マシンの CIDR 値を更新します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

1 必要な Neutron サブネットに一致する値 (例: **192.0.2.0/24**) を挿入します。

   - 値を手動で設定するには、ファイルを開き、**networking.machineCIDR** の値を必要な Neutron サブネットに一致する値に設定します。

12.7.13.9. コンピュートマシンプールを空にする

独自のインフラストラクチャーを使用するインストールを実行するには、インストール設定ファイルのコンピュートマシンの数をゼロに設定します。その後、これらのマシンを手動で作成します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

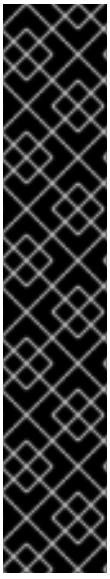
```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   - 値を手動で設定するには、ファイルを開き、**compute.<first entry>.replicas** の値を **0** に設定します。

12.7.14. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

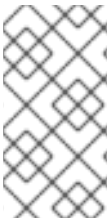
一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

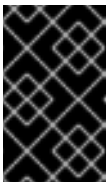
```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. メタデータファイルの **infraID** キーを環境変数としてエクスポートします。

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

ヒント

metadata.json から **infraID** キーを抽出し、作成するすべての RHOSP リソースの接頭辞として使用します。これを実行することで、同じプロジェクトで複数のデプロイメントを実行する際に名前の競合が発生しないようにします。

12.7.15. ブートストラップ Ignition ファイルの準備

OpenShift Container Platform インストールプロセスは、ブートストラップ Ignition 設定ファイルから作成されるブートストラップマシンに依存します。

ファイルを編集し、アップロードします。次に、Red Hat OpenStack Platform (RHOSP) がプライマリーファイルをダウンロードする際に使用するセカンダリーブートストラップ Ignition 設定ファイルを作成します。

前提条件

- インストーラープログラムが生成するブートストラップ Ignition ファイル **bootstrap.ign** があります。
- インストーラーのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成** を参照してください。
- HTTP(S) でアクセス可能な方法でブートストラップ Ignition ファイルを保存できます。
  - 記載された手順では RHOSP イメージサービス (Glance) を使用しますが、RHOSP ストレージサービス (Swift)、Amazon S3、内部 HTTP サーバー、またはアドホックの Nova

サーバーを使用することもできます。

手順

1. 以下の Python スクリプトを実行します。スクリプトはブートストラップ Ignition ファイルを変更して、ホスト名および利用可能な場合は、実行時の CA 証明書ファイルを設定します。

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64
    }
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64
        }
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

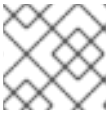
2. RHOSP CLI を使用して、ブートストラップ Ignition ファイルを使用するイメージを作成します。

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. イメージの詳細を取得します。

```
$ openstack image show <image_name>
```

file 値をメモします。これは **v2/images/<image_ID>/file** パターンをベースとしています。



注記

作成したイメージがアクティブであることを確認します。

4. イメージサービスのパブリックアドレスを取得します。

```
$ openstack catalog show image
```

5. パブリックアドレスとイメージ **file** 値を組み合わせ、結果を保存場所として保存します。この場所は、**<image_service_public_URL>/v2/images/<image_ID>/file** パターンをベースとしています。

6. 認証トークンを生成し、トークン ID を保存します。

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** というファイルに以下のコンテンツを挿入し、独自の値に一致するようにプレースホルダーを編集します。

```
{
  "ignition": {
    "config": {
      "merge": [{
        "source": "<storage_url>", 1
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>" 4
        }]
      }
    },
    "version": "3.2.0"
  }
}
```

1. **ignition.config.append.source** の値をブートストラップ Ignition ファイルのストレージ URL に置き換えます。
2. **httpHeaders** の **name** を **"X-Auth-Token"** に設定します。
3. **httpHeaders** の **value** をトークンの ID に設定します。
4. ブートストラップ Ignition ファイルサーバーが自己署名証明書を使用する場合は、base64 でエンコードされた証明書を含めます。

8. セカンダリー Ignition 設定ファイルを保存します。

ブートストラップ Ignition データはインストール時に RHOSP に渡されます。

警告

ブートストラップ Ignition ファイルには、**clouds.yaml** 認証情報などの機密情報が含まれます。これを安全な場所に保存し、インストールプロセスの完了後に削除します。

12.7.16. RHOSP でのコントロールプレーンの Ignition 設定ファイルの作成

独自のインフラストラクチャーを使用して OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールするには、コントロールプレーンの Ignition 設定ファイルが必要です。複数の設定ファイルを作成する必要があります。

注記

ブートストラップ Ignition 設定と同様に、各コントロールプレーンマシンのホスト名を明示的に定義する必要があります。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、Kubernetes マニフェストおよび Ignition 設定ファイルの作成を参照してください。

手順

- コマンドラインで、以下の Python スクリプトを実行します。

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
storage = ignition.get('storage', {});
files = storage.get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
storage['files'] = files;
ignition['storage'] = storage
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

以下の 3 つのコントロールプレーン Ignition ファイルが作成されます。**<INFRA_ID>-master-0-ignition.json**、**<INFRA_ID>-master-1-ignition.json**、および **<INFRA_ID>-master-2-ignition.json**。

12.7.17. RHOSP でのネットワークリソースの作成

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを作成します。時間を節約するには、セキュリティーグループ、ネットワーク、サブネット、ルーター、およびポートを生成する指定された Ansible Playbook を実行します。

前提条件

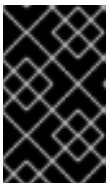
- Python 3 がマシンにインストールされている。
- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。

手順

1. オプション: 外部ネットワークの値を **inventory.yaml** Playbook に追加します。

   inventory.yaml Ansible Playbook の外部ネットワーク値の例

   ```
   ...
       # The public network providing connectivity to the cluster. If not
       # provided, the cluster external connectivity must be provided in another
       # way.

       # Required for os_api_fip, os_ingress_fip, os_bootstrap_fip.
       os_external_network: 'external'
   ...
   ```

   重要

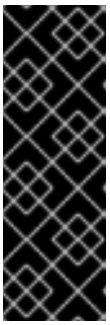
   inventory.yaml ファイルの **os_external_network** の値を指定しなかった場合は、仮想マシンが Glance および外部接続にアクセスできるようにする必要があります。

2. オプション: 外部ネットワークおよび Floating IP (FIP) アドレスの値を **inventory.yaml** Playbook に追加します。

   inventory.yaml Ansible Playbook の FIP 値の例

   ```
   ...
       # OpenShift API floating IP address. If this value is non-empty, the
       # corresponding floating IP will be attached to the Control Plane to
       # serve the OpenShift API.
       os_api_fip: '203.0.113.23'

       # OpenShift Ingress floating IP address. If this value is non-empty, the
       # corresponding floating IP will be attached to the worker nodes to serve
       # the applications.
   ```

```
os_ingress_fip: '203.0.113.19'

# If this value is non-empty, the corresponding floating IP will be
# attached to the bootstrap machine. This is needed for collecting logs
# in case of install failure.
os_bootstrap_fip: '203.0.113.20'
```



重要

os_api_fip および **os_ingress_fip** の値を定義しない場合、インストール後のネットワーク設定を実行する必要があります。

os_bootstrap_fip の値を定義しない場合、インストーラーは失敗したインストールからデバッグ情報をダウンロードできません。

詳細は、環境へのアクセスの有効化を参照してください。

3. コマンドラインで、**security-groups.yaml** Playbook を実行してセキュリティーグループを作成します。

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

4. コマンドラインで、**network.yaml** Playbook を実行して、ネットワーク、サブネット、およびルーターを作成します。

```
$ ansible-playbook -i inventory.yaml network.yaml
```

5. オプション: Nova サーバーが使用するデフォルトのリゾルバーを制御する必要がある場合は、RHOSP CLI コマンドを実行します。

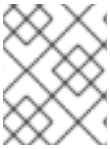
```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

オプションで、作成した **inventory.yaml** ファイルを使用してインストールをカスタマイズできます。たとえば、ベアメタルマシンを使用するクラスターをデプロイすることができます。

12.7.17.1. ベアメタルマシンを使用したクラスターのデプロイ

クラスターがベアメタルマシンを使用する必要がある場合は、**inventory.yaml** ファイルを変更します。クラスターには、ベアメタル上でコントロールプレーンとコンピュートマシンの両方を実行させることも、コンピュートマシンのみを実行させることもできます。

ベアメタルコンピュートマシンは、Kuryr を使用するクラスターではサポートされません。



注記

install-config.yaml ファイルで、ベアメタルワーカーに使用する RHOSP ネットワークが Floating IP アドレスをサポートするかどうかが反映されていることを確認します。

前提条件

- RHOSP の ベアメタルサービス (Ironic) は有効にされており、RHOSP Compute API でアクセスできる。

- ベアメタルは RHOSP フレーバー として利用可能である。
- RHOSP ネットワークは、仮想マシンとベアメタルサーバー接続の両方をサポートする。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。
- マシンを既存のネットワークにデプロイする必要がある場合、RHOSP サブネットがプロビジョニングされる。
- マシンをインストーラーでプロビジョニングされるネットワークに場合、RHOSP ベアメタルサービス (Ironic) はテナントネットワークで実行される Preboot eXecution Environment (PXE) ブートマシンをリッスンし、これと対話できます。
- **inventory.yaml** ファイルを OpenShift Container Platform インストールプロセスの一部として作成している。

手順

1. **inventory.yaml** ファイルで、マシンのフレーバーを編集します。
   a. ベアメタルコントロールプレーンマシンを使用する場合は、**os_flavor_master** の値をベアメタルフレーバーに変更します。
   b. **os_flavor_worker** の値をベアメタルフレーバーに変更します。

   ベアメタルの inventory.yaml のサンプルファイル

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'my-bare-metal-flavor' 1
      os_flavor_worker: 'my-bare-metal-flavor' 2
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
...
```

1. ベアメタルのコントロールプレーンマシンを使用する必要がある場合は、この値をベアメタルのフレーバーに変更します。
2. この値を、コンピュートマシンに使用するベアメタルのフレーバーに変更します。

更新された **inventory.yaml** ファイルを使用してインストールプロセスを完了します。デプロイメント時に作成されるマシンは、ファイルに追加したフレーバーを使用します。

注記

インストーラーは、ベアメタルマシンの起動中にタイムアウトする可能性があります。

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
./openshift-install wait-for install-complete --log-level debug
```

12.7.18. RHOSP でのブートストラップマシンの作成

ブートストラップマシンを作成し、これに Red Hat OpenStack Platform (RHOSP) で実行するために必要なネットワークアクセスを付与します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、 **common.yaml**、および **bootstrap.yaml** Ansible Playbook は共通ディレクトリーにある。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

3. ブートストラップサーバーがアクティブになった後に、ログを表示し、Ignition ファイルが受信されたことを確認します。

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

12.7.19. RHOSP でのコントロールプレーンの作成

生成した Ignition 設定ファイルを使用して 3 つのコントロールプレーンマシンを作成します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。

- **inventory.yaml**、 **common.yaml**、および **control-plane.yaml** Ansible Playbook は共通ディレクトリーにあります。
- コントロールプレーンの Ignition 設定ファイルの作成で作成された 3 つの Ignition ファイルがある。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コントロールプレーン Ignition 設定ファイルが作業ディレクトリーにない場合、それらをここにコピーします。
3. コマンドラインで、**control-plane.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

4. 以下のコマンドを実行してブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete
```

コントロールプレーンマシンが実行され、クラスターに参加していることを確認できるメッセージが表示されます。

```
INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

12.7.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.7.21. RHOSP からのブートストラップリソースの削除

不要になったブートストラップリソースを削除します。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- **inventory.yaml**、**common.yaml**、および **down-bootstrap.yaml** Ansible Playbook が共通ディレクトリーにある。
- コントロールプレーンマシンが実行中である。
  - マシンのステータスが不明な場合は、クラスターステータスの確認を参照してください。

手順

1. コマンドラインで、作業ディレクトリーを Playbook の場所に切り替えます。
2. コマンドラインで、**down-bootstrap.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

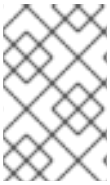
ブートストラップポート、サーバー、および Floating IP アドレスが削除されます。

警告

ブートストラップ Ignition ファイル URL をまだ無効にしていない場合は、無効にしてください。

12.7.22. コンピュートマシン用の SR-IOV ネットワークの作成

Red Hat OpenStack Platform (RHOSP) デプロイメントで Single Root I/O Virtualization (SR-IOV) をサポートする場合、コンピュートマシンを実行する SR-IOV ネットワークをプロビジョニングすることができます。

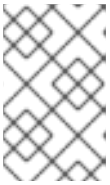


注記

以下の手順では、コンピュートマシンへの接続が可能な外部のフラットネットワークおよび外部の VLAN ベースのネットワークを作成します。RHOSP のデプロイメントによっては、ネットワークの他のタイプが必要になる場合があります。

前提条件

- クラスターは SR-IOV をサポートしている。



注記

クラスターがサポートするかどうかが不明な場合は、OpenShift Container Platform SR-IOV ハードウェアネットワークについてのドキュメントを参照してください。

- RHOSP デプロイメントの一部として、無線とアップリンクのプロバイダーネットワークを作成している。これらのネットワークを表すために **radio** および **uplink** の名前がすべてのコマンド例で使用されています。

手順

1. コマンドラインで、無線の RHOSP ネットワークを作成します。

```
$ openstack network create radio --provider-physical-network radio --provider-network-type flat --external
```

2. アップリンクの RHOSP ネットワークを作成します。

```
$ openstack network create uplink --provider-physical-network uplink --provider-network-type vlan --external
```

3. 無線ネットワーク用のサブネットを作成します。

```
$ openstack subnet create --network radio --subnet-range <radio_network_subnet_range> radio
```

4. アップリンクネットワーク用のサブネットを作成します。

```
$ openstack subnet create --network uplink --subnet-range <uplink_network_subnet_range> uplink
```

12.7.23. SR-IOV ネットワークで実行されるコンピュートマシンの作成

コントロールプレーンの起動後に、コンピュートマシン用の SR-IOV ネットワークの作成で作成した SR-IOV ネットワークで実行されるコンピュートマシンを作成します。

前提条件

- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- インストール Playbook のダウンロードで Playbook をダウンロードしている。
- インストールプログラムが作成した **metadata.yaml** ファイルが Ansible Playbook と同じディレクトリーにある。
- コントロールプレーンが有効である。

- コンピュートマシン用の SR-IOV ネットワークの作成で説明されているように **radio** および **uplink** SR-IOV ネットワークを作成している。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。

2. **additionalNetworks** パラメーターを使用して、**radio** および **uplink** ネットワークを **inventory.yaml** ファイルの末尾に追加します。

```
....
# If this value is non-empty, the corresponding floating IP will be
# attached to the bootstrap machine. This is needed for collecting logs
# in case of install failure.
   os_bootstrap_fip: '203.0.113.20'

   additionalNetworks:
   - id: radio
     count: 4 (1)
     type: direct
     port_security_enabled: no
   - id: uplink
     count: 4 (2)
     type: direct
     port_security_enabled: no
```

(1) (2) **count** パラメーターは、各ワーカーノードに割り当てる SR-IOV 仮想機能 (VF) の数を定義します。この場合、各ネットワークには 4 つの VF があります。

3. **compute-nodes.yaml** ファイルの内容を以下のテキストに置き換えます。

例12.1compute-nodes.yaml

```
- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  vars:
    worker_list: []
    port_name_list: []
    nic_list: []

  tasks:
  # Create the SDN/primary port for each worker node
  - name: 'Create the Compute ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_worker }}"
      allowed_address_pairs:
      - ip_address: "{{ os_ingressVIP }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
```

```
    register: ports

  # Tag each SDN/primary port with cluster name
  - name: 'Set Compute ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

  - name: 'List the Compute Trunks'
    command:
      cmd: "openstack network trunk list"
    when: os_networking_type == "Kuryr"
    register: compute_trunks

  - name: 'Create the Compute trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
    - os_networking_type == "Kuryr"
    - "os_compute_trunk_name|string not in compute_trunks.stdout"

  - name: ‘Call additional-port processing’
    include_tasks: additional-ports.yaml

  # Create additional ports in OpenStack
  - name: ‘Create additionalNetworks ports’
    os_port:
      name:  "{{ item.0 }}-{{ item.1.name }}"
      vnic_type: "{{ item.1.type }}"
      network: "{{ item.1.uuid }}"
      port_security_enabled: "{{ item.1.port_security_enabled|default(omit) }}"
      no_security_groups: "{{ 'true' if item.1.security_groups is not defined else omit }}"
      security_groups: "{{ item.1.security_groups | default(omit) }}"
    with_nested:
      - "{{ worker_list }}"
      - "{{ port_name_list }}"

  # Tag the ports with the cluster info
  - name: 'Set additionalNetworks ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.0 }}-{{ item.1.name }}"
    with_nested:
      - "{{ worker_list }}"
      - "{{ port_name_list }}"

  # Build the nic list to use for server create
  - name: Build nic list
    set_fact:
      nic_list: "{{ nic_list | default([]) + [ item.name ] }}"
    with_items: "{{ port_name_list }}"

  # Create the servers
  - name: 'Create the Compute servers'
    vars:
```

```
      worker_nics: "{{ [ item.1 ] | product(nic_list) | map('join','-') | map('regex_replace', '(.*)',
'port-name=\\1') | list }}"
    os_server:
      name: "{{ item.1 }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_worker }}"
      auto_ip: no
      userdata: "{{ lookup('file', 'worker.ign') | string }}"
      security_groups: []
      nics:  "{{ [ 'port-name=' + os_port_worker + '-' + item.0|string ] + worker_nics }}"
      config_drive: yes
    with_indexed_items: "{{ worker_list }}"
```

4. **additional-ports.yaml** というローカルファイルに、以下の内容を挿入します。

例12.2 additional-ports.yaml

```
# Build a list of worker nodes with indexes
- name: 'Build worker list'
  set_fact:
    worker_list: "{{ worker_list | default([]) + [ item.1 + '-' + item.0 | string ] }}"
  with_indexed_items: "{{ [ os_compute_server_name ] * os_compute_nodes_number }}"

# Ensure that each network specified in additionalNetworks exists
- name: 'Verify additionalNetworks'
  os_networks_info:
    name: "{{ item.id }}"
  with_items: "{{ additionalNetworks }}"
  register: network_info

# Expand additionalNetworks by the count parameter in each network definition
- name: 'Build port and port index list for additionalNetworks'
  set_fact:
    port_list: "{{ port_list | default([]) + [ {
                  'net_name' : item.1.id,
                  'uuid' : network_info.results[item.0].openstack_networks[0].id,
                  'type' : item.1.type|default('normal'),
                  'security_groups' : item.1.security_groups|default(omit),
                  'port_security_enabled' : item.1.port_security_enabled|default(omit)
                  } ] * item.1.count|default(1) }}"
    index_list: "{{ index_list | default([]) + range(item.1.count|default(1)) | list }}"
  with_indexed_items: "{{ additionalNetworks }}"

# Calculate and save the name of the port
# The format of the name is cluster_name-worker-workerID-networkUUID(partial)-count
# i.e. fdp-nz995-worker-1-99bcd111-1
- name: 'Calculate port name'
  set_fact:
    port_name_list: "{{ port_name_list | default([]) + [ item.1 | combine( {'name' : item.1.uuid
| regex_search('([^-]+)') + '-' + index_list[item.0]|string } ) ] }}"
  with_indexed_items: "{{ port_list }}"
  when: port_list is defined
```

5. コマンドラインで、**compute-nodes.yaml** Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

12.7.24. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

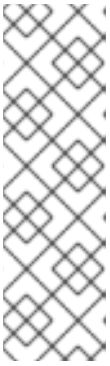
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

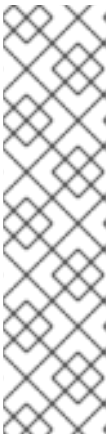
```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

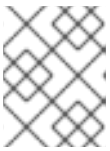
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

12.7.25. インストールの正常な実行の確認

OpenShift Container Platform のインストールが完了していることを確認します。

前提条件

- インストールプログラム (**openshift-install**) があります。

手順

- コマンドラインで、以下を入力します。

```
$ openshift-install --log-level debug wait-for install-complete
```

プログラムはコンソール URL と管理者のログイン情報を出力します。

クラスターが機能しています。ただし、これを SR-IOV ネットワーク用に設定する前に、追加のタスクを実行する必要があります。

12.7.26. SR-IOV 向けに RHOSP で実行されるクラスターの準備

Red Hat OpenStack Platform (RHOSP) で実行されるクラスターで single root I/O virtualization (SR-IOV) を使用する前に、RHOSP メタデータをドライブとしてマウント可能にし、仮想機能 I/O (VFIO) ドライバーの非 IOMMU Operator を有効にします。

12.7.26.1. RHOSP メタデータサービスをマウント可能なドライブとして有効化

マシン設定をマシンプールに適用することで、Red Hat OpenStack Platform (RHOSP) メタデータサービスをマウント可能なドライブとして利用可能にすることができます。

以下のマシン設定により、SR-IOV ネットワーク Operator 内から RHOSP ネットワーク UUID を表示できます。この設定により、SR-IOV リソースのクラスター SR-IOV リソースへの関連付けが単純化されます。

手順

1. 以下のテンプレートからマシン設定ファイルを作成します。

マウント可能なメタデータサービスマシン設定ファイル

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 20-mount-config (1)
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: create-mountpoint-var-config.service
          enabled: true
          contents: |
            [Unit]
            Description=Create mountpoint /var/config
            Before=kubelet.service

            [Service]
            ExecStart=/bin/mkdir -p /var/config

            [Install]
            WantedBy=var-config.mount
```

```
      - name: var-config.mount
        enabled: true
        contents: |
          [Unit]
          Before=local-fs.target
          [Mount]
          Where=/var/config
          What=/dev/disk/by-label/config-2
          [Install]
          WantedBy=local-fs.target
```

1 選択する名前に置き換えることができます。

2. コマンドラインからマシン設定を適用します。

```
$ oc apply -f <machine_config_file_name>.yaml
```

12.7.26.2. RHOSP VFIO ドライバーの非 IOMMU 機能の有効化

マシン設定をマシンプールに適用することで、Red Hat OpenStack Platform (RHOSP) 仮想機能 I/O (VFIO) ドライバーの非 IOMMU 機能を有効にすることができます。RHOSP vfio-pci ドライバーではこの機能が必要です。

手順

1. 以下のテンプレートからマシン設定ファイルを作成します。

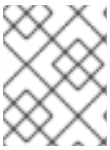
非 IOMMU VFIO マシン設定ファイル

```
kind: MachineConfig
apiVersion: machineconfiguration.openshift.io/v1
metadata:
  name: 99-vfio-noiommu 1
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - path: /etc/modprobe.d/vfio-noiommu.conf
        mode: 0644
        contents:
          source:
data:;base64,b3B0aW9ucyB2ZmlvIGVuYWJsZV91bnNhZmVfbm9pb21tdV9tb2RlPTEK
```

1 選択する名前に置き換えることができます。

2. コマンドラインからマシン設定を適用します。

```
$ oc apply -f <machine_config_file_name>.yaml
```



注記

マシン設定をマシンプールに適用した後に、 マシン設定プールのステータスを確認 し、マシンが利用可能になるタイミングを確認できます。

クラスターがインストールされ、SR-IOV 設定用に準備されます。次のステップにあるように SR-IOV 設定タスクを実行する必要があります。

12.7.27. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.7.28. 関連情報

- リアルタイムの実行および低レイテンシーのデプロイメントについての詳細は、低レイテンシーノード向けの Performance Addon Operator について参照してください。

12.7.29. 次のステップ

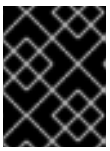
- クラスターの SR-IOV 設定を完了するには、以下を実行します。
  - Performance Addon Operator をインストール します。
  - Huge Page のサポートのある Performance Addon Operator を設定 します。
  - SR-IOV Operator をインストール します。
  - SR-IOV ネットワークデバイスを設定 します。
- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.8. ネットワークが制限された環境での OPENSTACK へのクラスターのインストール

OpenShift Container Platform 4.8 では、インストールリリースコンテンツの内部ミラーを作成して、クラスターをネットワークが制限された環境で Red Hat OpenStack Platform (RHOSP) にインストールできます。

12.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- OpenShift クラスターでサポートされるプラットフォーム のセクションを参照し、OpenShift Container Platform 4.8 がお使いの RHOSP バージョンと互換性があることを確認している。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- RHOSP でメタデータサービスが有効化されている。

12.8.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

12.8.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

12.8.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

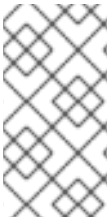
表12.35 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

12.8.3.1. コントロールプレーンマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.8.3.2. コンピュートマシン

デフォルトでは、OpenShift Container Platform インストールプロセスは 3 つのコンピューティングマシンを作成します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 8 GB のメモリー、2 つの vCPU および 100 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

12.8.3.3. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 100 GB のストレージ領域があるフレーバー

12.8.4. OpenShift Container Platform のインターネットアクセス

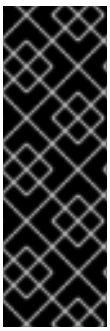
OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を

無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

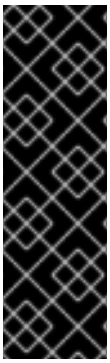


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.8.5. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。



重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

12.8.6. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: 'devuser'
           password: XXX
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。
   a. 認証局ファイルをマシンにコピーします。
   b. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

     ```
     clouds:
       shiftstack:
         ...
         cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
     ```

ヒント

カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

12.8.7. ネットワークが制限されたインストール用の RHCOS イメージの作成

Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された Red Hat OpenStack Platform (RHOSP) 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリースト上に置かれます。

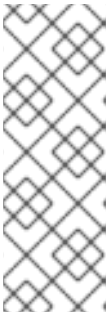
手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.8 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**イメージをダウンロードします。
4. イメージを展開します。



注記

クラスターが使用する前にイメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. 圧縮解除したイメージを、Glance などの bastion サーバーからアクセス可能な場所にアップロードします。以下に例を示します。

```
$ openstack image create --file rhcos-44.81.202003110027-0-openstack.x86_64.qcow2 --
disk-format qcow2 rhcos-${RHCOS_VERSION}
```

重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

12.8.8. インストール設定ファイルの作成

Red Hat OpenStack Platform (RHOSP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得する。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、これをアクセス可能な場所にアップロードする。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

      iii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

      iv. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

      v. コントロールプレーンノードに使用する少なくとも 16 GB の RAM とコンピュートノードに使用する 8 GB の RAM を持つ RHOSP フレーバーを指定します。

      vi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

      vii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

      viii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルで、**platform.openstack.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

   ```
   platform:
   ```

```
 openstack:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
openstack.x86_64.qcow2.gz?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

   a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

   ```
   pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

   b. **additionalTrustBundle** パラメーターおよび値を追加します。

   ```
   additionalTrustBundle: |
    -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    -----END CERTIFICATE-----
   ```

   この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

   c. 以下のようなイメージコンテンツリソースを追加します。

   ```
   imageContentSources:
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: quay.example.com/openshift-release-dev/ocp-release
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: registry.example.com/ocp/release
   ```

   これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

12.8.8.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照

するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

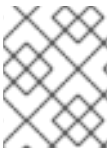
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

12.8.8.2. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.8.8.2.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.36 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.8.8.2.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.37 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

12.8.8.2.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.38 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列

パラメーター	説明	値
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

12.8.8.2.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表12.39 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。

パラメーター	説明	値
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。 このプロパティーは非推奨にされています。すべてのマシンプールのデフォルトとしてフレーバーを使用するには、これを **platform.openstack.defaultMachinePlatform** プロパティーで **type** キーの値として追加します。それぞれのマシンプールのフレーバー値を個別に設定することもできます。	文字列 (例: **m1.xlarge**)。

12.8.8.2.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表12.40 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
compute.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
compute.platform.openstack.rootVolume.zones	コンピュートマシンの root ボリュームをインストールするアベイラビリティーゾーン。このパラメーターに値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
controlPlane.platform.openstack.zones	マシンをインストールする RHOSP Compute (Nova) アベイラビリティーゾーン (AZs)。このパラメーターが設定されていない場合、インストーラーは RHOSP 管理者が設定した Nova のデフォルト設定に依存します。 Kuryr を使用するクラスターでは、RHOSP Octavia はアベイラビリティーゾーンをサポートしません。ロードバランサーおよび Amphora プロバイダードライバーを使用している場合、Amphora 仮想マシンに依存する OpenShift Container Platform サービスは、このプロパティーの値に基づいて作成されません。	文字列の一覧(例: **["zone-1", "zone-2"]**)。
controlPlane.platform.openstack.rootVolume.zones	コントロールプレーンマシンの root ボリュームをインストールするアベイラビリティーゾーン。この値を設定しない場合、インストーラーはデフォルトのアベイラビリティーゾーンを選択します。	文字列の一覧 (例: **["zone-1", "zone-2"]**)。
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

パラメーター	説明	値
platform.openstack.clusterOSImageProperties	Glance のインストーラーでアップロードされた ClusterOSImage に追加するプロパティー。このプロパティーは、**platform.openstack.clusterOSImage** が既存の Glance イメージに設定されている場合は無視されます。 このプロパティーを使用し、ノードあたり 26 PV の RHOSP のデフォルト永続ボリューム (PV) の制限を超過することができます。制限を超えるには、**hw_scsi_model** プロパティーの値を **virtio-scsi** に設定し、**hw_disk_bus** の値を **scsi** に設定します。 このプロパティーを使用し、**hw_qemu_guest_agent** プロパティーを **yes** の値で追加して QEMU ゲストエージェントを有効にすることもできます。	キーと値の文字列のペアの一覧。例: **["hw_scsi_model": "virtio-scsi", "hw_disk_bus": "scsi"]**

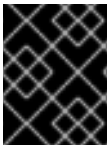
platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.ingressFloatingIP	Ingress ポートに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。

パラメーター	説明	値
platform.openstack.apiFloatingIP	API ロードバランサーに関連付ける既存の Floating IP アドレス。このプロパティーを使用するには、**platform.openstack.externalNetwork** プロパティーも定義する必要があります。	IP アドレス (例: **128.0.0.1**)。
platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。

12.8.8.3. 制限された OpenStack インストールのカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。



重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform: {}
```

```
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineCIDR: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    region: region1
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    apiFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
additionalTrustBundle: |

  -----BEGIN CERTIFICATE-----

  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

  -----END CERTIFICATE-----

imageContentSources:
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

12.8.9. コンピュートマシンのアフィニティーの設定

オプションで、インストール時にコンピュートマシンのアフィニティーポリシーを設定できます。インストーラーは、デフォルトでコンピュートマシンのアフィニティーポリシーを選択しません。

インストール後に特定の RHOSP サーバーグループを使用するマシンセットを作成することもできます。

注記

コントロールプレーンマシンは、**soft-anti-affinity** ポリシーで作成されます。

ヒント

RHOSP インスタンスのスケジューリングおよび配置 の詳細は、RHOSP のドキュメントを参照してください。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. RHOSP コマンドラインインターフェイスを使用して、コンピュートマシンのサーバーグループを作成します。以下に例を示します。

```
$ openstack \
    --os-compute-api-version=2.15 \
    server group create \
    --policy anti-affinity \
    my-openshift-worker-group
```

   詳細は、 **server group create** コマンドのドキュメント を参照してください。

2. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

   ここでは、以下のようになります。

   installation_directory
   クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

3. **MachineSet** 定義ファイルの **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** を作成します。

4. **spec.template.spec.providerSpec.value** プロパティーの下にある定義に、プロパティー **serverGroupID** を追加します。以下に例を示します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_ID>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  template:
    metadata:
```

```
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID>
      machine.openshift.io/cluster-api-machine-role: <node_role>
      machine.openshift.io/cluster-api-machine-type: <node_role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role>
  spec:
    providerSpec:
      value:
        apiVersion: openstackproviderconfig.openshift.io/v1alpha1
        cloudName: openstack
        cloudsSecret:
          name: openstack-cloud-credentials
          namespace: openshift-machine-api
        flavor: <nova_flavor>
        image: <glance_image_name_or_location>
        serverGroupID: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (1)
        kind: OpenstackProviderSpec
        networks:
        - filter: {}
          subnets:
          - filter:
              name: <subnet_name>
              tags: openshiftClusterID=<infrastructure_ID>
        securityGroups:
        - filter: {}
          name: <infrastructure_ID>-<node_role>
        serverMetadata:
          Name: <infrastructure_ID>-<node_role>
          openshiftClusterID: <infrastructure_ID>
        tags:
        - openshiftClusterID=<infrastructure_ID>
        trunk: true
        userDataSecret:
          name: <node_role>-user-data
        availabilityZone: <optional_openstack_availability_zone>
```

(1) サーバーグループの UUID をここに追加します。

5. オプション: **manifests/99_openshift-cluster-api_worker-machineset-0.yaml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

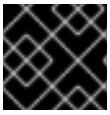
クラスターのインストール時に、インストーラーは変更した **MachineSet** 定義を使用して RHOSP サーバーグループ内にコンピュートマシンを作成します。

12.8.10. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 **~/.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.8.11. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

インストール時に Floating IP アドレス (FIP) を使用して OpenShift Container Platform API およびアプリケーションのアクセスを設定できます。FIP を設定せずにインストールを完了することもできますが、インストーラーは API またはアプリケーションを外部からアクセスする方法を設定しません。

12.8.11.1. floating IP アドレスを使ったアクセスの有効化

OpenShift Container Platform API およびクラスターアプリケーションへの外部アクセス用に Floating IP (FIP) アドレスを作成します。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>"
<external_network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external_network>
```

3. API および Ingress FIP の DNS サーバーに、これらのパターンに準拠するレコードを追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御していない場合は、次のようなクラスタードメイン名を **/etc/hosts** ファイルに追加することで、クラスターにアクセスできます。

- **<api_floating_ip> api.<cluster_name>.<base_domain>**
- **<application_floating_ip> grafana-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> oauth-openshift.apps.<cluster_name>.<base_domain>**
- **<application_floating_ip> console-openshift-console.apps.<cluster_name>.<base_domain>**
- **application_floating_ip integrated-oauth-server-openshift-authentication.apps.<cluster_name>.<base_domain>**

/etc/hosts ファイル内のクラスタードメイン名により、クラスターの Web コンソールおよび監視インターフェイスへのローカルアクセスが許可されます。**kubectl** または **oc** を使用することもできます。<application_floating_ip> を指す追加のエントリーを使用して、ユーザーアプリケーションにアクセスできます。このアクションにより、API およびアプリケーションは他のユーザーがアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

4. FIP を、以下のパラメーターの値として **install-config.yaml** ファイルに追加します。
   - **platform.openstack.ingressFloatingIP**
   - **platform.openstack.apiFloatingIP**

これらの値を使用する場合には、**install-config.yaml** ファイルの **platform.openstack.externalNetwork** パラメーターの値として外部ネットワークを入力する必要もあります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

12.8.11.2. Floating IP アドレスなしでのインストールの完了

Floating IP アドレスを指定せずに OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールすることができます。

install-config.yaml ファイルで以下のパラメーターを定義しないでください。

- **platform.openstack.ingressFloatingIP**
- **platform.openstack.apiFloatingIP**

外部ネットワークを提供できない場合は、**platform.openstack.externalNetwork** を空白のままにすることもできます。**platform.openstack.externalNetwork** の値を指定しない場合はルーターが作成されず、追加のアクションがない場合は、インストーラーは Glance からのイメージの取得に失敗します。外部接続を独自に設定する必要があります。

Floating IP アドレスまたは名前解決がないために、クラスター API に到達できないシステムからインストーラーを実行すると、インストールに失敗します。このような場合にインストールが失敗するのを防ぐために、プロキシーネットワークを使用するか、マシンと同じネットワークにあるシステムからインストーラーを実行できます。

注記

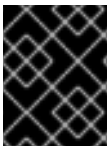
API および Ingress ポートの DNS レコードを作成して、名前解決を有効にできます。以下に例を示します。

```
api.<cluster_name>.<base_domain>.  IN  A  <api_port_IP>
*.apps.<cluster_name>.<base_domain>. IN  A <ingress_port_IP>
```

DNS サーバーを制御しない場合は、**/etc/hosts** ファイルにレコードを追加できます。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

12.8.12. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

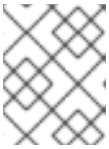
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

重要

- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

- 24 時間証明書はクラスターのインストール後 16 時間から 22 時間にローテーションするため、Ignition 設定ファイルは、生成後 12 時間以内に使用することをお勧めします。12 時間以内に Ignition 設定ファイルを使用することにより、インストール中に証明書の更新が実行された場合のインストールの失敗を回避できます。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

12.8.13. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

12.8.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

12.8.15. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

12.8.16. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 し

て、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

12.8.17. 次のステップ

- クラスターをカスタマイズします。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- RHOSP が Floating IP アドレス上でアプリケーショントラフィックを受け入れるように設定しなかった場合には、RHOSP のアクセスを Floating IP アドレスで設定 します。

12.9. OPENSTACK でのクラスターのアンインストール

Red Hat OpenStack Platform (RHOSP) にデプロイしたクラスターを削除できます。

12.9.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

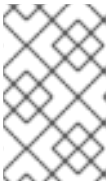
- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

12.10. 独自のインフラストラクチャーからの RHOSP のクラスターのアンインストール

ユーザーによってプロビジョニングされたインフラストラクチャーの Red Hat OpenStack Platform (RHOSP) にデプロイしたクラスターを削除することができます。

12.10.1. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでの削除プロセスを簡素化する Ansible Playbook には、複数の Python モジュールが必要です。プロセスを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされている。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

12.10.2. 独自のインフラストラクチャーを使用する RHOSP からのクラスターの削除

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) の OpenShift Container Platform クラスターを削除できます。削除プロセスを迅速に完了するには、複数の Ansible Playbook を実行します。

前提条件

- Python 3 がマシンにインストールされている。
- Playbook 依存関係のダウンロードでモジュールをダウンロードしている。
- クラスターのインストールに使用した Playbook がある。
- 対応するインストール Playbook に加えた変更を反映するように **down-** の接頭辞が付けられた Playbook を変更している。たとえば、**bootstrap.yaml** ファイルへの変更は **down-bootstrap.yaml** ファイルに反映されます。
- すべての Playbook は共通ディレクトリーにある。

手順

1. コマンドラインで、ダウンロードした Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml  \
 down-bootstrap.yaml      \
 down-control-plane.yaml  \
 down-compute-nodes.yaml  \
 down-load-balancers.yaml \
 down-network.yaml        \
 down-security-groups.yaml
```

2. OpenShift Container Platform インストールに対して加えた DNS レコードの変更を削除します。

OpenShift Container Platform はお使いのインフラストラクチャーから削除されます。

第13章 RHV へのインストール

13.1. RED HAT VIRTUALIZATION (RHV) へのインストールの準備

13.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- Support Matrix for OpenShift Container Platform on Red Hat Virtualization (RHV) に記載のあるサポートされるバージョンの組み合わせを使用できる。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

13.1.2. RHV に OpenShift Container Platform をインストールする方法の選択

OpenShift Container Platform をインストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform は、ユーザーによってプロビジョニングされるインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

13.1.2.1. インストーラーでプロビジョニングされるインフラストラクチャーへのクラスターのインストール

以下の方法のいずれかを使用して、OpenShift Container Platform インストールプログラムでプロビジョニングされる Red Hat Virtualization (RHV) 仮想マシンに、クラスターをインストールできます。

- **クラスターの RHV へのクイックインストール**: OpenShift Container Platform インストールプログラムでプロビジョニングされる RHV 仮想マシンに OpenShift Container Platform をクイックインストールできます。
- **カスタマイズによる RHV へのクラスターのインストール**: RHV のインストーラーでプロビジョニングされるゲストに、カスタマイズされた OpenShift Container Platform クラスターをインストールできます。インストールプログラムは、インストールの段階で一部のカスタマイズを適用できるようにします。その他の数多くのカスタマイズオプションは、インストール後 に利用できます。

13.1.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

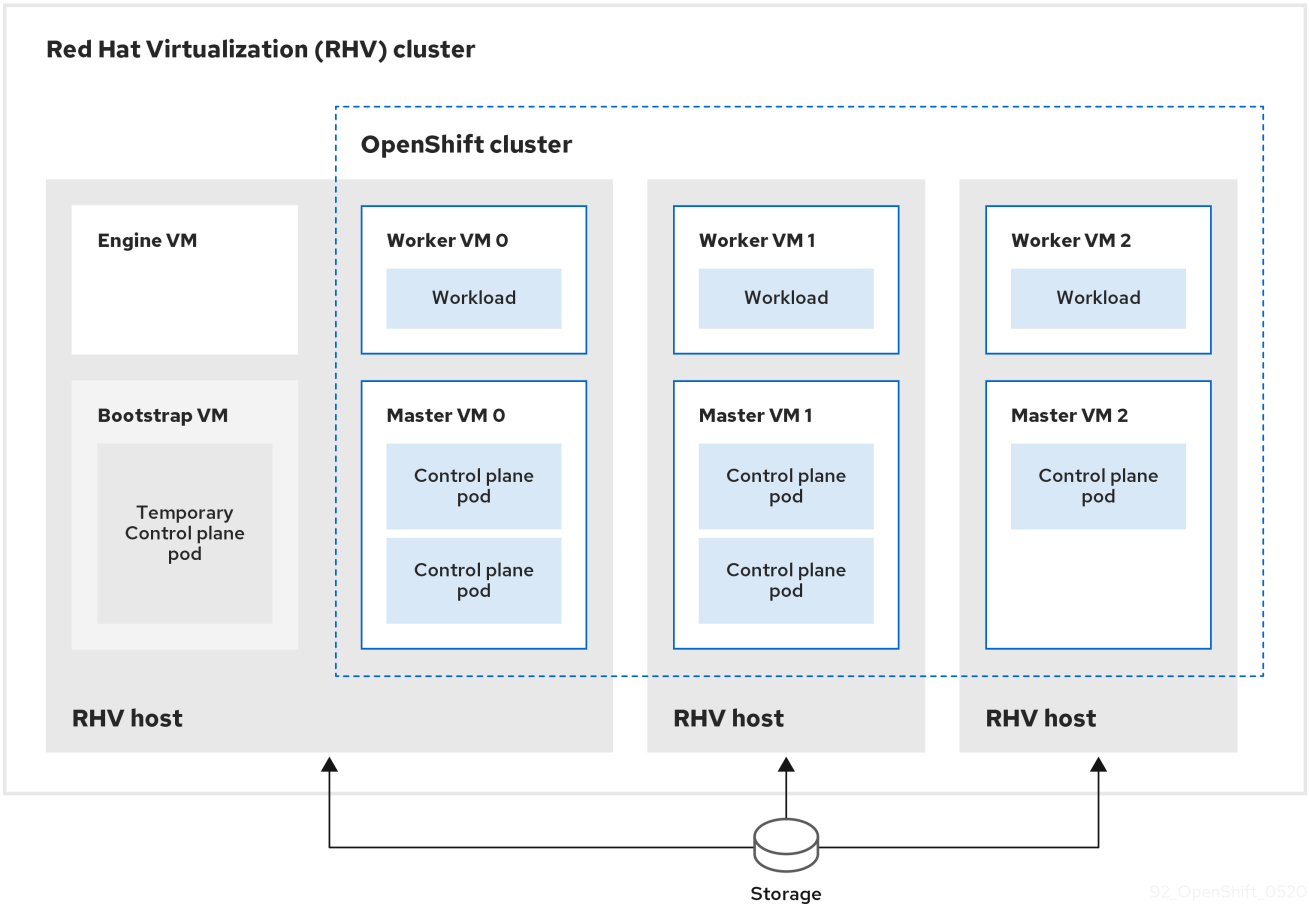
以下の方法のいずれかを使用して、独自にプロビジョニングする RHV 仮想マシンにクラスターをインストールできます。

- **ユーザーによってプロビジョニングされるインフラストラクチャーでの RHV へのクラスターのインストール**: 独自にプロビジョニングする RHV 仮想マシンに OpenShift Container Platform をインストールできます。提供される Ansible Playbook を使用してインストールを支援することができます。

- **ネットワークが制限された環境での RHV へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを作成して、ネットワークが制限された環境またはネットワークの非接続環境で OpenShift Container Platform を RHV にインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないユーザーによってプロビジョニングされるクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

13.2. RHV へのクラスターのクイックインストール

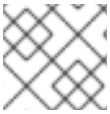
以下の図に示されるように、デフォルトの、カスタマイズされていない OpenShift Container Platform クラスターを Red Hat Virtualization (RHV) クラスターにすばやくインストールできます。



インストールプログラムは、インストーラーでプロビジョニングされるインフラストラクチャーを使用してクラスターの作成およびデプロイを自動化します。

デフォルトのクラスターをインストールするには、環境を準備し、インストールプログラムを実行してプロンプトに応答します。次に、インストールプログラムは OpenShift Container Platform クラスターを作成します。

デフォルトクラスターの代替インストール方法については、カスタマイズによるクラスターのインストール について参照してください。



注記

このインストールプログラムは、Linux および macOS でのみ利用できます。

13.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- Support Matrix for OpenShift Container Platform on Red Hat Virtualization (RHV) に記載のあるサポートされるバージョンの組み合わせを使用できる。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

13.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

13.2.3. RHV 環境の要件

OpenShift Container Platform バージョン 4.8 クラスターをインストールし、実行するには、RHV 環境が以下の要件を満たしている必要があります。

これらの要件を満たさないと、インストールまたはプロセスが失敗する可能性があります。さらに、これらの要件を満たしていないと、OpenShift Container Platform クラスターはインストールしてから数日または数週間後に失敗する可能性があります。

CPU、メモリー、ストレージリソースについての以下の要件は、インストールプログラムが作成する仮想マシンのデフォルト数で乗算した **デフォルト** 値に基づいています。これらのリソースは、RHV 環境が OpenShift Container Platform 以外の操作に使用するものに **加え**、利用可能でなければなりません。

デフォルトでは、インストールプログラムは 7 つの仮想マシンをインストールプロセスで作成します。まず、ブートストラップ仮想マシンを作成し、OpenShift Container Platform クラスターの残りの部分を作成する間に一時サービスとコントロールプレーンを提供します。インストールプログラムがクラスターの作成を終了すると、ブートストラップマシンが削除され、そのリソースが解放されます。

RHV 環境の仮想マシン数を増やす場合は、リソースを適宜増やす必要があります。

要件

- RHV のバージョンは 4.4 である。
- RHV 環境に **Up** 状態のデータセンターが 1 つあること。
- RHV データセンターに RHV クラスターが含まれていること。
- RHV クラスターに OpenShift Container Platform クラスター専用の以下のリソースがあること。
  - 最小 28 vCPU: インストール時に作成される 7 仮想マシンのそれぞれに 4 vCPU。
  - 以下を含む 112 GiB 以上の RAM。
    - 一時的なコントロールプレーンを提供するブートストラップマシン用に 16 GiB 以上。
    - コントロールプレーンを提供する 3 つのコントロールプレーンマシンのそれぞれに 16 GiB 以上。
    - アプリケーションワークロードを実行する 3 つのコンピュートマシンのそれぞれに 16 GiB 以上。
- RHV ストレージドメインは、これらの etcd バックエンドのパフォーマンス要件 を満たす必要があります。
- アフィニティーグループのサポートの場合: RHV クラスター内の 3 つ以上のホスト。必要に応じて、アフィニティーグループを無効にすることができます。詳細は、**カスタマイズによる RHV へのクラスターのインストール**の**実稼働以外のラボセットアップのすべてのアフィニティーグループを削除する例**を参照してください。
- 実稼働環境では、各仮想マシンに 120 GiB 以上が必要です。そのため、ストレージドメインはデフォルトの OpenShift Container Platform クラスターに 840 GiB 以上を提供する必要があります。リソースに制約のある環境または非実稼働環境では、各仮想マシンに 32 GiB 以上を指定する必要があるため、ストレージドメインにはデフォルトの OpenShift Container Platform クラスター用に 230 GiB 以上が必要になります。
- インストールおよび更新中に Red Hat Ecosystem Catalog からイメージをダウンロードするには、RHV クラスターがインターネット接続にアクセスできる必要があります。また、サブスクリプションおよびエンタイトルメントプロセスを単純化するために Telemetry サービスにもインターネット接続が必要です。
- RHV クラスターには、RHV Manager の REST API にアクセスできる仮想ネットワークが必要です。インストーラーが作成する仮想マシンが DHCP を使用して IP アドレスを取得するため、DHCP がこのネットワークで有効にされていることを確認します。
- ターゲット RHV クラスターに OpenShift Container Platform クラスターをインストールし、管理するための以下の最小限の権限を持つユーザーアカウントおよびグループ。
  - **DiskOperator**
  - **DiskCreator**
  - **UserTemplateBasedVm**
  - **TemplateOwner**

- **TemplateCreator**
- ターゲットクラスターの **ClusterAdmin**

警告

最小権限の原則を適用します。インストールプロセスで RHV で **SuperUser** 権限を持つ管理者アカウントを使用することを避けます。インストールプログラムは、ユーザーが指定する認証情報を、危険にさらされる可能性のある一時的な **ovirt-config.yaml** ファイルに保存します。

関連情報

- 実稼働以外のラボセットアップのすべてのアフィニティーグループを削除する例

13.2.4. RHV 環境の要件の確認

RHV 環境が OpenShift Container Platform クラスターをインストールし、実行するための要件を満たしていることを確認します。これらの要件を満たさないと、エラーが発生する可能性があります。

重要

これらの要件は、インストールプログラムがコントロールプレーンおよびコンピュートマシンの作成に使用するデフォルトのリソースに基づいています。これらのリソースには、vCPU、メモリー、およびストレージが含まれます。これらのリソースを変更するか、または OpenShift Container Platform マシンの数を増やす場合は、これらの要件を適宜調整します。

手順

1. RHV のバージョンが OpenShift Container Platform バージョン 4.8 のインストールをサポートすることを確認します。
   a. RHV Administration Portal の右上にある ? ヘルプアイコンをクリックし、**About** を選択します。
   b. 開かれるウィンドウで、**RHV ソフトウェアのバージョン** をメモします。
   c. RHV のバージョンが 4.4 であることを確認します。サポートされるバージョンの組み合わせについての詳細は、Support Matrix for OpenShift Container Platform on RHV を参照してください。
2. データセンター、クラスター、およびストレージを検査します。
   a. RHV 管理ポータルで、**Compute → Data Centers** をクリックします。
   b. OpenShift Container Platform をインストールする予定のデータセンターにアクセスできることを確認します。
   c. そのデータセンターの名前をクリックします。

d. データセンターの詳細の **Storage** タブで、OpenShift Container Platform をインストールする予定のストレージドメインが **Active** であることを確認します。

e. 後で使用できるように **ドメイン名** を記録します。

f. **空き領域** に 230 GiB 以上あることを確認します。

g. ストレージドメインが これらの etcd バックエンドのパフォーマンス要件 を満たしていることを確認します。これは、fio パフォーマンスベンチマークツールを使用して測定できます。

h. データセンターの詳細で、**Clusters** タブをクリックします。

i. OpenShift Container Platform をインストールする予定の RHV クラスターを見つけます。後で使用できるようにクラスター名を記録します。

3. RHV ホストリソースを確認します。

a. RHV 管理ポータルで、**Compute** > **Clusters** をクリックします。

b. OpenShift Container Platform をインストールする予定のクラスターをクリックします。

c. クラスターの詳細で、**Hosts** タブをクリックします。

d. ホストを検査し、それらに OpenShift Container Platform クラスター **専用** として利用可能な **論理 CPU コア** の合計が 28 つ以上であることを確認します。

e. 後で使用できるように、利用可能な **論理 CPU コア** の数を記録します。

f. これらの CPU コアが分散され、インストール時に作成された 7 つの仮想マシンのそれぞれに 4 つのコアを持たせることができることを確認します。

g. ホストには、以下の OpenShift Container Platform マシンのそれぞれの要件を満たすように **新規仮想マシンをスケジュールするための最大空きメモリー** として 112 GiB があることを確認します。

- ブートストラップマシンに 16 GiB が必要です。
- 3 つのコントロールプレーンマシンのそれぞれに 16 GiB が必要です。
- 3 つのコンピュートマシンのそれぞれに 16 GiB が必要です。

h. 後で使用できるように **新規仮想マシンをスケジュールするための最大空きメモリー** の量を記録します。

4. OpenShift Container Platform をインストールするための仮想ネットワークが RHV Manager の REST API にアクセスできることを確認します。このネットワーク上の仮想マシンから、RHV Manager の REST API に到達するために curl を使用します。

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** については、RHV で OpenShift Container Platform クラスターを作成および管理する権限を持つ RHV アカウントのユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。**<password>** に、そのユーザー名のパスワードを指定します。

2 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

13.2.5. RHV でのネットワーク環境の準備

OpenShift Container Platform クラスターの 2 つの静的 IP アドレスを設定し、これらのアドレスを使用して DNS エントリーを作成します。

手順

1. 2 つの静的 IP アドレスを予約します。
   a. OpenShift Container Platform をインストールするネットワークで、DHCP リースプール外にある 2 つの静的 IP アドレスを特定します。
   b. このネットワーク上のホストに接続し、それぞれの IP アドレスが使用されていないことを確認します。たとえば、Address Resolution Protocol (ARP) を使用して、IP アドレスのいずれにもエントリーがないことを確認します。

      ```
      $ arp 10.35.1.19
      ```

      出力例

      ```
      10.35.1.19 (10.35.1.19) -- no entry
      ```

   c. ネットワーク環境の標準的な方法に従って、2 つの静的 IP アドレスを予約します。
   d. 今後の参照用にこれらの IP アドレスを記録します。
2. 以下の形式を使用して、OpenShift Container Platform REST API およびアプリケーションドメイン名の DNS エントリーを作成します。

```
api.<cluster-name>.<base-domain>   <ip-address> 1
*.apps.<cluster-name>.<base-domain>   <ip-address> 2
```

1 **<cluster-name>**、**<base-domain>**、および **<ip-address>** には、クラスター名、ベースドメイン、および OpenShift Container Platform API の静的 IP アドレスを指定します。

2 Ingress およびロードバランサー用に OpenShift Container Platform アプリケーションのクラスター名、ベースドメイン、および静的 IP アドレスを指定します。

以下に例を示します。

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```

13.2.6. OpenShift Container Platform OpenStack クラスターの RHV への非セキュアモードでのインストール

デフォルトで、インストーラーは CA 証明書を作成し、確認を求めるプロンプトを出し、インストール時に使用する証明書を保存します。これは、手動で作成したりインストールしたりする必要はありません。

推奨されていませんが、OpenShift Container Platform を RHV に **非セキュアモード** でインストールして、この機能を上書きし、証明書の検証なしに OpenShift Container Platform をインストールすることができます。

警告

非セキュア モードでのインストールは推奨されていません。これにより、攻撃者が中間者 (Man-in-the-Middle) 攻撃を実行し、ネットワーク上の機密の認証情報を取得できる可能性が生じるためです。

手順

1. **~/.ovirt/ovirt-config.yaml** という名前のファイルを作成します。
2. 以下の内容を **ovirt-config.yaml** に追加します。

```
ovirt_url: https://ovirt.example.com/ovirt-engine/api (1)
ovirt_fqdn: ovirt.example.com (2)
ovirt_pem_url: ""
ovirt_username: ocpadmin@internal
ovirt_password: super-secret-password (3)
ovirt_insecure: true
```

(1) oVirt エンジンのホスト名またはアドレスを指定します。

(2) oVirt エンジンの完全修飾ドメイン名を指定します。

(3) oVirt エンジンの管理者パスワードを指定します。

3. インストーラーを実行します。

13.2.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

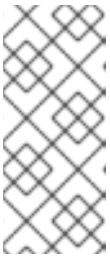
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

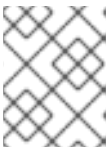
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

13.2.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

13.2.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- インストーラーを実行するマシンから **ovirt-imageio** ポートを Manager へのポートを開放する。デフォルトでは、ポートは **54322** です。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

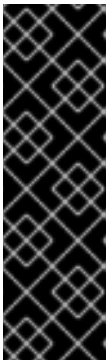
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

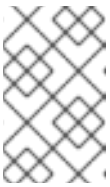


重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

インストールプログラムのプロンプトに対応します。

a. オプション: **SSH Public Key** には、 パスワードなしのパブリックキー (例: **~/.ssh/id_rsa.pub**) を選択します。このキーは、新規 OpenShift Container Platform クラスターとの接続を認証します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターには、**ssh-agent** プロセスが使用する SSH キーを選択します。

b. **Platform** には、**ovirt** を選択します。

c. **Engine FQDN[:PORT]** に、RHV 環境の完全修飾ドメイン名 (FQDN) を入力します。以下に例を示します。

```
rhv-env.virtlab.example.com:443
```

d. インストーラーは CA 証明書を自動的に生成します。**Would you like to use the above certificate to connect to the Manager?** では、**y** または **N** のいずれかで回答します。**N** と回答する場合は、OpenShift Container Platform を非セキュアモードでインストールする必要があります。

e. **Engine username** には、この形式を使用して RHV 管理者のユーザー名およびプロファイルを入力します。

```
<username>@<profile> 1
```

1 **<username>** に、RHV 管理者のユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。例: **admin@internal**

f. **Engine password** に、RHV 管理者パスワードを入力します。

g. **Cluster** には、OpenShift Container Platform をインストールするための RHV クラスターを選択します。

h. **Storage domain** には、OpenShift Container Platform をインストールするためのストレージドメインを選択します。

i. **Network** には、RHV Manager REST API へのアクセスのある仮想ネットワークを選択します。

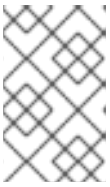
j. **Internal API Virtual IP** に、クラスターの REST API とは別の静的 IP アドレスを入力します。

k. **Ingress virtual IP** に、ワイルドカードアプリドメイン用に予約した静的 IP アドレスを入力します。

l. **Base Domain** に、OpenShift Container Platform クラスターのベースドメインを入力します。このクラスターが外部に公開される場合、これは DNS インフラストラクチャーが認識する有効なドメインである必要があります。たとえば、**virtlab.example.com** を入力します。

m. **Cluster Name** に、クラスターの名前を入力します。例: **my-cluster**OpenShift Container Platform REST API およびアプリケーションドメイン名向けに作成した外部登録/解決可能な DNS エントリーのクラスター名を使用します。インストールプログラムは、この名前を RHV 環境のクラスターにも指定します。

n. **Pull secret** には、先にダウンロードした **pull-secret.txt** ファイルからプルシークレットをコピーし、ここに貼り付けます。Red Hat OpenShift Cluster Manager から同じプルシークレット のコピーを取得することもできます。



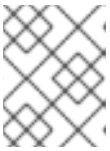
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

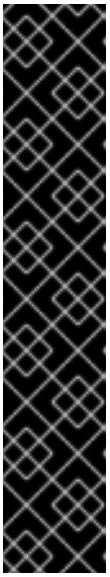
出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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

重要

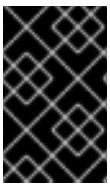
インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

重要

クラスターのインストールに必要な手順を完了している必要があります。残りの手順では、クラスターを検証し、インストールのトラブルシューティングを行う方法を説明します。

13.2.10. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

詳細は、Getting started with the OpenShift CLI を参照してください。

13.2.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

関連情報

- OpenShift Container Platform Web コンソールへのアクセスおよび詳細については、Web コンソールへのアクセス を参照してください。

13.2.12. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

トラブルシューティング

インストールが失敗すると、インストールプログラムがタイムアウトし、エラーメッセージが表示されます。詳細は、インストールに関する問題のトラブルシューティング を参照してください。

13.2.13. RHV での OpenShift Container Platform Web コンソールへのアクセス

OpenShift Container Platform クラスターの初期化後に、OpenShift Container Platform Web コンソールにログインできます。

手順

1. オプション: Red Hat Virtualization (RHV) Administration Portal で、**Compute → Cluster** を開きます。
2. インストールプログラムが仮想マシンを作成することを確認します。
3. インストールプログラムが実行されているコマンドラインに戻ります。インストールプログラムが完了すると、OpenShift Container Platform Web コンソールにログインするためのユーザー名およびパスワードの一時パスワードが表示されます。
4. ブラウザーから OpenShift Container Platform の Web コンソールの URL を開きます。URL は以下の形式を使用します。

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

1 **<clustername>.<basedomain>** に、クラスター名およびベースドメインを指定します。

以下に例を示します。

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

13.2.14. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

13.2.15. Red Hat Virtualization (RHV) へのインストールに関するよくある問題のトラブルシューティング

以下に、一般的な問題およびそれらについて考えられる原因および解決策を記載します。

13.2.15.1. CPU 負荷が増大し、ノードが Not Ready 状態になる

- **現象**: CPU 負荷が大幅に増大し、ノードが **Not Ready** 状態に切り替わり始める。
- **原因**: 特にコントロールプレーンノード (別名マスターノード) の場合、ストレージドメインのレイテンシーが高すぎる可能性があります。
- **解決策**:
  Kubelet サービスを再起動して、ノードを再度 Ready 状態にします。

  ```
  $ systemctl restart kubelet
  ```

  OpenShift Container Platform メトリクスサービスを検査します。これは、etcd ディスクの同期期間などの有用なデータを収集し、これについて報告します。クラスターが機能している場合は、このデータを使用して、ストレージのレイテンシーまたはスループットが根本的な問題かどうかを判断します。その場合、レイテンシーが短く、スループットの高いストレージリソースの使用を検討してください。

  未加工メトリクスを取得するには、kubeadmin または cluster-admin 権限を持つユーザーで以下のコマンドを実行します。

  ```
  $ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics
  ```

  詳細は、Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x を参照してください。

13.2.15.2. OpenShift Container Platform クラスター API に接続できない

- **現象**: インストールプログラムは完了するが、OpenShift Container Platform クラスター API は利用できない。ブートストラップの仮想マシンは、ブートストラッププロセスの完了後も起動した状態になります。以下のコマンドを入力すると、応答がタイムアウトします。

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- **原因**: ブートストラップ仮想マシンがインストールプログラムによって削除されず、クラスターの API IP アドレスをリリースしない。
- **解決策**: **wait-for** サブコマンドを使用して、ブートストラッププロセスの完了時に通知を受信する。

```
$ ./openshift-install wait-for bootstrap-complete
```

ブートストラッププロセスが完了したら、ブートストラップ仮想マシンを削除します。

```
$ ./openshift-install destroy bootstrap
```

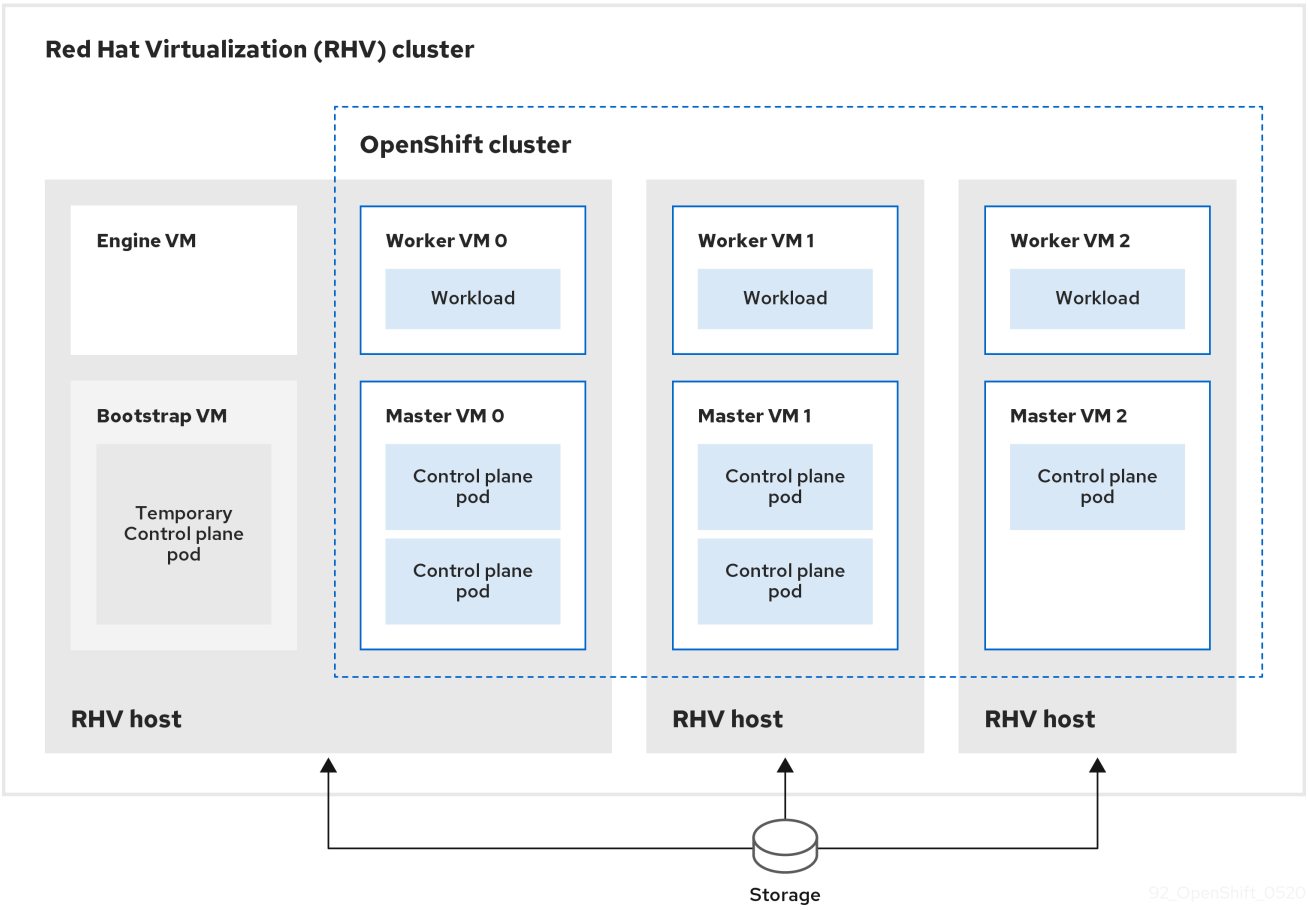
13.2.16. インストール後のタスク

OpenShift Container Platform クラスターの初期化後に、以下のタスクを実行できます。

- オプション: デプロイメント後に、OpenShift Container Platform で Machine Config Operator (MCO) を使用して SSH キーを追加するか、または置き換えます。
- オプション: **kubeadmin** ユーザーを削除します。代わりに、認証プロバイダーを使用して cluster-admin 権限を持つユーザーを作成します。

13.3. カスタマイズによる RHV へのクラスターのインストール

以下の図に示されるように、OpenShift Container Platform クラスターを Red Hat Virtualization (RHV) でカスタマイズし、インストールすることができます。



インストールプログラムは、インストーラーでプロビジョニングされるインフラストラクチャーを使用してクラスターの作成およびデプロイを自動化します。

カスタマイズされたクラスターをインストールするには、環境を準備し、以下の手順を実行します。

1. インストールプログラムを実行し、そのプロンプトに応答して、インストール設定ファイル **install-config.yaml** ファイルを作成します。
2. **install-config.yaml** ファイルでパラメーターを検査し、変更します。
3. **install-config.yaml** ファイルの作業用コピーを作成します。
4. **install-config.yaml** ファイルのコピーを使ってインストールプログラムを実行します。

次に、インストールプログラムは OpenShift Container Platform クラスターを作成します。

カスタマイズされたクラスターをインストールする代替方法については、デフォルトのクラスターのインストール を参照してください。

注記

このインストールプログラムは、Linux および macOS でのみ利用できます。

13.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

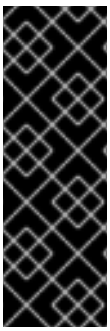
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- Support Matrix for OpenShift Container Platform on Red Hat Virtualization (RHV) に記載のあるサポートされるバージョンの組み合わせを使用できる。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

13.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

13.3.3. RHV 環境の要件

OpenShift Container Platform バージョン 4.8 クラスターをインストールし、実行するには、RHV 環境が以下の要件を満たしている必要があります。

これらの要件を満たさないと、インストールまたはプロセスが失敗する可能性があります。さらに、これらの要件を満たしていないと、OpenShift Container Platform クラスターはインストールしてから数日または数週間後に失敗する可能性があります。

CPU、メモリー、ストレージリソースについての以下の要件は、インストールプログラムが作成する仮想マシンのデフォルト数で乗算した **デフォルト** 値に基づいています。これらのリソースは、RHV 環境が OpenShift Container Platform 以外の操作に使用するものに **加え**、利用可能でなければなりません。

デフォルトでは、インストールプログラムは 7 つの仮想マシンをインストールプロセスで作成します。まず、ブートストラップ仮想マシンを作成し、OpenShift Container Platform クラスターの残りの部分を作成する間に一時サービスとコントロールプレーンを提供します。インストールプログラムがクラスターの作成を終了すると、ブートストラップマシンが削除され、そのリソースが解放されます。

RHV 環境の仮想マシン数を増やす場合は、リソースを適宜増やす必要があります。

要件

- RHV のバージョンは 4.4 である。
- RHV 環境に **Up** 状態のデータセンターが 1 つあること。
- RHV データセンターに RHV クラスターが含まれていること。
- RHV クラスターに OpenShift Container Platform クラスター専用の以下のリソースがあること。
    - 最小 28 vCPU: インストール時に作成される 7 仮想マシンのそれぞれに 4 vCPU。
    - 以下を含む 112 GiB 以上の RAM。
        - 一時的なコントロールプレーンを提供するブートストラップマシン用に 16 GiB 以上。
        - コントロールプレーンを提供する 3 つのコントロールプレーンマシンのそれぞれに 16 GiB 以上。
        - アプリケーションワークロードを実行する 3 つのコンピュートマシンのそれぞれに 16 GiB 以上。
- RHV ストレージドメインは、これらの etcd バックエンドのパフォーマンス要件 を満たす必要があります。
- アフィニティーグループのサポートの場合:
  ワーカーまたはコントロールプレーンごとに 1 台の物理マシン。ワーカーとコントロールプレーンは、同じ物理マシン上に置くことができます。たとえば、3 つのワーカーと 3 つのコントロールプレーンがある場合、3 台の物理マシンが必要です。4 つのワーカーと 3 つのコントロールプレーンがある場合は、4 台の物理マシンが必要です。
    - 強い非アフィニティーの場合 (デフォルト): 最低 3 台の物理マシン。3 つを超えるワーカーノードの場合、ワーカーまたはコントロールプレーンごとに 1 台の物理マシン。ワーカーとコントロールプレーンは、同じ物理マシン上に置くことができます。
    - カスタムアフィニティーグループの場合: リソースが、定義するアフィニティーグループルールに適していることを確認します。
- 実稼働環境では、各仮想マシンに 120 GiB 以上が必要です。そのため、ストレージドメインはデフォルトの OpenShift Container Platform クラスターに 840 GiB 以上を提供する必要があります。リソースに制約のある環境または非実稼働環境では、各仮想マシンに 32 GiB 以上を指定する必要があるため、ストレージドメインにはデフォルトの OpenShift Container Platform クラスター用に 230 GiB 以上が必要になります。
- インストールおよび更新中に Red Hat Ecosystem Catalog からイメージをダウンロードするには、RHV クラスターがインターネット接続にアクセスできる必要があります。また、サブスクリプションおよびエンタイトルメントプロセスを単純化するために Telemetry サービスにもインターネット接続が必要です。
- RHV クラスターには、RHV Manager の REST API にアクセスできる仮想ネットワークが必要です。インストーラーが作成する仮想マシンが DHCP を使用して IP アドレスを取得するため、DHCP がこのネットワークで有効にされていることを確認します。
- ターゲット RHV クラスターに OpenShift Container Platform クラスターをインストールし、管理するための以下の最小限の権限を持つユーザーアカウントおよびグループ。
    - **DiskOperator**

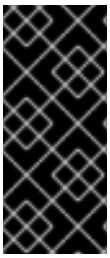
- **DiskCreator**
- **UserTemplateBasedVm**
- **TemplateOwner**
- **TemplateCreator**
- ターゲットクラスターの **ClusterAdmin**

警告

最小権限の原則を適用します。インストールプロセスで RHV で **SuperUser** 権限を持つ管理者アカウントを使用することを避けます。インストールプログラムは、ユーザーが指定する認証情報を、危険にさらされる可能性のある一時的な **ovirt-config.yaml** ファイルに保存します。

13.3.4. RHV 環境の要件の確認

RHV 環境が OpenShift Container Platform クラスターをインストールし、実行するための要件を満たしていることを確認します。これらの要件を満たさないと、エラーが発生する可能性があります。



重要

これらの要件は、インストールプログラムがコントロールプレーンおよびコンピュートマシンの作成に使用するデフォルトのリソースに基づいています。これらのリソースには、vCPU、メモリー、およびストレージが含まれます。これらのリソースを変更するか、または OpenShift Container Platform マシンの数を増やす場合は、これらの要件を適宜調整します。

手順

1. RHV のバージョンが OpenShift Container Platform バージョン 4.8 のインストールをサポートすることを確認します。
   a. RHV Administration Portal の右上にある ? ヘルプアイコンをクリックし、**About** を選択します。
   b. 開かれるウィンドウで、**RHV ソフトウェアのバージョン** をメモします。
   c. RHV のバージョンが 4.4 であることを確認します。サポートされるバージョンの組み合わせについての詳細は、Support Matrix for OpenShift Container Platform on RHV を参照してください。
2. データセンター、クラスター、およびストレージを検査します。
   a. RHV 管理ポータルで、**Compute → Data Centers** をクリックします。
   b. OpenShift Container Platform をインストールする予定のデータセンターにアクセスできることを確認します。

c. そのデータセンターの名前をクリックします。

d. データセンターの詳細の **Storage** タブで、OpenShift Container Platform をインストールする予定のストレージドメインが **Active** であることを確認します。

e. 後で使用できるように **ドメイン名** を記録します。

f. **空き領域** に 230 GiB 以上あることを確認します。

g. ストレージドメインが これらの etcd バックエンドのパフォーマンス要件 を満たしていることを確認します。これは、fio パフォーマンスベンチマークツールを使用して測定できます。

h. データセンターの詳細で、**Clusters** タブをクリックします。

i. OpenShift Container Platform をインストールする予定の RHV クラスターを見つけます。後で使用できるようにクラスター名を記録します。

3. RHV ホストリソースを確認します。

a. RHV 管理ポータルで、**Compute** > **Clusters** をクリックします。

b. OpenShift Container Platform をインストールする予定のクラスターをクリックします。

c. クラスターの詳細で、**Hosts** タブをクリックします。

d. ホストを検査し、それらに OpenShift Container Platform クラスター **専用** として利用可能な **論理 CPU コア** の合計が 28 つ以上であることを確認します。

e. 後で使用できるように、利用可能な **論理 CPU コア** の数を記録します。

f. これらの CPU コアが分散され、インストール時に作成された 7 つの仮想マシンのそれぞれに 4 つのコアを持たせることができることを確認します。

g. ホストには、以下の OpenShift Container Platform マシンのそれぞれの要件を満たすように **新規仮想マシンをスケジュールするための最大空きメモリー** として 112 GiB があることを確認します。

- ブートストラップマシンに 16 GiB が必要です。
- 3 つのコントロールプレーンマシンのそれぞれに 16 GiB が必要です。
- 3 つのコンピュートマシンのそれぞれに 16 GiB が必要です。

h. 後で使用できるように **新規仮想マシンをスケジュールするための最大空きメモリー** の量を記録します。

4. OpenShift Container Platform をインストールするための仮想ネットワークが RHV Manager の REST API にアクセスできることを確認します。このネットワーク上の仮想マシンから、RHV Manager の REST API に到達するために curl を使用します。

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** については、RHV で OpenShift Container Platform クラスターを作成および管理する権限を持つ RHV アカウントのユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ロ

グインページに移動し、 **Profile** ドロップダウンリストで確認できます。**<password>** に、そのユーザー名のパスワードを指定します。

2 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

13.3.5. RHV でのネットワーク環境の準備

OpenShift Container Platform クラスターの 2 つの静的 IP アドレスを設定し、これらのアドレスを使用して DNS エントリーを作成します。

手順

1. 2 つの静的 IP アドレスを予約します。
   a. OpenShift Container Platform をインストールするネットワークで、DHCP リースプール外にある 2 つの静的 IP アドレスを特定します。
   b. このネットワーク上のホストに接続し、それぞれの IP アドレスが使用されていないことを確認します。たとえば、Address Resolution Protocol (ARP) を使用して、IP アドレスのいずれにもエントリーがないことを確認します。

      ```
      $ arp 10.35.1.19
      ```

      出力例

      ```
      10.35.1.19 (10.35.1.19) -- no entry
      ```

   c. ネットワーク環境の標準的な方法に従って、2 つの静的 IP アドレスを予約します。
   d. 今後の参照用にこれらの IP アドレスを記録します。
2. 以下の形式を使用して、OpenShift Container Platform REST API およびアプリケーションドメイン名の DNS エントリーを作成します。

```
api.<cluster-name>.<base-domain>   <ip-address> 1
*.apps.<cluster-name>.<base-domain>   <ip-address> 2
```

1 **<cluster-name>**、**<base-domain>**、および **<ip-address>** には、クラスター名、ベースドメイン、および OpenShift Container Platform API の静的 IP アドレスを指定します。

2 Ingress およびロードバランサー用に OpenShift Container Platform アプリケーションのクラスター名、ベースドメイン、および静的 IP アドレスを指定します。

以下に例を示します。

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```

13.3.6. OpenShift Container Platform OpenStack クラスターの RHV への非セキュアモードでのインストール

デフォルトで、インストーラーは CA 証明書を作成し、確認を求めるプロンプトを出し、インストール時に使用する証明書を保存します。これは、手動で作成したりインストールしたりする必要はありません。

推奨されていませんが、OpenShift Container Platform を RHV に **非セキュアモード** でインストールして、この機能を上書きし、証明書の検証なしに OpenShift Container Platform をインストールすることができます。

警告

非セキュア モードでのインストールは推奨されていません。これにより、攻撃者が中間者 (Man-in-the-Middle) 攻撃を実行し、ネットワーク上の機密の認証情報を取得できる可能性が生じるためです。

手順

1. **~/.ovirt/ovirt-config.yaml** という名前のファイルを作成します。
2. 以下の内容を **ovirt-config.yaml** に追加します。

```
ovirt_url: https://ovirt.example.com/ovirt-engine/api (1)
ovirt_fqdn: ovirt.example.com (2)
ovirt_pem_url: ""
ovirt_username: ocpadmin@internal
ovirt_password: super-secret-password (3)
ovirt_insecure: true
```

(1) oVirt エンジンのホスト名またはアドレスを指定します。

(2) oVirt エンジンの完全修飾ドメイン名を指定します。

(3) oVirt エンジンの管理者パスワードを指定します。

3. インストーラーを実行します。

13.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

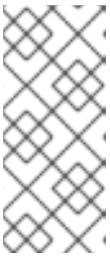
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

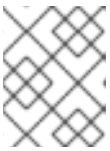
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

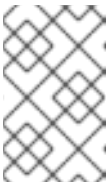
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

13.3.8. インストールプログラムの取得

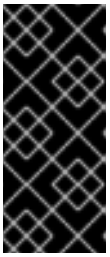
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

13.3.9. インストール設定ファイルの作成

Red Hat Virtualization (RHV) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

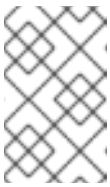
1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. インストールプログラムのプロンプトに対応します。

i. **SSH Public Key** では、パスワードなしのパブリックキー (例: **~/.ssh/id_rsa.pub**) を選択します。このキーは、新規 OpenShift Container Platform クラスターとの接続を認証します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターには、**ssh-agent** プロセスが使用する SSH キーを選択します。

ii. **Platform** には、**ovirt** を選択します。

iii. **Enter oVirt's API endpoint URL** に、この形式を使用して RHV API の URL を入力します。

```
https://<engine-fqdn>/ovirt-engine/api 1
```

1 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

iv. **Is the oVirt CA trusted locally?** には、CA 証明書がすでに設定されているため **Yes** を入力します。そうでない場合は、**No** と入力します。

v. **oVirt's CA bundle** には、前の質問で **Yes** を入力している場合には、**/etc/pki/ca-trust/source/anchors/ca.pem** の内容をコピーし、ここに貼り付けます。その後、**Enter** を 2 回押します。そうでない場合、つまり、前の質問で **No** と入力している場合は、この質問は表示されません。

vi. **oVirt engine username** には、この形式を使用して RHV 管理者のユーザー名およびプロファイルを入力します。

```
<username>@<profile> 1
```

1 **<username>** に、RHV 管理者のユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、**Profile** ドロップダウンリストで確認できます。

ユーザー名とプロファイルは以下のようになります。

```
ocpadmin@internal
```

vii. **oVirt engine password** に、RHV 管理者パスワードを入力します。

viii. **oVirt cluster** には、OpenShift Container Platform をインストールするためのクラスターを選択します。

ix. **oVirt storage domain** には、OpenShift Container Platform をインストールするためのストレージドメインを選択します。

x. **oVirt network** には、RHV Manager REST API へのアクセスのある仮想ネットワークを選択します。

xi. **Internal API Virtual IP** に、クラスターの REST API とは別の静的 IP アドレスを入力します。

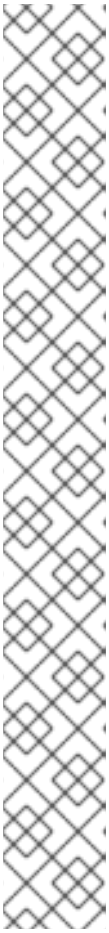
xii. **Ingress virtual IP** に、ワイルドカードアプリドメイン用に予約した静的 IP アドレスを入力します。

xiii. **Base Domain** に、OpenShift Container Platform クラスターのベースドメインを入力します。このクラスターが外部に公開される場合、これは DNS インフラストラクチャーが認識する有効なドメインである必要があります。たとえば、**virtlab.example.com** を入力します。

xiv. **Cluster Name** に、クラスターの名前を入力します。例: **my-cluster**OpenShift Container Platform REST API およびアプリケーションドメイン名向けに作成した外部登録/解決可能な DNS エントリーのクラスター名を使用します。インストールプログラムは、この名前を RHV 環境のクラスターにも指定します。

xv. **Pull secret** には、先にダウンロードした **pull-secret.txt** ファイルからプルシークレットをコピーし、ここに貼り付けます。Red Hat OpenShift Cluster Manager から同じプルシークレット のコピーを取得することもできます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。



注記

Manager に中間 CA 証明書がある場合は、証明書が **ovirt-config.yaml** ファイルおよび **install-config.yaml** ファイルに表示されることを確認します。表示されない場合は、以下のように追加します。

1. ~/**.ovirt**/**ovirt-config.yaml** ファイルの場合:

```
[ovirt_ca_bundle]: |
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA>
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    <INTERMEDIATE_CA>
    -----END CERTIFICATE-----
```

2. **install-config.yaml** ファイルの場合:

```
[additionalTrustBundle]: |
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA>
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    <INTERMEDIATE_CA>
    -----END CERTIFICATE-----
```

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

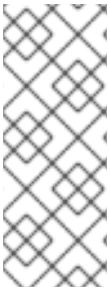
13.3.9.1. Red Hat Virtualization (RHV) のサンプル install-config.yaml ファイル

install-config.yaml ファイルのパラメーターおよびパラメーター値を変更して、インストールプログラムが作成する OpenShift Container Platform クラスターをカスタマイズできます。

以下は、RHV への OpenShift Container Platform のインストールに固有の例です。

install-config.yaml は、以下のコマンドを実行した際に指定した **<installation_directory>** にあります。

```
$ ./openshift-install create install-config --dir <installation_directory>
```



注記

- これらのサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。
- **install-config.yaml** ファイルを変更すると、クラスターに必要なリソースを増やすことができます。RHV 環境にそれらの追加リソースがあることを確認します。これらがない場合は、インストールまたはクラスターが失敗します。

デフォルトの install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: my-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 192.168.1.5
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
publish: External
pullSecret: '{"auths": ...}'
sshKey: ssh-ed12345 AAAA...
```

最小の install-config.yaml ファイルの例

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: test-cluster
```

```
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed12345 AAAA...
```

install-config.yaml ファイルのカスタムマシンプールの例

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 2
      memoryMB: 65536
      osDisk:
        sizeGB: 100
      vmType: server
  replicas: 3
compute:
- name: worker
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 4
      memoryMB: 65536
      osDisk:
        sizeGB: 200
      vmType: server
  replicas: 5
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

Enforcing 以外のアフィニティーグループの例
可能であれば、できるだけ多くのクラスターを使用するために、コントロールプレーンとワーカーを分散するために、enforcing 以外のアフィニティーグループを追加することをお勧めします。

```
platform:
  ovirt:
    affinityGroups:
    - description: AffinityGroup to place each compute machine on a separate host
      enforcing: true
      name: compute
      priority: 3
    - description: AffinityGroup to place each control plane machine on a separate host
      enforcing: true
      name: controlplane
      priority: 5
    - description: AffinityGroup to place worker nodes and control plane nodes on separate hosts
      enforcing: false
      name: openshift
      priority: 5
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    ovirt:
      affinityGroupsNames:
      - compute
      - openshift
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    ovirt:
      affinityGroupsNames:
      - controlplane
      - openshift
  replicas: 3
```

実稼働以外のラボセットアップのすべてのアフィニティーグループを削除する例
実稼働以外のラボセットアップでは、すべてのアフィニティーグループを削除して、OpenShift Container Platform クラスターをいくつかのホストに集中させる必要があります。

```
platform:
  ovirt:
    affinityGroups: []
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    ovirt:
      affinityGroupsNames: []
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
```

```
    ovirt:
      affinityGroupsNames: []
  replicas: 3
```

13.3.9.2. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

13.3.9.2.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表13.1 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト

パラメーター	説明	値
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

13.3.9.2.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表13.2 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

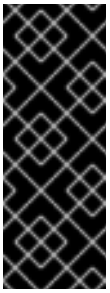
13.3.9.2.3. オプションの設定パラメーター

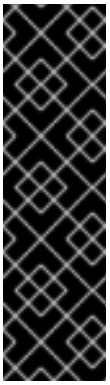
オプションのインストール設定パラメーターは、以下の表で説明されています。

表13.3 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

13.3.9.2.4. 追加の Red Hat Virtualization (RHV) 設定パラメーター

追加の RHV 設定パラメーターは以下の表で説明されています。

表13.4 クラスターの追加の Red Hat Virtualization (RHV) パラメーター

パラメーター	説明	値
platform.ovirt.ovirt_cluster_id	必須。仮想マシンが作成されるクラスター。	文字列。例: **68833f9f-e89c-4891-b768-e2ba0815b76b**
platform.ovirt.ovirt_storage_domain_id	必須。仮想マシンディスクが作成されるストレージドメイン ID。	文字列。例: **ed7b0f4e-0e96-492a-8fff-279213ee1468**

パラメーター	説明	値
platform.ovirt.ovirt_network_name	必須。仮想マシン NIC が作成されるネットワーク名。	文字列。例: **ocpcluster**
platform.ovirt.vnicProfileID	必須。仮想マシンネットワークインターフェイスの vNIC プロファイル ID。これは、クラスターネットワークに単一のプロファイルがある場合に示唆されます。	文字列。例: **3fa86930-0be5-4052-b667-b79f0a729692**
platform.ovirt.api_vip	必須。API 仮想 IP (VIP) に割り当てられるマシンネットワークの IP アドレス。このエンドポイントで OpenShift API にアクセスできます。	文字列。例: **10.46.8.230**
platform.ovirt.ingress_vip	必須。Ingress 仮想 IP (VIP) に割り当てられるマシンネットワークの IP アドレス。	文字列。例: **10.46.8.232**
platform.ovirt.affinityGroups	オプション。インストールプロセス中に作成するアフィニティーグループの一覧。	オブジェクトの一覧
platform.ovirt.affinityGroups.description	**platform.ovirt.affinityGroups** を含める場合は必須です。アフィニティーグループのの説明	文字列。例: **AffinityGroup for spreading each compute machine to a different host**
platform.ovirt.affinityGroups.enforcing	**platform.ovirt.affinityGroups** を含める場合は必須です。**true** に設定すると、十分なハードウェアノードが使用できない場合、RHV はマシンをプロビジョニングしません。**false** に設定すると、十分なハードウェアノードが使用できない場合でも、RHV はマシンをプロビジョニングするため、複数の仮想マシンが同じ物理マシンでホストされます。	文字列。例: **true**
platform.ovirt.affinityGroups.name	**platform.ovirt.affinityGroups** を含める場合は必須です。アフィニティーグループの名前。	文字列。例: **compute**

パラメーター	説明	値
platform.ovirt.affinityGroups.priority	**platform.ovirt.affinityGroups** を含める場合は必須です。**platform.ovirt.affinityGroups.enforcing = false** の場合に、アフィニティーグループに与えられる優先度。RHV は、優先順位の高い順にアフィニティーグループを適用します。この場合、小さい番号よりも大きい番号が優先されます。複数のアフィニティーグループの優先度が同じである場合、それらが適用される順序は保証されません。	整数例: **3**

13.3.9.2.5. マシンプールの追加 RHV パラメーター

マシンプールの追加の RHV 設定パラメーターは以下の表で説明されています。

表13.5 マシンプールの追加 RHV パラメーター

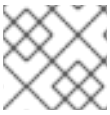
パラメーター	説明	値
<machine-pool>.platform.ovirt.cpu	オプション。仮想マシンの CPU を定義します。	オブジェクト
<machine-pool>.platform.ovirt.cpu.cores	**<machine-pool>.platform.ovirt.cpu** を使用する場合に必須です。コア数。仮想 CPU (vCPU) の合計はコア * ソケットです。	整数
<machine-pool>.platform.ovirt.cpu.sockets	**<machine-pool>.platform.ovirt.cpu** を使用する場合に必須です。コアあたりのソケット数。仮想 CPU (vCPU) の合計はコア * ソケットです。	整数
<machine-pool>.platform.ovirt.memoryMB	オプション。仮想マシンのメモリー (MiB 単位)。	整数

パラメーター	説明	値
<machine-pool>.platform.ovirt.instanceTypeID	オプション。**00000009-0009-0009-0009-0000000000f1** などのインスタンスタイプ UUID。これは **https://<engine-fqdn>/ovirt-engine/api/instancetypes** エンドポイントから取得できます。 **警告** **instance_type_id** フィールドは非推奨となり、今後のリリースで削除されます。	UUID の文字列
<machine-pool>.platform.ovirt.osDisk	オプション。仮想マシンの起動可能な初回の、および起動可能なディスクを定義します。	文字列
<machine-pool>.platform.ovirt.osDisk.sizeGB	**<machine-pool>.platform.ovirt.osDisk** を使用する場合に必須です。ディスクのサイズ (GiB 単位)。	数字

パラメーター	説明	値
<machine-pool>.platform.ovirt.vmType	オプション。**high-performance**、**server**、 または **desktop** などの仮想マシンワークロードタイプ。デフォルトでは、コントロールプレーンノードは **high performance** を使用し、ワーカーノードは **server** を使用します。詳細は、**仮想マシン管理ガイド**の 仮想マシンの一般設定に関する説明 および ハイパフォーマンス仮想マシン、テンプレート、およびプールの設定 を参照してください。  **注記** **high_performance** により、仮想マシンのパフォーマンスが向上しますが、制限があります。たとえば、グラフィカルコンソールを使用して仮想マシンにはアクセスできません。詳細は、**仮想マシン管理ガイド**の ハイパフォーマンス仮想マシン、テンプレート、およびプールの設定 を参照してください。	文字列

パラメーター	説明	値

<machine-pool>.platform.ovirt.affinityGroupsNames	オプション。仮想マシンに適用する必要があるアフィニティーグループ名の一覧。アフィニティーグループは RHV に存在するか、このトピックの**クラスターの追加 RHV パラメーター**で説明されているように、インストール中に作成する必要があります。このエントリーは空にすることができます。
	
	2 つのアフィニティーグループの例
	
	この例では、**compute** および **clusterWideNonEnforcing** という名前の 2 つのアフィニティーグループを定義します。
	
	`<machine-pool>:`
	`  platform:`
	`  ovirt:`
	`  affinityGroupNames:`
	`  - compute`
	`  - clusterWideNonEnforcing`
	
	この例では、アフィニティーグループを定義していません。
	
	`<machine-pool>:`
	`  platform:`
	`  ovirt:`
	`  affinityGroupNames: []`
		文字列

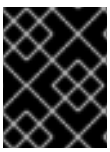


注記

<machine-pool> を **controlPlane** または **compute** に置き換えることができます。

13.3.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- インストーラーを実行するマシンから **ovirt-imageio** ポートを Manager へのポートを開放する。デフォルトでは、ポートは **54322** です。

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

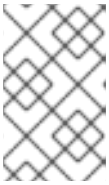
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

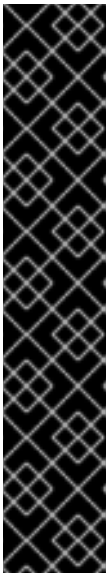
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

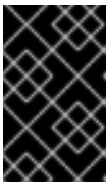
注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

重要

クラスターのインストールに必要な手順を完了している必要があります。残りの手順では、クラスターを検証し、インストールのトラブルシューティングを行う方法を説明します。

13.3.11. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

13.3.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

詳細は、Getting started with the OpenShift CLI を参照してください。

13.3.13. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

トラブルシューティング

インストールが失敗すると、インストールプログラムがタイムアウトし、エラーメッセージが表示されます。詳細は、インストールに関する問題のトラブルシューティング を参照してください。

13.3.14. RHV での OpenShift Container Platform Web コンソールへのアクセス

OpenShift Container Platform クラスターの初期化後に、OpenShift Container Platform Web コンソールにログインできます。

手順

1. オプション: Red Hat Virtualization (RHV) Administration Portal で、**Compute → Cluster** を開きます。
2. インストールプログラムが仮想マシンを作成することを確認します。
3. インストールプログラムが実行されているコマンドラインに戻ります。インストールプログラムが完了すると、OpenShift Container Platform Web コンソールにログインするためのユーザー名およびパスワードの一時パスワードが表示されます。
4. ブラウザーから OpenShift Container Platform の Web コンソールの URL を開きます。URL は以下の形式を使用します。

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

1 **<clustername>.<basedomain>** に、クラスター名およびベースドメインを指定します。

以下に例を示します。

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

13.3.15. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

13.3.16. Red Hat Virtualization (RHV) へのインストールに関するよくある問題のトラブルシューティング

以下に、一般的な問題およびそれらについて考えられる原因および解決策を記載します。

13.3.16.1. CPU 負荷が増大し、ノードが Not Ready 状態になる

- **現象**: CPU 負荷が大幅に増大し、ノードが **Not Ready** 状態に切り替わり始める。
- **原因**: 特にコントロールプレーンノード (別名マスターノード) の場合、ストレージドメインのレイテンシーが高すぎる可能性があります。
- **解決策**:
  Kubelet サービスを再起動して、ノードを再度 Ready 状態にします。

  ```
  $ systemctl restart kubelet
  ```

  OpenShift Container Platform メトリクスサービスを検査します。これは、etcd ディスクの同期期間などの有用なデータを収集し、これについて報告します。クラスターが機能している場合は、このデータを使用して、ストレージのレイテンシーまたはスループットが根本的な問題かどうかを判断します。その場合、レイテンシーが短く、スループットの高いストレージリソースの使用を検討してください。

  未加工メトリクスを取得するには、kubeadmin または cluster-admin 権限を持つユーザーで以下のコマンドを実行します。

  ```
  $ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics
  ```

  詳細は、Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x を参照してください。

13.3.16.2. OpenShift Container Platform クラスター API に接続できない

- **現象**: インストールプログラムは完了するが、OpenShift Container Platform クラスター API は利用できない。ブートストラップの仮想マシンは、ブートストラッププロセスの完了後も起動した状態になります。以下のコマンドを入力すると、応答がタイムアウトします。

  ```
  $ oc login -u kubeadmin -p *** <apiurl>
  ```

- **原因**: ブートストラップ仮想マシンがインストールプログラムによって削除されず、クラスターの API IP アドレスをリリースしない。
- **解決策**: **wait-for** サブコマンドを使用して、ブートストラッププロセスの完了時に通知を受信する。

```
$ ./openshift-install wait-for bootstrap-complete
```

ブートストラッププロセスが完了したら、ブートストラップ仮想マシンを削除します。

```
$ ./openshift-install destroy bootstrap
```

13.3.17. インストール後のタスク

OpenShift Container Platform クラスターの初期化後に、以下のタスクを実行できます。

- オプション: デプロイメント後に、OpenShift Container Platform で Machine Config Operator (MCO) を使用して SSH キーを追加するか、または置き換えます。
- オプション: **kubeadmin** ユーザーを削除します。代わりに、認証プロバイダーを使用して cluster-admin 権限を持つユーザーを作成します。

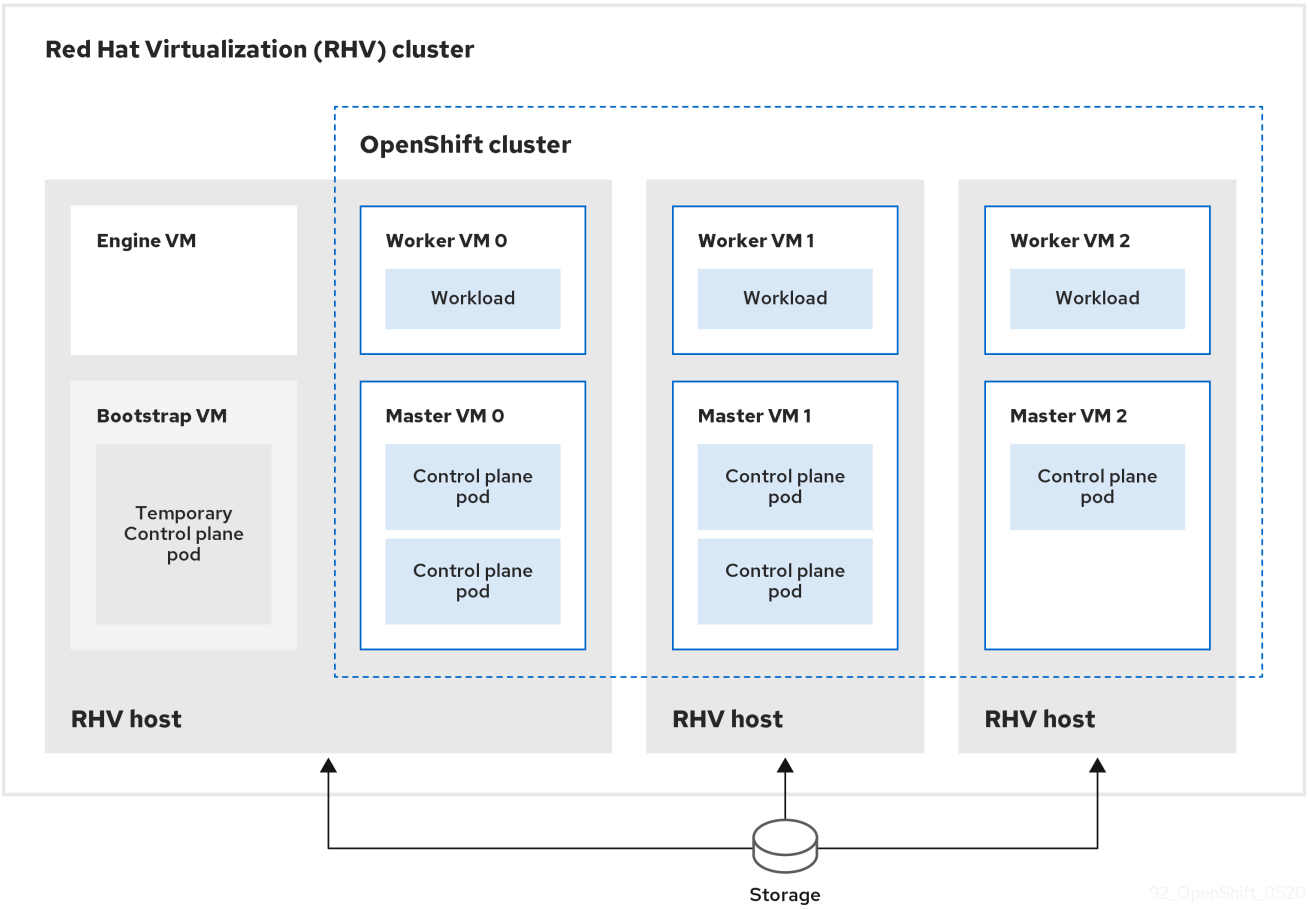
13.3.18. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

13.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した RHV へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、Red Hat Virtualization (RHV) および提供する他のインフラストラクチャーにカスタマイズされた OpenShift Container Platform クラスターをインストールできます。OpenShift Container Platform ドキュメントでは、**ユーザーによってプロビジョニングされるインフラストラクチャー** という用語を使用して、このインフラストラクチャータイプに言及しています。

以下の図は、RHV クラスターで実行される可能性のある OpenShift Container Platform クラスターの例を示しています。



RHV ホストは、コントロールプレーンとコンピュート Pod の両方が含まれる仮想マシンを実行します。ホストのいずれかが Manage 仮想マシンと、一時的なコントロールプレーン Pod を含むブートストラップ仮想マシンも実行します。

13.4.1. 前提条件

OpenShift Container Platform クラスターを RHV 環境にインストールするには、以下の要件を満たしている必要があります。

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- Support Matrix for OpenShift Container Platform on Red Hat Virtualization (RHV) に記載のあるサポートされるバージョンの組み合わせを使用できる。

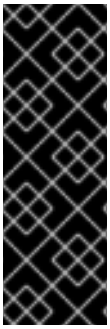
13.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

13.4.3. RHV 環境の要件

OpenShift Container Platform バージョン 4.8 クラスターをインストールし、実行するには、RHV 環境が以下の要件を満たしている必要があります。

これらの要件を満たさないと、インストールまたはプロセスが失敗する可能性があります。さらに、これらの要件を満たしていないと、OpenShift Container Platform クラスターはインストールしてから数日または数週間後に失敗する可能性があります。

CPU、メモリー、ストレージリソースについての以下の要件は、インストールプログラムが作成する仮想マシンのデフォルト数で乗算した **デフォルト** 値に基づいています。これらのリソースは、RHV 環境が OpenShift Container Platform 以外の操作に使用するものに **加え**、利用可能でなければなりません。

デフォルトでは、インストールプログラムは 7 つの仮想マシンをインストールプロセスで作成します。まず、ブートストラップ仮想マシンを作成し、OpenShift Container Platform クラスターの残りの部分を作成する間に一時サービスとコントロールプレーンを提供します。インストールプログラムがクラスターの作成を終了すると、ブートストラップマシンが削除され、そのリソースが解放されます。

RHV 環境の仮想マシン数を増やす場合は、リソースを適宜増やす必要があります。

要件

- RHV のバージョンは 4.4 である。
- RHV 環境に **Up** 状態のデータセンターが 1 つあること。
- RHV データセンターに RHV クラスターが含まれていること。
- RHV クラスターに OpenShift Container Platform クラスター専用の以下のリソースがあること。
  - 最小 28 vCPU: インストール時に作成される 7 仮想マシンのそれぞれに 4 vCPU。
  - 以下を含む 112 GiB 以上の RAM。
    - 一時的なコントロールプレーンを提供するブートストラップマシン用に 16 GiB 以上。
    - コントロールプレーンを提供する 3 つのコントロールプレーンマシンのそれぞれに 16 GiB 以上。
    - アプリケーションワークロードを実行する 3 つのコンピュートマシンのそれぞれに 16 GiB 以上。

- RHV ストレージドメインは、これらの etcd バックエンドのパフォーマンス要件 を満たす必要があります。
- 実稼働環境では、各仮想マシンに 120 GiB 以上が必要です。そのため、ストレージドメインはデフォルトの OpenShift Container Platform クラスターに 840 GiB 以上を提供する必要があります。リソースに制約のある環境または非実稼働環境では、各仮想マシンに 32 GiB 以上を指定する必要があるため、ストレージドメインにはデフォルトの OpenShift Container Platform クラスター用に 230 GiB 以上が必要になります。
- インストールおよび更新中に Red Hat Ecosystem Catalog からイメージをダウンロードするには、RHV クラスターがインターネット接続にアクセスできる必要があります。また、サブスクリプションおよびエンタイトルメントプロセスを単純化するために Telemetry サービスにもインターネット接続が必要です。
- RHV クラスターには、RHV Manager の REST API にアクセスできる仮想ネットワークが必要です。インストーラーが作成する仮想マシンが DHCP を使用して IP アドレスを取得するため、DHCP がこのネットワークで有効にされていることを確認します。
- ターゲット RHV クラスターに OpenShift Container Platform クラスターをインストールし、管理するための以下の最小限の権限を持つユーザーアカウントおよびグループ。
  - **DiskOperator**
  - **DiskCreator**
  - **UserTemplateBasedVm**
  - **TemplateOwner**
  - **TemplateCreator**
  - ターゲットクラスターの **ClusterAdmin**

警告

最小権限の原則を適用します。インストールプロセスで RHV で **SuperUser** 権限を持つ管理者アカウントを使用することを避けます。インストールプログラムは、ユーザーが指定する認証情報を、危険にさらされる可能性のある一時的な **ovirt-config.yaml** ファイルに保存します。

13.4.4. RHV 環境の要件の確認

RHV 環境が OpenShift Container Platform クラスターをインストールし、実行するための要件を満たしていることを確認します。これらの要件を満たさないと、エラーが発生する可能性があります。

重要

これらの要件は、インストールプログラムがコントロールプレーンおよびコンピュートマシンの作成に使用するデフォルトのリソースに基づいています。これらのリソースには、vCPU、メモリー、およびストレージが含まれます。これらのリソースを変更するか、または OpenShift Container Platform マシンの数を増やす場合は、これらの要件を適宜調整します。

手順

1. RHV のバージョンが OpenShift Container Platform バージョン 4.8 のインストールをサポートすることを確認します。
   a. RHV Administration Portal の右上にある ? ヘルプアイコンをクリックし、**About** を選択します。
   b. 開かれるウィンドウで、**RHV ソフトウェアのバージョン** をメモします。
   c. RHV のバージョンが 4.4 であることを確認します。サポートされるバージョンの組み合わせについての詳細は、Support Matrix for OpenShift Container Platform on RHV を参照してください。
2. データセンター、クラスター、およびストレージを検査します。
   a. RHV 管理ポータルで、**Compute → Data Centers** をクリックします。
   b. OpenShift Container Platform をインストールする予定のデータセンターにアクセスできることを確認します。
   c. そのデータセンターの名前をクリックします。
   d. データセンターの詳細の **Storage** タブで、OpenShift Container Platform をインストールする予定のストレージドメインが **Active** であることを確認します。
   e. 後で使用できるように **ドメイン名** を記録します。
   f. **空き領域** に 230 GiB 以上あることを確認します。
   g. ストレージドメインが これらの etcd バックエンドのパフォーマンス要件 を満たしていることを確認します。これは、fio パフォーマンスベンチマークツールを使用して測定できます。
   h. データセンターの詳細で、**Clusters** タブをクリックします。
   i. OpenShift Container Platform をインストールする予定の RHV クラスターを見つけます。後で使用できるようにクラスター名を記録します。
3. RHV ホストリソースを確認します。
   a. RHV 管理ポータルで、**Compute > Clusters** をクリックします。
   b. OpenShift Container Platform をインストールする予定のクラスターをクリックします。
   c. クラスターの詳細で、**Hosts** タブをクリックします。
   d. ホストを検査し、それらに OpenShift Container Platform クラスター **専用** として利用可能な **論理 CPU コア** の合計が 28 つ以上であることを確認します。

e. 後で使用できるように、利用可能な **論理 CPU コア** の数を記録します。

f. これらの CPU コアが分散され、インストール時に作成された 7 つの仮想マシンのそれぞれに 4 つのコアを持たせることができることを確認します。

g. ホストには、以下の OpenShift Container Platform マシンのそれぞれの要件を満たすように **新規仮想マシンをスケジュールするための最大空きメモリー** として 112 GiB があることを確認します。

- ブートストラップマシンに 16 GiB が必要です。
- 3 つのコントロールプレーンマシンのそれぞれに 16 GiB が必要です。
- 3 つのコンピュートマシンのそれぞれに 16 GiB が必要です。

h. 後で使用できるように **新規仮想マシンをスケジュールするための最大空きメモリー** の量を記録します。

4. OpenShift Container Platform をインストールするための仮想ネットワークが RHV Manager の REST API にアクセスできることを確認します。このネットワーク上の仮想マシンから、RHV Manager の REST API に到達するために curl を使用します。

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** については、RHV で OpenShift Container Platform クラスターを作成および管理する権限を持つ RHV アカウントのユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。**<password>** に、そのユーザー名のパスワードを指定します。

2 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

13.4.5. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

ファイアウォール

クラスターが必要なサイトにアクセスできるようにファイアウォールを設定します。

以下も参照してください。

- Red Hat Virtualization Manager ファイアウォールの要件
- ホストのファイアウォール要件

ロードバランサー

レイヤー 4 のロードバランサーを 1 つまたは 2 つ (推奨) 設定します。

- コントロールプレーンおよびブートストラップマシンのポート **6443** および **22623** に対して負荷分散を行います。ポート **6443** は Kubernetes API サーバーへのアクセスを提供し、内外で到達可能である必要があります。ポート **22623** はクラスター内のノードからアクセスできる必要があります。
- Ingress ルーターを実行するマシン (通常はデフォルト設定のコンピュートノード) 向けに、ポート **443** および **80** に対する負荷分散を行います。いずれのポートもクラスター内外でアクセスできる必要があります。

DNS

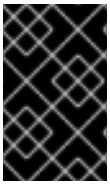
インフラストラクチャーで提供される DNS を設定して、主要なコンポーネントとサービスの正しい解決を許可します。1 つのロードバランサーのみを使用する場合、これらの DNS レコードは同じ IP アドレスを参照できます。

- **api.<cluster_name>.<base_domain>** (内部および外部解決) と、コントロールプレーンマシンのロードバランサーを参照する **api-int.<cluster_name>.<base_domain>** (内部解決) の DNS レコードを作成します。
- Ingress ルーターのロードバランサーを参照する ***.apps.<cluster_name>.<base_domain>** の DNS レコードを作成します。たとえば、コンピュートマシンのポート **443** および **80** などが含まれます。

13.4.5.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表13.6 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト

プロトコル	ポート	説明
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表13.7 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表13.8 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

13.4.6. インストールマシンの設定

バイナリー **openshift-install** インストールプログラムおよび Ansible スクリプトを実行するには、Manager 上の RHV 環境および REST API にネットワークでアクセスできるように、RHV Manager または Red Hat Enterprise Linux (RHEL) を設定します。

手順

1. Python3 および Ansible を更新またはインストールします。以下に例を示します。

```
# dnf update python3 ansible
```

2. **python3-ovirt-engine-sdk4** パッケージをインストール して、Python Software Development Kit を取得します。

3. **ovirt.image-template** Ansible ロールをインストールします。RHV Manager およびその他の Red Hat Enterprise Linux (RHEL) マシンでは、このロールは **ovirt-ansible-image-template** パッケージとして提供されます。たとえば、 以下を入力します。

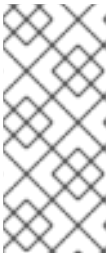
```
# dnf install ovirt-ansible-image-template
```

4. **ovirt.vm-infra** Ansible ロールをインストールします。RHV Manager およびその他の RHEL マシンでは、このロールは **ovirt-ansible-vm-infra** パッケージとして提供されます。

```
# dnf install ovirt-ansible-vm-infra
```

5. 環境変数を作成し、その環境変数に絶対パスまたは相対パスを割り当てます。たとえば、 以下を入力します。

```
$ export ASSETS_DIR=./wrk
```



注記

インストールプログラムはこの変数を使用して、重要なインストール関連のファイルを保存するディレクトリーを作成します。その後、インストールプロセスはこの変数を再利用して、これらのアセットファイルを見つけます。このアセットディレクトリーを削除しないでください。これは、クラスターのアンインストールに必要になります。

13.4.7. OpenShift Container Platform OpenStack クラスターの RHV への非セキュアモードでのインストール

デフォルトで、インストーラーは CA 証明書を作成し、確認を求めるプロンプトを出し、インストール時に使用する証明書を保存します。これは、手動で作成したりインストールしたりする必要はありません。

推奨されていませんが、OpenShift Container Platform を RHV に **非セキュアモード** でインストールして、この機能を上書きし、証明書の検証なしに OpenShift Container Platform をインストールすることができます。

警告

非セキュア モードでのインストールは推奨されていません。これにより、攻撃者が中間者 (Man-in-the-Middle) 攻撃を実行し、ネットワーク上の機密の認証情報を取得できる可能性が生じるためです。

手順

1. **~/.ovirt/ovirt-config.yaml** という名前のファイルを作成します。
2. 以下の内容を **ovirt-config.yaml** に追加します。

```
ovirt_url: https://ovirt.example.com/ovirt-engine/api (1)
ovirt_fqdn: ovirt.example.com (2)
ovirt_pem_url: ""
ovirt_username: ocpadmin@internal
ovirt_password: super-secret-password (3)
ovirt_insecure: true
```

(1) oVirt エンジンのホスト名またはアドレスを指定します。

(2) oVirt エンジンの完全修飾ドメイン名を指定します。

(3) oVirt エンジンの管理者パスワードを指定します。

3. インストーラーを実行します。

13.4.8. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

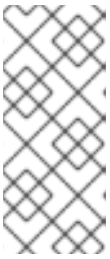
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

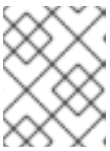
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

13.4.9. インストールプログラムの取得

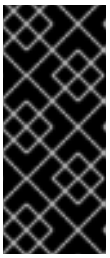
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

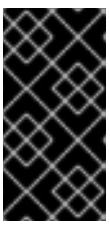
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

13.4.10. Ansible Playbook のダウンロード

RHV に OpenShift Container Platform バージョン 4.8 をインストールするために Ansible Playbook をダウンロードします。

手順

- インストールマシンで、以下のコマンドを実行します。

```
$ mkdir playbooks
```

```
$ cd playbooks
```

```
$ curl -s -L -X GET https://api.github.com/repos/openshift/installer/contents/upi/ovirt?
ref=release-4.8 |
grep 'download_url.*\.yml' |
awk '{ print $2 }' | sed -r 's/("|",)//g' |
xargs -n 1 curl -O
```

次のステップ

- これらの Ansible Playbook をダウンロードしたら、インストールプログラムを実行してインストール設定ファイルを作成する前に、アセットディレクトリーの環境変数を作成し、**inventory.yml** ファイルをカスタマイズする必要もあります。

13.4.11. inventory.yml ファイル

inventory.yml ファイルを使用して、インストールする OpenShift Container Platform クラスターの各種の要素を定義し、作成します。これには、Red Hat Enterprise Linux CoreOS(RHCOS) イメージ、仮想マシンテンプレート、ブートストラップマシン、コントロールプレーンノード、ワーカーノードなどの要素が含まれます。また、**inventory.yml** を使用してクラスターを破棄します。

以下の **inventory.yml** の例は、パラメーターとそれらのデフォルト値を示しています。これらのデフォルト値の量と数は、RHV 環境で実稼働用の OpenShift Container Platform クラスターを実行するための要件を満たしています。

inventory.yml ファイルの例

```
---
all:
  vars:

    ovirt_cluster: "Default"
```

```
    ocp:
      assets_dir: "{{ lookup('env', 'ASSETS_DIR') }}"
      ovirt_config_path: "{{ lookup('env', 'HOME') }}/.ovirt/ovirt-config.yaml"

    # ---
    # {op-system} section
    # ---
    rhcos:
      image_url: "https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.8/latest/rhcos-
openstack.x86_64.qcow2.gz"
      local_cmp_image_path: "/tmp/rhcos.qcow2.gz"
      local_image_path: "/tmp/rhcos.qcow2"

    # ---
    # Profiles section
    # ---
    control_plane:
      cluster: "{{ ovirt_cluster }}"
      memory: 16GiB
      sockets: 4
      cores: 1
      template: rhcos_tpl
      operating_system: "rhcos_x64"
      type: high_performance
      graphical_console:
        headless_mode: false
        protocol:
        - spice
        - vnc
      disks:
      - size: 120GiB
        name: os
        interface: virtio_scsi
        storage_domain: depot_nvme
      nics:
      - name: nic1
        network: lab
        profile: lab

    compute:
      cluster: "{{ ovirt_cluster }}"
      memory: 16GiB
      sockets: 4
      cores: 1
      template: worker_rhcos_tpl
      operating_system: "rhcos_x64"
      type: high_performance
      graphical_console:
        headless_mode: false
        protocol:
        - spice
        - vnc
      disks:
      - size: 120GiB
        name: os
        interface: virtio_scsi
```

```
    storage_domain: depot_nvme
  nics:
  - name: nic1
    network: lab
    profile: lab

# ---
# Virtual machines section
# ---
vms:
- name: "{{ metadata.infraID }}-bootstrap"
  ocp_type: bootstrap
  profile: "{{ control_plane }}"
  type: server
- name: "{{ metadata.infraID }}-master0"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master1"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master2"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-worker0"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker1"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker2"
  ocp_type: worker
  profile: "{{ compute }}"
```

重要

Enter から始まる説明のあるパラメーターの値を入力します。 それ以外の場合は、デフォルト値を使用するか、またはこえを新しい値に置き換えることができます。

General セクション

- **ovirt_cluster**: OpenShift Container Platform クラスターをインストールする既存の RHV クラスターの名前を入力します。
- **ocp.assets_dir**: **openshift-install** インストールプログラムが生成するファイルを保存するために作成するディレクトリーのパス。
- **ocp.ovirt_config_path**: インストールプログラムが生成する **ovirt-config.yaml** ファイルのパス (**./wrk/install-config.yaml** など)。このファイルには、Manager の REST API との対話に必要な認証情報が含まれます。

Red Hat Enterprise Linux CoreOS (RHCOS) セクション

- **image_url**: ダウンロード用に指定した RHCOS イメージの URL を入力します。

- **local_cmp_image_path**: 圧縮された RHCOS イメージのローカルダウンロードディレクトリーのパス。
- **local_image_path**: 展開した RHCOS イメージのローカルディレクトリーのパス。

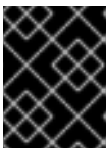
Profiles セクション

このセクションは、2 つのプロファイルで設定されます。

- **control_plane**: ブートストラップおよびコントロールプレーンノードのプロファイル。
- **compute**: コンピュートプレーン内のワーカーノードのプロファイル。

これらのプロファイルには以下のパラメーターが含まれます。パラメーターのデフォルト値は、実稼働クラスターを実行するために必要な最小要件を満たします。これらの値は、ワークロードの要件に応じて増減したり、カスタマイズしたりできます。

- **cluster**: 値は、General セクションの **ovirt_cluster** からクラスター名を取得します。
- **memory**: 仮想マシンに必要なメモリーの量 (GB)。
- **sockets**: 仮想マシンのソケット数。
- **cores**: 仮想マシンのコア数。
- **template**: 仮想マシンテンプレートの名前。複数のクラスターをインストールする計画があり、これらのクラスターが異なる仕様が含まれるテンプレートを使用する場合には、テンプレート名の先頭にクラスターの ID を付けます。
- **operating_system**: 仮想マシンのゲストオペレーティングシステムのタイプ。oVirt/RHV バージョン 4.4 では、**Ignition script** の値を仮想マシンに渡すことができるようにするために、この値を **rhcos_x64** にする必要があります。
- **type**: 仮想マシンのタイプとして **server** を入力します。



重要

type パラメーターの値を **high_performance** から **server** に変更する必要があります。

- **disks**: ディスクの仕様。**control_plane** と **compute** ノードには、異なるストレージドメインを設定できます。
- **size**: ディスクの最小サイズ。
- **name**: RHV のターゲットクラスターに接続されたディスクの名前を入力します。
- **interface**: 指定したディスクのインターフェイスタイプを入力します。
- **storage_domain**: 指定したディスクのストレージドメインを入力します。
- **nics**: 仮想マシンが使用する **name** および **network** を入力します。仮想ネットワークインターフェイスプロファイルを指定することもできます。デフォルトでは、NIC は oVirt/RHV MAC プールから MAC アドレスを取得します。

仮想マシンセクション

この最後のセクション **vms** は、クラスターで作成およびデプロイする予定の仮想マシンを定義します。デフォルトで、実稼働環境用の最小数のコントロールプレーンおよびワーカーノードが提供されます。

vms には 3 つの必須要素が含まれます。

- **name**: 仮想マシンの名前。この場合、**metadata.infraID** は、仮想マシン名の先頭に **metadata.yml** ファイルのインフラストラクチャー ID を付けます。
- **ocp_type**: OpenShift Container Platform クラスター内の仮想マシンのロール。使用できる値は **bootstrap**、**master**、**worker** です。
- **profile**: それぞれの仮想マシンが仕様を継承するプロファイルの名前。この例で使用可能な値は **control_plane** または **compute** です。
  仮想マシンがプロファイルから継承する値を上書きできます。これを実行するには、**inventory.yml** の仮想マシンに profile 属性の名前を追加し、これに上書きする値を割り当てます。この例を確認するには、直前の **inventory.yml** の例の **name: "{{ metadata.infraID }}-bootstrap"** 仮想マシンを検査します。これには値が **server** の **type** 属性があり、この仮想マシンがそれ以外の場合に **control_plane** プロファイルから継承する **type** 属性の値を上書きします。

メタデータ変数

仮想マシンの場合、**metadata.infraID** は、仮想マシンの名前の先頭に、Ignition ファイルのビルド時に作成する **metadata.json** ファイルのインフラストラクチャー ID を付けます。

Playbook は以下のコードを使用して、**ocp.assets_dir** にある特定のファイルから **infraID** を読み取ります。

```
---
- name: include metadata.json vars
  include_vars:
    file: "{{ ocp.assets_dir }}/metadata.json"
    name: metadata

  ...
```

13.4.12. RHCOS イメージ設定の指定

inventory.yml ファイルの Red Hat Enterprise Linux CoreOS (RHCOS) イメージ設定を更新します。後にこのファイルを Playbook のいずれかとして実行すると、圧縮された Red Hat Enterprise Linux CoreOS (RHCOS) イメージが **image_url** URL から **local_cmp_image_path** ディレクトリーにダウンロードされます。次に Playbook はイメージを **local_image_path** ディレクトリーに展開し、これを使用して oVirt/RHV テンプレートを作成します。

手順

1. インストールする OpenShift Container Platform バージョンの RHCOS イメージダウンロードページを見つけます (例: /pub/openshift-v4/dependencies/rhcos/latest/latest のインデックス)。
2. そのダウンロードページから、**https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.8/latest/rhcos-openstack.x86_64.qcow2.gz** などの OpenStack **qcow2** イメージの URL をコピーします。

3. 先のステップでダウンロードした **inventory.yml** Playbook を編集します。この中で、URL を **image_url** の値として貼り付けます。以下に例を示します。

```
rhcos:
  "https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.8/latest/rhcos-openstack.x86_64.qcow2.gz"
```

13.4.13. インストール設定ファイルの作成

インストールプログラム **openshift-install** を実行し、先に指定または収集した情報でプロンプトに応答し、インストール設定ファイルを作成します。

プロンプトに応答すると、インストールプログラムは、以前に指定したアセットディレクトリーの **install-config.yaml** ファイルの初期バージョンを作成します (例: **./wrk/install-config.yaml**)。

インストールプログラムは、Manager に到達して REST API を使用するために必要なすべての接続パラメーターが含まれる **$HOME/.ovirt/ovirt-config.yaml** ファイルも作成します。

注: インストールプロセスでは、**Internal API virtual IP** および **Ingress virtual IP** などの一部のパラメーターに指定する値を使用しません。それらの値はインフラストラクチャー DNS にすでに設定されているためです。

また、**oVirt cluster**、 **oVirt storage**、および **oVirt network** などの値のような **inventory.yml** のパラメーターに指定する値を使用します。また、スクリプトを使用して **install-config.yaml** の同じ値を削除するか、またはこれを前述の **virtual IPs** に置き換えます。

手順

1. インストールプログラムを実行します。

```
$ openshift-install create install-config --dir $ASSETS_DIR
```

2. インストールプログラムのプロンプトに応答し、システムに関する情報を提供します。

出力例

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
```

```
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

Internal API virtual IP および **Ingress virtual IP** について、DNS サービスの設定時に指定した IP アドレスを指定します。

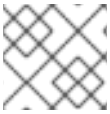
さらに、**oVirt cluster** および **Base Domain** プロンプトに対して入力する値は REST API および作成するアプリケーションの URL の一部を設定します (例: **https://api.ocp4.example.org:6443/** and **https://console-openshift-console.apps.ocp4.example.org**)。

Red Hat OpenShift Cluster Manager からプルシークレット を取得できます。

13.4.14. install-config.yaml のカスタマイズ

ここでは、3 つの python スクリプトを使用して、インストールプログラムのデフォルト動作の一部を上書きします。

- デフォルトでは、インストールプログラムはマシン API を使用してノードを作成します。このデフォルトの動作を上書きするには、コンピュートノードの数をゼロ (0) レプリカに設定します。後に Ansible Playbook を使用してコンピュートノードを作成します。
- デフォルトでは、インストールプログラムはノードのマシンネットワークの IP 範囲を設定します。このデフォルトの動作を上書きするには、インフラストラクチャーに一致するように IP 範囲を設定します。
- デフォルトでは、インストールプログラムはプラットフォームを **ovirt** に設定します。ただし、ユーザーによってプロビジョニングされるインフラストラクチャーにクラスターをインストールすることは、ベアメタルにクラスターをインストールすることに似ています。したがって、ovirt プラットフォームセクションを **install-config.yaml** から削除し、プラットフォームを **none** に変更します。代わりに、**inventory.yml** を使用して、必要な設定をすべて指定します。



注記

これらのスニペットは Python 3 および Python 2 で動作します。

手順

1. コンピュートノードの数をゼロ (0) レプリカに設定します。

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["compute"][0]["replicas"] = 0
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

2. マシンネットワークの IP 範囲を設定します。たとえば、範囲を **172.16.0.0/16** に設定するには、以下を実行します。

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["networking"]["machineNetwork"][0]["cidr"] = "172.16.0.0/16"
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

3. **ovirt** セクションを削除し、プラットフォームを **none** に変更します。

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
platform = conf["platform"]
del platform["ovirt"]
platform["none"] = {}
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールするのと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

13.4.15. マニフェストファイルの生成

インストールプログラムを使用して、アセットディレクトリーにマニフェストファイルのセットを生成します。

マニフェストファイルを生成するコマンドにより、**install-config.yaml** ファイルを使用する前に警告メッセージが表示されます。

install-config.yaml ファイルを再利用する予定の場合には、マニフェストファイルを生成する前にバックアップしてからバックアップコピーを作成してください。

手順

1. オプション: **install-config.yaml** ファイルのバックアップコピーを作成します。

```
$ cp install-config.yaml install-config.yaml.backup
```

2. アセットディレクトリーにマニフェストのセットを生成します。

```
$ openshift-install create manifests --dir $ASSETS_DIR
```

このコマンドにより、以下の情報が表示されます。

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

このコマンドにより、以下のマニフェストファイルが生成されます。

出力例

```
$ tree
.
└── wrk
    ├── manifests
    │   ├── 04-openshift-machine-config-operator.yaml
    │   ├── cluster-config.yaml
    │   ├── cluster-dns-02-config.yml
    │   ├── cluster-infrastructure-02-config.yml
    │   ├── cluster-ingress-02-config.yml
    │   ├── cluster-network-01-crd.yml
    │   ├── cluster-network-02-config.yml
    │   ├── cluster-proxy-01-config.yaml
    │   ├── cluster-scheduler-02-config.yml
    │   ├── cvo-overrides.yaml
    │   ├── etcd-ca-bundle-configmap.yaml
    │   ├── etcd-client-secret.yaml
    │   ├── etcd-host-service-endpoints.yaml
    │   ├── etcd-host-service.yaml
    │   ├── etcd-metric-client-secret.yaml
    │   ├── etcd-metric-serving-ca-configmap.yaml
    │   ├── etcd-metric-signer-secret.yaml
    │   ├── etcd-namespace.yaml
    │   ├── etcd-service.yaml
    │   ├── etcd-serving-ca-configmap.yaml
    │   ├── etcd-signer-secret.yaml
    │   ├── kube-cloud-config.yaml
    │   ├── kube-system-configmap-root-ca.yaml
    │   ├── machine-config-server-tls-secret.yaml
    │   └── openshift-config-secret-pull-secret.yaml
    └── openshift
        ├── 99_kubeadmin-password-secret.yaml
        ├── 99_openshift-cluster-api_master-user-data-secret.yaml
        ├── 99_openshift-cluster-api_worker-user-data-secret.yaml
        ├── 99_openshift-machineconfig_99-master-ssh.yaml
        ├── 99_openshift-machineconfig_99-worker-ssh.yaml
        └── openshift-install-manifests.yaml
```

次のステップ

- コントロールプレーンノードをスケジュール対象外にします。

13.4.16. コントロールプレーンノードのスケジュール対象外の設定

コントロールプレーンマシンを手動で作成し、デプロイしているので、コントロールプレーンノードをスケジュール対象外にするようにマニフェストファイルを設定する必要があります。

手順

1. コントロールプレーンノードをスケジュール対象外にするには、以下を入力します。

```
$ python3 -c 'import os, yaml
path = "%s/manifests/cluster-scheduler-02-config.yml" % os.environ["ASSETS_DIR"]
data = yaml.safe_load(open(path))
data["spec"]["mastersSchedulable"] = False
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

13.4.17. Ignition ファイルのビルド

生成および変更したマニフェストファイルから Ignition ファイルを作成するには、インストールプログラムを実行します。このアクションにより、Ignition ファイルをフェッチし、ノードを作成するために必要な設定を実行する Red Hat Enterprise Linux CoreOS (RHCOS) マシン **initramfs** が作成されます。

Ignition ファイルのほかに、インストールプログラムは以下を生成します。

- **oc** および **kubectl** ユーティリティーを使用してクラスターに接続するための管理者認証情報が含まれる **auth** ディレクトリー。
- OpenShift Container Platform クラスター名、クラスター ID、および現行インストールのインフラストラクチャー ID などの情報を含む **metadata.json** ファイル。

このインストールプロセスの Ansible Playbook は、**infraID** の値を、作成する仮想マシンの接頭辞として使用します。これにより、同じ oVirt/RHV クラスターに複数のインストールがある場合の命名の競合が回避されます。

注記

Ignition 設定ファイルの証明書は 24 時間後に有効期限が切れます。最初の証明書のローテーションが終了するように、クラスターのインストールを完了し、クラスターを動作が低下していない状態で 24 時間実行し続ける必要があります。

手順

1. Ignition ファイルをビルドするには、以下を入力します。

```
$ openshift-install create ignition-configs --dir $ASSETS_DIR
```

出力例

```
$ tree
.
└── wrk
    ├── auth
    │   ├── kubeadmin-password
    │   └── kubeconfig
    ├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

13.4.18. テンプレートおよび仮想マシンの作成

inventory.yml の変数を確認した後に、最初の Ansible プロビジョニング Playbook **create-templates-and-vms.yml** を実行します。

この Playbook は、**$HOME/.ovirt/ovirt-config.yaml** から RHV Manager の接続パラメーターを使用し、アセットディレクトリーで **metadata.json** を読み取ります。

ローカルの Red Hat Enterprise Linux CoreOS (RHCOS) イメージが存在しない場合、Playbook は **inventory.yml** の **image_url** に指定した URL からダウンロードします。これはイメージを展開し、これを RHV にアップロードしてテンプレートを作成します。

Playbook は、**inventory.yml** ファイルの **control_plane** と **compute** プロファイルに基づいてテンプレートを作成します。これらのプロファイルの名前が異なる場合、2 つのテンプレートが作成されます。

Playbook が完了すると、作成される仮想マシンは停止します。他のインフラストラクチャー要素の設定に役立つ情報を取得できます。たとえば、仮想マシンの MAC アドレスを取得して、仮想マシンに永続的な IP アドレスを割り当てるように DHCP を設定できます。

手順

1. **inventory.yml** の **control_plane** および **compute** 変数で、**type: high_performance** の 両方のインスタンスを **type: server** に変更します。
2. オプション: 同じクラスターに複数のインストールを実行する予定の場合には、OpenShift Container Platform インストールごとに異なるテンプレートを作成します。**inventory.yml** ファイルで、**template** の値の先頭に **infraID** を付けます。以下に例を示します。

```
control_plane:
  cluster: "{{ ovirt_cluster }}"
  memory: 16GiB
  sockets: 4
  cores: 1
  template: "{{ metadata.infraID }}-rhcos_tpl"
  operating_system: "rhcos_x64"
  ...
```

3. テンプレートおよび仮想マシンを作成します。

```
$ ansible-playbook -i inventory.yml create-templates-and-vms.yml
```

13.4.19. ブートストラップマシンの作成

bootstrap.yml Playbook を実行してブートストラップマシンを作成します。この Playbook はブートストラップ仮想マシンを起動し、これをアセットディレクトリーから **bootstrap.ign** Ignition ファイルに渡します。ブートストラップノードは、Ignition ファイルをコントロールプレーンノードに送信できるように設定します。

ブートストラッププロセスをモニターするには、RHV 管理ポータルでコンソールを使用するか、SSH を使用して仮想マシンに接続します。

手順

1. ブートストラップマシンを作成します。

```
$ ansible-playbook -i inventory.yml bootstrap.yml
```

2. 管理ポータルまたは SSH のコンソールを使用してブートストラップマシンに接続します。**<bootstrap_ip>** をブートストラップノードの IP アドレスに置き換えます。SSH を使用するには、以下を入力します。

```
$ ssh core@<boostrap.ip>
```

3. ブートストラップノードからリリースイメージサービスについての **bootkube.service** journald ユニットログを収集します。

```
[core@ocp4-lk6b4-bootstrap ~]$ journalctl -b -f -u release-image.service -u bootkube.service
```

注記

ブートストラップノードの **bootkube.service** ログは、etcd の **connection refused** エラーを出力し、ブートストラップサーバーがコントロールプレーンノード (別名マスターノード) の etcd に接続できないことを示します。etcd が各コントロールプレーンノードで起動し、ノードがクラスターに参加した後には、エラーは発生しなくなるはずです。

13.4.20. コントロールプレーンノードの作成

masters.yml Playbook を実行してコントロールプレーンノードを作成します。この Playbook は **master.ign** Ignition ファイルをそれぞれの仮想マシンに渡します。Ignition ファイルには、**https://api-int.ocp4.example.org:22623/config/master** などの URL から Ignition を取得するためのコントロールプレーンノードのディレクティブが含まれます。この URL のポート番号はロードバランサーによって管理され、クラスター内でのみアクセスできます。

手順

1. コントロールプレーンノードを作成します。

```
$ ansible-playbook -i inventory.yml masters.yml
```

2. Playbook がコントロールプレーンを作成する間に、ブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete --dir $ASSETS_DIR
```

出力例

```
INFO API v1.18.3+b74c5ed up
INFO Waiting up to 40m0s for bootstrapping to complete...
```

3. コントロールプレーンノードおよび etcd のすべての Pod が実行されている場合、インストールプログラムは以下の出力を表示します。

出力例

```
INFO It is now safe to remove the bootstrap resources
```

13.4.21. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=$ASSETS_DIR/auth/kubeconfig
```

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

13.4.22. ブートストラップマシンの削除

wait-for コマンドがブートストラッププロセスが完了したことを示していることを確認したら、ブートストラップ仮想マシンを削除してコンピュート、メモリー、およびストレージリソースを解放する必要があります。また、ロードバランサーディレクティブからブートストラップマシンの設定を削除します。

手順

1. クラスターからブートストラップマシンを削除するには、以下を実行します。

```
$ ansible-playbook -i inventory.yml retire-bootstrap.yml
```

2. ロードバランサーディレクティブからブートストラップマシンの設定を削除します。

13.4.23. ワーカーノードの作成およびインストールの完了

ワーカーノードの作成は、コントロールプレーンノードの作成と同様です。ただし、ワーカーノードはクラスターに自動的に参加しません。これらをクラスターに追加するには、ワーカーの保留状態の CSR(証明書署名要求) を確認し、承認します。

最初の要求の承認後に、ワーカーノードがすべて承認されるまで CSR の承認を継続します。このプロセスが完了すると、ワーカーノードは **Ready** になり、Pod がそれらで実行されるようにスケジュールできます。

最後に、コマンドラインを監視し、インストールプロセスが完了するタイミングを確認します。

手順

1. ワーカーノードを作成します。

```
$ ansible-playbook -i inventory.yml workers.yml
```

2. すべての CSR を一覧表示するには、以下を入力します。

```
$ oc get csr -A
```

   最終的に、このコマンドはノードごとに 1 つの CSR を表示します。以下に例を示します。

   出力例

```
NAME        AGE    SIGNERNAME                                    REQUESTOR
CONDITION
csr-2lnxd   63m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master0.ocp4.example.org                                         Approved,Issued
csr-hff4q   64m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-hsn96   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master2.ocp4.example.org                                         Approved,Issued
csr-m724n   6m2s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-p4dz2   60m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-t9vfj   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master1.ocp4.example.org                                         Approved,Issued
csr-tggtr   61m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-wcbrf   7m6s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
```

3. 一覧をフィルターし、保留中の CSR のみを表示するには、以下を実行します。

```
$ watch "oc get csr -A | grep pending -i"
```

   このコマンドは 2 秒ごとに出力を更新し、保留中の CSR のみを表示します。以下に例を示します。

出力例

```
Every 2.0s: oc get csr -A | grep pending -i

csr-m724n   10m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-wcbrf   11m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
```

4. 保留中のそれぞれの要求を検査します。以下に例を示します。

出力例

```
$ oc describe csr csr-m724n
```

出力例

```
Name:               csr-m724n
Labels:             <none>
Annotations:        <none>
CreationTimestamp:  Sun, 19 Jul 2020 15:59:37 +0200
Requesting User:    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper
Signer:             kubernetes.io/kube-apiserver-client-kubelet
Status:             Pending
Subject:
        Common Name:    system:node:ocp4-lk6b4-worker1.ocp4.example.org
        Serial Number:
        Organization:   system:nodes
Events:  <none>
```

5. CSR 情報が正しい場合は、要求を承認します。

```
$ oc adm certificate approve csr-m724n
```

6. インストールプロセスが完了するまで待機します。

```
$ openshift-install wait-for install-complete --dir $ASSETS_DIR --log-level debug
```

インストールが完了すると、コマンドラインには OpenShift Container Platform Web コンソールの URL と、管理者のユーザー名およびパスワードが表示されます。

13.4.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

13.5. ネットワークが制限された環境での RHV へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、インストールリリースコンテンツの内部ミラーを作成して、カスタマイズされた OpenShift Container Platform クラスターをネットワークが制限された環境で Red Hat Virtualization (RHV) にインストールできます。

13.5.1. 前提条件

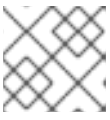
OpenShift Container Platform クラスターを RHV 環境にインストールするには、以下の要件を満たしている必要があります。

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- Support Matrix for OpenShift Container Platform on RHV に記載のサポートされるバージョンの組み合わせを使用できる。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

13.5.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベ

アメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

13.5.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

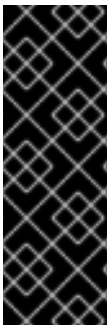
- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

13.5.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

13.5.4. RHV 環境の要件

OpenShift Container Platform バージョン 4.8 クラスターをインストールし、実行するには、RHV 環境が以下の要件を満たしている必要があります。

これらの要件を満たさないと、インストールまたはプロセスが失敗する可能性があります。さらに、これらの要件を満たしていないと、OpenShift Container Platform クラスターはインストールしてから数日または数週間後に失敗する可能性があります。

CPU、メモリー、ストレージリソースについての以下の要件は、インストールプログラムが作成する仮想マシンのデフォルト数で乗算した **デフォルト** 値に基づいています。これらのリソースは、RHV 環境が OpenShift Container Platform 以外の操作に使用するものに **加え**、利用可能でなければなりません。

デフォルトでは、インストールプログラムは 7 つの仮想マシンをインストールプロセスで作成します。まず、ブートストラップ仮想マシンを作成し、OpenShift Container Platform クラスターの残りの部分を作成する間に一時サービスとコントロールプレーンを提供します。インストールプログラムがクラスターの作成を終了すると、ブートストラップマシンが削除され、そのリソースが解放されます。

RHV 環境の仮想マシン数を増やす場合は、リソースを適宜増やす必要があります。

要件

- RHV のバージョンは 4.4 である。
- RHV 環境に **Up** 状態のデータセンターが 1 つあること。
- RHV データセンターに RHV クラスターが含まれていること。
- RHV クラスターに OpenShift Container Platform クラスター専用の以下のリソースがあること。
    - 最小 28 vCPU: インストール時に作成される 7 仮想マシンのそれぞれに 4 vCPU。
    - 以下を含む 112 GiB 以上の RAM。
        - 一時的なコントロールプレーンを提供するブートストラップマシン用に 16 GiB 以上。
        - コントロールプレーンを提供する 3 つのコントロールプレーンマシンのそれぞれに 16 GiB 以上。
        - アプリケーションワークロードを実行する 3 つのコンピュートマシンのそれぞれに 16 GiB 以上。
- RHV ストレージドメインは、これらの etcd バックエンドのパフォーマンス要件 を満たす必要があります。
- 実稼働環境では、各仮想マシンに 120 GiB 以上が必要です。そのため、ストレージドメインはデフォルトの OpenShift Container Platform クラスターに 840 GiB 以上を提供する必要があります。リソースに制約のある環境または非実稼働環境では、各仮想マシンに 32 GiB 以上を指定する必要があるため、ストレージドメインにはデフォルトの OpenShift Container Platform クラスター用に 230 GiB 以上が必要になります。
- インストールおよび更新中に Red Hat Ecosystem Catalog からイメージをダウンロードするには、RHV クラスターがインターネット接続にアクセスできる必要があります。また、サブスクリプションおよびエンタイトルメントプロセスを単純化するために Telemetry サービスにもインターネット接続が必要です。
- RHV クラスターには、RHV Manager の REST API にアクセスできる仮想ネットワークが必要です。インストーラーが作成する仮想マシンが DHCP を使用して IP アドレスを取得するため、DHCP がこのネットワークで有効にされていることを確認します。
- ターゲット RHV クラスターに OpenShift Container Platform クラスターをインストールし、管理するための以下の最小限の権限を持つユーザーアカウントおよびグループ。
    - **DiskOperator**
    - **DiskCreator**
    - **UserTemplateBasedVm**
    - **TemplateOwner**

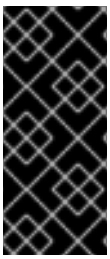
- **TemplateCreator**
- ターゲットクラスターの **ClusterAdmin**

警告

最小権限の原則を適用します。インストールプロセスで RHV で **SuperUser** 権限を持つ管理者アカウントを使用することを避けます。インストールプログラムは、ユーザーが指定する認証情報を、危険にさらされる可能性のある一時的な **ovirt-config.yaml** ファイルに保存します。

13.5.5. RHV 環境の要件の確認

RHV 環境が OpenShift Container Platform クラスターをインストールし、実行するための要件を満たしていることを確認します。これらの要件を満たさないと、エラーが発生する可能性があります。



重要

これらの要件は、インストールプログラムがコントロールプレーンおよびコンピュートマシンの作成に使用するデフォルトのリソースに基づいています。これらのリソースには、vCPU、メモリー、およびストレージが含まれます。これらのリソースを変更するか、または OpenShift Container Platform マシンの数を増やす場合は、これらの要件を適宜調整します。

手順

1. RHV のバージョンが OpenShift Container Platform バージョン 4.8 のインストールをサポートすることを確認します。
   a. RHV Administration Portal の右上にある ? ヘルプアイコンをクリックし、**About** を選択します。
   b. 開かれるウィンドウで、**RHV ソフトウェアのバージョン** をメモします。
   c. RHV のバージョンが 4.4 であることを確認します。サポートされるバージョンの組み合わせについての詳細は、Support Matrix for OpenShift Container Platform on RHV を参照してください。
2. データセンター、クラスター、およびストレージを検査します。
   a. RHV 管理ポータルで、**Compute → Data Centers** をクリックします。
   b. OpenShift Container Platform をインストールする予定のデータセンターにアクセスできることを確認します。
   c. そのデータセンターの名前をクリックします。
   d. データセンターの詳細の **Storage** タブで、OpenShift Container Platform をインストールする予定のストレージドメインが **Active** であることを確認します。
   e. 後で使用できるように **ドメイン名** を記録します。

f. **空き領域** に 230 GiB 以上あることを確認します。

g. ストレージドメインが これらの etcd バックエンドのパフォーマンス要件 を満たしていることを確認します。これは、fio パフォーマンスベンチマークツールを使用して測定できます。

h. データセンターの詳細で、**Clusters** タブをクリックします。

i. OpenShift Container Platform をインストールする予定の RHV クラスターを見つけます。後で使用できるようにクラスター名を記録します。

3. RHV ホストリソースを確認します。

a. RHV 管理ポータルで、**Compute** > **Clusters** をクリックします。

b. OpenShift Container Platform をインストールする予定のクラスターをクリックします。

c. クラスターの詳細で、**Hosts** タブをクリックします。

d. ホストを検査し、それらに OpenShift Container Platform クラスター **専用** として利用可能な **論理 CPU コア** の合計が 28 つ以上であることを確認します。

e. 後で使用できるように、利用可能な **論理 CPU コア** の数を記録します。

f. これらの CPU コアが分散され、インストール時に作成された 7 つの仮想マシンのそれぞれに 4 つのコアを持たせることができることを確認します。

g. ホストには、以下の OpenShift Container Platform マシンのそれぞれの要件を満たすように **新規仮想マシンをスケジュールするための最大空きメモリー** として 112 GiB があることを確認します。

- ブートストラップマシンに 16 GiB が必要です。
- 3 つのコントロールプレーンマシンのそれぞれに 16 GiB が必要です。
- 3 つのコンピュートマシンのそれぞれに 16 GiB が必要です。

h. 後で使用できるように **新規仮想マシンをスケジュールするための最大空きメモリー** の量を記録します。

4. OpenShift Container Platform をインストールするための仮想ネットワークが RHV Manager の REST API にアクセスできることを確認します。このネットワーク上の仮想マシンから、RHV Manager の REST API に到達するために curl を使用します。

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** については、RHV で OpenShift Container Platform クラスターを作成および管理する権限を持つ RHV アカウントのユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。**<password>** に、そのユーザー名のパスワードを指定します。

2 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ocpadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

13.5.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

ファイアウォール

クラスターが必要なサイトにアクセスできるようにファイアウォールを設定します。

以下も参照してください。

- Red Hat Virtualization Manager ファイアウォールの要件
- ホストのファイアウォール要件

DNS

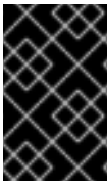
インフラストラクチャーで提供される DNS を設定して、主要なコンポーネントとサービスの正しい解決を許可します。1 つのロードバランサーのみを使用する場合、これらの DNS レコードは同じ IP アドレスを参照できます。

- **api.<cluster_name>.<base_domain>** (内部および外部解決) と、コントロールプレーンマシンのロードバランサーを参照する **api-int.<cluster_name>.<base_domain>** (内部解決) の DNS レコードを作成します。
- Ingress ルーターのロードバランサーを参照する ***.apps.<cluster_name>.<base_domain>** の DNS レコードを作成します。たとえば、コンピュートマシンのポート **443** および **80** などが含まれます。

13.5.6.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表13.9 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表13.10 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表13.11 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

13.5.7. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

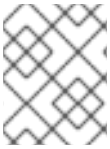
以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表13.12 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コンピュートマシン	**<worker><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

13.5.7.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

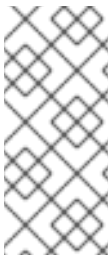
BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例13.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
```

```
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例13.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
```

```
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

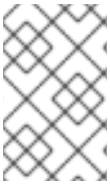
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

13.5.7.2. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



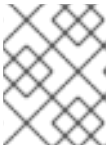
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表13.13 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。

- 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

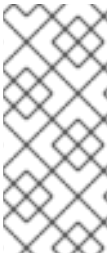
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表13.14 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

13.5.7.2.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例13.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
```

```
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

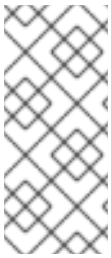
2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

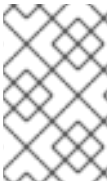


注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

13.5.8. インストールマシンの設定

バイナリー **openshift-install** インストールプログラムおよび Ansible スクリプトを実行するには、Manager 上の RHV 環境および REST API にネットワークでアクセスできるように、RHV Manager または Red Hat Enterprise Linux (RHEL) を設定します。

手順

1. Python3 および Ansible を更新またはインストールします。以下に例を示します。

```
# dnf update python3 ansible
```

2. **python3-ovirt-engine-sdk4** パッケージをインストール して、Python Software Development Kit を取得します。

3. **ovirt.image-template** Ansible ロールをインストールします。RHV Manager およびその他の Red Hat Enterprise Linux (RHEL) マシンでは、このロールは **ovirt-ansible-image-template** パッケージとして提供されます。たとえば、 以下を入力します。

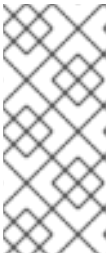
```
# dnf install ovirt-ansible-image-template
```

4. **ovirt.vm-infra** Ansible ロールをインストールします。RHV Manager およびその他の RHEL マシンでは、このロールは **ovirt-ansible-vm-infra** パッケージとして提供されます。

```
# dnf install ovirt-ansible-vm-infra
```

5. 環境変数を作成し、その環境変数に絶対パスまたは相対パスを割り当てます。たとえば、 以下を入力します。

```
$ export ASSETS_DIR=./wrk
```



注記

インストールプログラムはこの変数を使用して、重要なインストール関連のファイルを保存するディレクトリーを作成します。その後、インストールプロセスはこの変数を再利用して、これらのアセットファイルを見つけます。このアセットディレクトリーを削除しないでください。これは、クラスターのアンインストールに必要になります。

13.5.9. RHV 用の CA 証明書の設定

Red Hat Virtualization (RHV) Manager から CA 証明書をダウンロードし、インストールマシンにこれを設定します。

RHV Manager からの Web サイトまたは **curl** コマンドを使用して、証明書をダウンロードできます。

その後、インストールプログラムに証明書を提供します。

手順

1. 以下の 2 つの方法のいずれかを使用して CA 証明書をダウンロードします。

   - Manager の Web ページ (**https://<engine-fqdn>/ovirt-engine/**) に移動します。次に、**Downloads** で **CA Certificate** のリンクをクリックします。

   - 以下のコマンドを実行します。

```
$ curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem
```
(1)

(1) **<engine-fqdn>** には、RHV Manager の完全修飾ドメイン名 (例: **rhv-env.virtlab.example.com**) を指定します。

2. ルートレスユーザーに Manager へのアクセスを付与するように CA ファイルを設定します。CA ファイルのパーミッションを 8 進数の **0644** に設定します (シンボリック値: **-rw-r—r--**):

```
$ sudo chmod 0644 /tmp/ca.pem
```

3. Linux の場合は、サーバー証明書のディレクトリーに CA 証明書をコピーします。**-p** を使用してパーミッションを保存します。

```
$ sudo cp -p /tmp/ca.pem /etc/pki/ca-trust/source/anchors/ca.pem
```

4. オペレーティングシステム用の証明書マネージャーに証明書を追加します。

- MacOS の場合は、証明書ファイルをダブルクリックして、**Keychain Access** ユーティリティーを使用してファイルを **System** キーチェーンに追加します。
- Linux の場合は、CA 信頼を更新します。

```
$ sudo update-ca-trust
```

注記

独自の認証局を使用する場合は、システムがこれを信頼することを確認します。

関連情報

- 詳細は、RHV ドキュメントの Authentication and Security を参照してください。

13.5.10. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

13.5.11. Ansible Playbook のダウンロード

RHV に OpenShift Container Platform バージョン 4.8 をインストールするために Ansible Playbook をダウンロードします。

手順

- インストールマシンで、以下のコマンドを実行します。

```
$ mkdir playbooks
```

```
$ cd playbooks
```

```
$ curl -s -L -X GET https://api.github.com/repos/openshift/installer/contents/upi/ovirt?
ref=release-4.8 |
grep 'download_url.*\.yml' |
awk '{ print $2 }' | sed -r 's/("|",)//g' |
xargs -n 1 curl -O
```

次のステップ

- これらの Ansible Playbook をダウンロードしたら、インストールプログラムを実行してインストール設定ファイルを作成する前に、アセットディレクトリーの環境変数を作成し、**inventory.yml** ファイルをカスタマイズする必要もあります。

13.5.12. inventory.yml ファイル

inventory.yml ファイルを使用して、インストールする OpenShift Container Platform クラスターの各種の要素を定義し、作成します。これには、Red Hat Enterprise Linux CoreOS(RHCOS) イメージ、仮

想マシンテンプレート、ブートストラップマシン、コントロールプレーンノード、ワーカーノードなどの要素が含まれます。また、**inventory.yml** を使用してクラスターを破棄します。

以下の **inventory.yml** の例は、パラメーターとそれらのデフォルト値を示しています。これらのデフォルト値の量と数は、RHV 環境で実稼働用の OpenShift Container Platform クラスターを実行するための要件を満たしています。

inventory.yml ファイルの例

```
---
all:
  vars:

    ovirt_cluster: "Default"
    ocp:
      assets_dir: "{{ lookup('env', 'ASSETS_DIR') }}"
      ovirt_config_path: "{{ lookup('env', 'HOME') }}/.ovirt/ovirt-config.yaml"

    # ---
    # {op-system} section
    # ---
    rhcos:
      image_url: "https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.8/latest/rhcos-openstack.x86_64.qcow2.gz"
      local_cmp_image_path: "/tmp/rhcos.qcow2.gz"
      local_image_path: "/tmp/rhcos.qcow2"

    # ---
    # Profiles section
    # ---
    control_plane:
      cluster: "{{ ovirt_cluster }}"
      memory: 16GiB
      sockets: 4
      cores: 1
      template: rhcos_tpl
      operating_system: "rhcos_x64"
      type: high_performance
      graphical_console:
        headless_mode: false
        protocol:
        - spice
        - vnc
      disks:
      - size: 120GiB
        name: os
        interface: virtio_scsi
        storage_domain: depot_nvme
      nics:
      - name: nic1
        network: lab
        profile: lab

    compute:
      cluster: "{{ ovirt_cluster }}"
      memory: 16GiB
```

```
  sockets: 4
  cores: 1
  template: worker_rhcos_tpl
  operating_system: "rhcos_x64"
  type: high_performance
  graphical_console:
    headless_mode: false
    protocol:
    - spice
    - vnc
  disks:
  - size: 120GiB
    name: os
    interface: virtio_scsi
    storage_domain: depot_nvme
  nics:
  - name: nic1
    network: lab
    profile: lab

# ---
# Virtual machines section
# ---
vms:
- name: "{{ metadata.infraID }}-bootstrap"
  ocp_type: bootstrap
  profile: "{{ control_plane }}"
  type: server
- name: "{{ metadata.infraID }}-master0"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master1"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-master2"
  ocp_type: master
  profile: "{{ control_plane }}"
- name: "{{ metadata.infraID }}-worker0"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker1"
  ocp_type: worker
  profile: "{{ compute }}"
- name: "{{ metadata.infraID }}-worker2"
  ocp_type: worker
  profile: "{{ compute }}"
```

重要

Enter から始まる説明のあるパラメーターの値を入力します。 それ以外の場合は、デフォルト値を使用するか、またはこえを新しい値に置き換えることができます。

General セクション

- **ovirt_cluster**: OpenShift Container Platform クラスターをインストールする既存の RHV クラスターの名前を入力します。
- **ocp.assets_dir**: **openshift-install** インストールプログラムが生成するファイルを保存するために作成するディレクトリーのパス。
- **ocp.ovirt_config_path**: インストールプログラムが生成する **ovirt-config.yaml** ファイルのパス (**./wrk/install-config.yaml** など)。このファイルには、Manager の REST API との対話に必要な認証情報が含まれます。

Red Hat Enterprise Linux CoreOS (RHCOS) セクション

- **image_url**: ダウンロード用に指定した RHCOS イメージの URL を入力します。
- **local_cmp_image_path**: 圧縮された RHCOS イメージのローカルダウンロードディレクトリーのパス。
- **local_image_path**: 展開した RHCOS イメージのローカルディレクトリーのパス。

Profiles セクション

このセクションは、2 つのプロファイルで設定されます。

- **control_plane**: ブートストラップおよびコントロールプレーンノードのプロファイル。
- **compute**: コンピュートプレーン内のワーカーノードのプロファイル。

これらのプロファイルには以下のパラメーターが含まれます。パラメーターのデフォルト値は、実稼働クラスターを実行するために必要な最小要件を満たします。これらの値は、ワークロードの要件に応じて増減したり、カスタマイズしたりできます。

- **cluster**: 値は、General セクションの **ovirt_cluster** からクラスター名を取得します。
- **memory**: 仮想マシンに必要なメモリーの量 (GB)。
- **sockets**: 仮想マシンのソケット数。
- **cores**: 仮想マシンのコア数。
- **template**: 仮想マシンテンプレートの名前。複数のクラスターをインストールする計画があり、これらのクラスターが異なる仕様が含まれるテンプレートを使用する場合には、テンプレート名の先頭にクラスターの ID を付けます。
- **operating_system**: 仮想マシンのゲストオペレーティングシステムのタイプ。oVirt/RHV バージョン 4.4 では、**Ignition script** の値を仮想マシンに渡すことができるようにするために、この値を **rhcos_x64** にする必要があります。
- **type**: 仮想マシンのタイプとして **server** を入力します。

重要

type パラメーターの値を **high_performance** から **server** に変更する必要があります。

- **disks**: ディスクの仕様。**control_plane** と **compute** ノードには、異なるストレージドメインを設定できます。

- **size**: ディスクの最小サイズ。
- **name**: RHV のターゲットクラスターに接続されたディスクの名前を入力します。
- **interface**: 指定したディスクのインターフェイスタイプを入力します。
- **storage_domain**: 指定したディスクのストレージドメインを入力します。
- **nics**: 仮想マシンが使用する **name** および **network** を入力します。仮想ネットワークインターフェイスプロファイルを指定することもできます。デフォルトでは、NIC は oVirt/RHV MAC プールから MAC アドレスを取得します。

仮想マシンセクション

この最後のセクション **vms** は、クラスターで作成およびデプロイする予定の仮想マシンを定義します。デフォルトで、実稼働環境用の最小数のコントロールプレーンおよびワーカーノードが提供されます。

vms には 3 つの必須要素が含まれます。

- **name**: 仮想マシンの名前。この場合、**metadata.infraID** は、仮想マシン名の先頭に **metadata.yml** ファイルのインフラストラクチャー ID を付けます。
- **ocp_type**: OpenShift Container Platform クラスター内の仮想マシンのロール。使用できる値は **bootstrap**、**master**、**worker** です。
- **profile**: それぞれの仮想マシンが仕様を継承するプロファイルの名前。この例で使用可能な値は **control_plane** または **compute** です。
  仮想マシンがプロファイルから継承する値を上書きできます。これを実行するには、**inventory.yml** の仮想マシンに profile 属性の名前を追加し、これに上書きする値を割り当てます。この例を確認するには、直前の **inventory.yml** の例の **name: "{{ metadata.infraID }}-bootstrap"** 仮想マシンを検査します。これには値が **server** の **type** 属性があり、この仮想マシンがそれ以外の場合に **control_plane** プロファイルから継承する **type** 属性の値を上書きします。

メタデータ変数

仮想マシンの場合、**metadata.infraID** は、仮想マシンの名前の先頭に、Ignition ファイルのビルド時に作成する **metadata.json** ファイルのインフラストラクチャー ID を付けます。

Playbook は以下のコードを使用して、**ocp.assets_dir** にある特定のファイルから **infraID** を読み取ります。

```
---
- name: include metadata.json vars
  include_vars:
    file: "{{ ocp.assets_dir }}/metadata.json"
    name: metadata

  ...
```

13.5.13. RHCOS イメージ設定の指定

inventory.yml ファイルの Red Hat Enterprise Linux CoreOS (RHCOS) イメージ設定を更新します。後にこのファイルを Playbook のいずれかとして実行すると、圧縮された Red Hat Enterprise Linux CoreOS (RHCOS) イメージが **image_url** URL から **local_cmp_image_path** ディレクトリーにダウン

ロードされます。次に Playbook はイメージを **local_image_path** ディレクトリーに展開し、これを使用して oVirt/RHV テンプレートを作成します。

手順

1. インストールする OpenShift Container Platform バージョンの RHCOS イメージダウンロードページを見つけます (例: /pub/openshift-v4/dependencies/rhcos/latest/latest のインデックス)。
2. そのダウンロードページから、**https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.8/latest/rhcos-openstack.x86_64.qcow2.gz** などの OpenStack **qcow2** イメージの URL をコピーします。
3. 先のステップでダウンロードした **inventory.yml** Playbook を編集します。この中で、URL を **image_url** の値として貼り付けます。以下に例を示します。

```
rhcos:
  "https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.8/latest/rhcos-openstack.x86_64.qcow2.gz"
```

13.5.14. インストール設定ファイルの作成

インストールプログラム **openshift-install** を実行し、先に指定または収集した情報でプロンプトに応答し、インストール設定ファイルを作成します。

プロンプトに応答すると、インストールプログラムは、以前に指定したアセットディレクトリーの **install-config.yaml** ファイルの初期バージョンを作成します (例: **./wrk/install-config.yaml**)。

インストールプログラムは、Manager に到達して REST API を使用するために必要なすべての接続パラメーターが含まれる **$HOME/.ovirt/ovirt-config.yaml** ファイルも作成します。

注: インストールプロセスでは、**Internal API virtual IP** および **Ingress virtual IP** などの一部のパラメーターに指定する値を使用しません。それらの値はインフラストラクチャー DNS にすでに設定されているためです。

また、**oVirt cluster**、 **oVirt storage**、および **oVirt network** などの値のような **inventory.yml** のパラメーターに指定する値を使用します。また、スクリプトを使用して **install-config.yaml** の同じ値を削除するか、またはこれを前述の **virtual IPs** に置き換えます。

手順

1. インストールプログラムを実行します。

```
$ openshift-install create install-config --dir $ASSETS_DIR
```

2. インストールプログラムのプロンプトに応答し、システムに関する情報を提供します。

出力例

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
```

```
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

```
? SSH Public Key /home/user/.ssh/id_dsa.pub
? Platform <ovirt>
? Engine FQDN[:PORT] [? for help] <engine.fqdn>
? Enter ovirt-engine username <ocpadmin@internal>
? Enter password <******>
? oVirt cluster <cluster>
? oVirt storage <storage>
? oVirt network <net>
? Internal API virtual IP <172.16.0.252>
? Ingress virtual IP <172.16.0.251>
? Base Domain <example.org>
? Cluster Name <ocp4>
? Pull Secret [? for help] <********>
```

Internal API virtual IP および **Ingress virtual IP** について、DNS サービスの設定時に指定した IP アドレスを指定します。

さらに、**oVirt cluster** および **Base Domain** プロンプトに対して入力する値は REST API および作成するアプリケーションの URL の一部を設定します (例: **https://api.ocp4.example.org:6443/** and **https://console-openshift-console.apps.ocp4.example.org**)。

Red Hat OpenShift Cluster Manager からプルシークレット を取得できます。

13.5.15. IBM Z のサンプル install-config.yaml ファイル

13.5.16. RHV のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

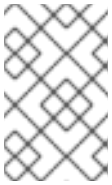
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
```

```
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



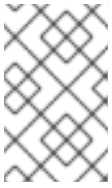
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



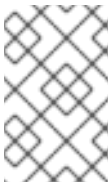
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネット



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。RHV インフラストラクチャー。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

13.5.16.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

–

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

13.5.17. install-config.yaml のカスタマイズ

ここでは、3 つの python スクリプトを使用して、インストールプログラムのデフォルト動作の一部を上書きします。

- デフォルトでは、インストールプログラムはマシン API を使用してノードを作成します。このデフォルトの動作を上書きするには、コンピュートノードの数をゼロ (0) レプリカに設定します。後に Ansible Playbook を使用してコンピュートノードを作成します。
- デフォルトでは、インストールプログラムはノードのマシンネットワークの IP 範囲を設定します。このデフォルトの動作を上書きするには、インフラストラクチャーに一致するように IP 範囲を設定します。
- デフォルトでは、インストールプログラムはプラットフォームを **ovirt** に設定します。ただし、ユーザーによってプロビジョニングされるインフラストラクチャーにクラスターをインストールすることは、ベアメタルにクラスターをインストールすることに似ています。したがっ

て、ovirt プラットフォームセクションを **install-config.yaml** から削除し、プラットフォームを **none** に変更します。代わりに、**inventory.yml** を使用して、必要な設定をすべて指定します。

注記

これらのスニペットは Python 3 および Python 2 で動作します。

手順

1. コンピュートノードの数をゼロ (0) レプリカに設定します。

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["compute"][0]["replicas"] = 0
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

2. マシンネットワークの IP 範囲を設定します。たとえば、範囲を **172.16.0.0/16** に設定するには、以下を実行します。

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
conf["networking"]["machineNetwork"][0]["cidr"] = "172.16.0.0/16"
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

3. **ovirt** セクションを削除し、プラットフォームを **none** に変更します。

```
$ python3 -c 'import os, yaml
path = "%s/install-config.yaml" % os.environ["ASSETS_DIR"]
conf = yaml.safe_load(open(path))
platform = conf["platform"]
del platform["ovirt"]
platform["none"] = {}
open(path, "w").write(yaml.dump(conf, default_flow_style=False))'
```

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールするのと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

13.5.18. マニフェストファイルの生成

インストールプログラムを使用して、アセットディレクトリーにマニフェストファイルのセットを生成します。

マニフェストファイルを生成するコマンドにより、**install-config.yaml** ファイルを使用する前に警告メッセージが表示されます。

install-config.yaml ファイルを再利用する予定の場合には、マニフェストファイルを生成する前にバックアップしてからバックアップコピーを作成してください。

手順

1. オプション: **install-config.yaml** ファイルのバックアップコピーを作成します。

```
$ cp install-config.yaml install-config.yaml.backup
```

2. アセットディレクトリーにマニフェストのセットを生成します。

```
$ openshift-install create manifests --dir $ASSETS_DIR
```

このコマンドにより、以下の情報が表示されます。

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

このコマンドにより、以下のマニフェストファイルが生成されます。

出力例

```
$ tree
.
└── wrk
    ├── manifests
    │   ├── 04-openshift-machine-config-operator.yaml
    │   ├── cluster-config.yaml
    │   ├── cluster-dns-02-config.yml
    │   ├── cluster-infrastructure-02-config.yml
    │   ├── cluster-ingress-02-config.yml
    │   ├── cluster-network-01-crd.yml
    │   ├── cluster-network-02-config.yml
    │   ├── cluster-proxy-01-config.yaml
    │   ├── cluster-scheduler-02-config.yml
    │   ├── cvo-overrides.yaml
    │   ├── etcd-ca-bundle-configmap.yaml
    │   ├── etcd-client-secret.yaml
    │   ├── etcd-host-service-endpoints.yaml
    │   ├── etcd-host-service.yaml
    │   ├── etcd-metric-client-secret.yaml
    │   ├── etcd-metric-serving-ca-configmap.yaml
    │   ├── etcd-metric-signer-secret.yaml
    │   ├── etcd-namespace.yaml
```

```
│   ├── etcd-service.yaml
│   ├── etcd-serving-ca-configmap.yaml
│   ├── etcd-signer-secret.yaml
│   ├── kube-cloud-config.yaml
│   ├── kube-system-configmap-root-ca.yaml
│   ├── machine-config-server-tls-secret.yaml
│   └── openshift-config-secret-pull-secret.yaml
└── openshift
    ├── 99_kubeadmin-password-secret.yaml
    ├── 99_openshift-cluster-api_master-user-data-secret.yaml
    ├── 99_openshift-cluster-api_worker-user-data-secret.yaml
    ├── 99_openshift-machineconfig_99-master-ssh.yaml
    ├── 99_openshift-machineconfig_99-worker-ssh.yaml
    └── openshift-install-manifests.yaml
```

次のステップ

- コントロールプレーンノードをスケジュール対象外にします。

13.5.19. コントロールプレーンノードのスケジュール対象外の設定

コントロールプレーンマシンを手動で作成し、デプロイしているので、コントロールプレーンノードをスケジュール対象外にするようにマニフェストファイルを設定する必要があります。

手順

1. コントロールプレーンノードをスケジュール対象外にするには、以下を入力します。

```
$ python3 -c 'import os, yaml
path = "%s/manifests/cluster-scheduler-02-config.yml" % os.environ["ASSETS_DIR"]
data = yaml.safe_load(open(path))
data["spec"]["mastersSchedulable"] = False
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

13.5.20. Ignition ファイルのビルド

生成および変更したマニフェストファイルから Ignition ファイルを作成するには、インストールプログラムを実行します。このアクションにより、Ignition ファイルをフェッチし、ノードを作成するために必要な設定を実行する Red Hat Enterprise Linux CoreOS (RHCOS) マシン **initramfs** が作成されます。

Ignition ファイルのほかに、インストールプログラムは以下を生成します。

- **oc** および **kubectl** ユーティリティーを使用してクラスターに接続するための管理者認証情報が含まれる **auth** ディレクトリー。
- OpenShift Container Platform クラスター名、クラスター ID、および現行インストールのインフラストラクチャー ID などの情報を含む **metadata.json** ファイル。

このインストールプロセスの Ansible Playbook は、**infraID** の値を、作成する仮想マシンの接頭辞として使用します。これにより、同じ oVirt/RHV クラスターに複数のインストールがある場合の命名の競合が回避されます。

注記

Ignition 設定ファイルの証明書は 24 時間後に有効期限が切れます。最初の証明書のローテーションが終了するように、クラスターのインストールを完了し、クラスターを動作が低下していない状態で 24 時間実行し続ける必要があります。

手順

1. Ignition ファイルをビルドするには、以下を入力します。

```
$ openshift-install create ignition-configs --dir $ASSETS_DIR
```

出力例

```
$ tree
.
└── wrk
    ├── auth
    │   ├── kubeadmin-password
    │   └── kubeconfig
    ├── bootstrap.ign
    ├── master.ign
    ├── metadata.json
    └── worker.ign
```

13.5.21. テンプレートおよび仮想マシンの作成

inventory.yml の変数を確認した後に、最初の Ansible プロビジョニング Playbook **create-templates-and-vms.yml** を実行します。

この Playbook は、**$HOME/.ovirt/ovirt-config.yaml** から RHV Manager の接続パラメーターを使用し、アセットディレクトリーで **metadata.json** を読み取ります。

ローカルの Red Hat Enterprise Linux CoreOS (RHCOS) イメージが存在しない場合、Playbook は **inventory.yml** の **image_url** に指定した URL からダウンロードします。これはイメージを展開し、これを RHV にアップロードしてテンプレートを作成します。

Playbook は、**inventory.yml** ファイルの **control_plane** と **compute** プロファイルに基づいてテンプレートを作成します。これらのプロファイルの名前が異なる場合、2 つのテンプレートが作成されます。

Playbook が完了すると、作成される仮想マシンは停止します。他のインフラストラクチャー要素の設定に役立つ情報を取得できます。たとえば、仮想マシンの MAC アドレスを取得して、仮想マシンに永続的な IP アドレスを割り当てるように DHCP を設定できます。

手順

1. **inventory.yml** の **control_plane** および **compute** 変数で、**type: high_performance** の 両方のインスタンスを **type: server** に変更します。

2. オプション: 同じクラスターに複数のインストールを実行する予定の場合には、OpenShift Container Platform インストールごとに異なるテンプレートを作成します。**inventory.yml** ファイルで、**template** の値の先頭に **infraID** を付けます。以下に例を示します。

```
control_plane:
```

```
    cluster: "{{ ovirt_cluster }}"
    memory: 16GiB
    sockets: 4
    cores: 1
    template: "{{ metadata.infraID }}-rhcos_tpl"
    operating_system: "rhcos_x64"
    ...
```

3. テンプレートおよび仮想マシンを作成します。

```
$ ansible-playbook -i inventory.yml create-templates-and-vms.yml
```

13.5.22. ブートストラップマシンの作成

bootstrap.yml Playbook を実行してブートストラップマシンを作成します。この Playbook はブートストラップ仮想マシンを起動し、これをアセットディレクトリーから **bootstrap.ign** Ignition ファイルに渡します。ブートストラップノードは、Ignition ファイルをコントロールプレーンノードに送信できるように設定します。

ブートストラッププロセスをモニターするには、RHV 管理ポータルでコンソールを使用するか、SSH を使用して仮想マシンに接続します。

手順

1. ブートストラップマシンを作成します。

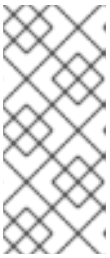
```
$ ansible-playbook -i inventory.yml bootstrap.yml
```

2. 管理ポータルまたは SSH のコンソールを使用してブートストラップマシンに接続します。**<bootstrap_ip>** をブートストラップノードの IP アドレスに置き換えます。SSH を使用するには、以下を入力します。

```
$ ssh core@<boostrap.ip>
```

3. ブートストラップノードからリリースイメージサービスについての **bootkube.service** journald ユニットログを収集します。

```
[core@ocp4-lk6b4-bootstrap ~]$ journalctl -b -f -u release-image.service -u bootkube.service
```



注記

ブートストラップノードの **bootkube.service** ログは、etcd の **connection refused** エラーを出力し、ブートストラップサーバーがコントロールプレーンノード (別名マスターノード) の etcd に接続できないことを示します。etcd が各コントロールプレーンノードで起動し、ノードがクラスターに参加した後には、エラーは発生しなくなるはずです。

13.5.23. コントロールプレーンノードの作成

masters.yml Playbook を実行してコントロールプレーンノードを作成します。この Playbook は **master.ign** Ignition ファイルをそれぞれの仮想マシンに渡します。Ignition ファイルには、**https://api-int.ocp4.example.org:22623/config/master** などの URL から Ignition を取得するためのコントロール

プレーンノードのディレクティブが含まれます。この URL のポート番号はロードバランサーによって管理され、クラスター内でのみアクセスできます。

手順

1. コントロールプレーンノードを作成します。

```
$ ansible-playbook -i inventory.yml masters.yml
```

2. Playbook がコントロールプレーンを作成する間に、ブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete --dir $ASSETS_DIR
```

出力例

```
INFO API v1.18.3+b74c5ed up
INFO Waiting up to 40m0s for bootstrapping to complete...
```

3. コントロールプレーンノードおよび etcd のすべての Pod が実行されている場合、インストールプログラムは以下の出力を表示します。

出力例

```
INFO It is now safe to remove the bootstrap resources
```

13.5.24. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=$ASSETS_DIR/auth/kubeconfig
```

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

13.5.25. ブートストラップマシンの削除

wait-for コマンドがブートストラッププロセスが完了したことを示していることを確認したら、ブートストラップ仮想マシンを削除してコンピュート、メモリー、およびストレージリソースを解放する必要があります。また、ロードバランサーディレクティブからブートストラップマシンの設定を削除します。

手順

1. クラスターからブートストラップマシンを削除するには、以下を実行します。

```
$ ansible-playbook -i inventory.yml retire-bootstrap.yml
```

2. ロードバランサーディレクティブからブートストラップマシンの設定を削除します。

13.5.26. ワーカーノードの作成およびインストールの完了

ワーカーノードの作成は、コントロールプレーンノードの作成と同様です。ただし、ワーカーノードはクラスターに自動的に参加しません。これらをクラスターに追加するには、ワーカーの保留状態の CSR(証明書署名要求) を確認し、承認します。

最初の要求の承認後に、ワーカーノードがすべて承認されるまで CSR の承認を継続します。このプロセスが完了すると、ワーカーノードは **Ready** になり、Pod がそれらで実行されるようにスケジュールできます。

最後に、コマンドラインを監視し、インストールプロセスが完了するタイミングを確認します。

手順

1. ワーカーノードを作成します。

```
$ ansible-playbook -i inventory.yml workers.yml
```

2. すべての CSR を一覧表示するには、以下を入力します。

```
$ oc get csr -A
```

最終的に、このコマンドはノードごとに 1 つの CSR を表示します。以下に例を示します。

出力例

```
NAME        AGE    SIGNERNAME                                    REQUESTOR
CONDITION
csr-2lnxd   63m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
master0.ocp4.example.org                                         Approved,Issued
csr-hff4q   64m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-hsn96   60m    kubernetes.io/kubelet-serving                 system:node:ocp4-lk6b4-
```

```
master2.ocp4.example.org                    Approved,Issued
csr-m724n   6m2s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-p4dz2   60m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-t9vfj   60m    kubernetes.io/kubelet-serving                system:node:ocp4-lk6b4-
master1.ocp4.example.org                    Approved,Issued
csr-tggtr   61m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
Approved,Issued
csr-wcbrf   7m6s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
```

3. 一覧をフィルターし、保留中の CSR のみを表示するには、以下を実行します。

```
$ watch "oc get csr -A | grep pending -i"
```

このコマンドは 2 秒ごとに出力を更新し、保留中の CSR のみを表示します。以下に例を示します。

出力例

```
Every 2.0s: oc get csr -A | grep pending -i

csr-m724n   10m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-wcbrf   11m   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
```

4. 保留中のそれぞれの要求を検査します。以下に例を示します。

出力例

```
$ oc describe csr csr-m724n
```

出力例

```
Name:              csr-m724n
Labels:            <none>
Annotations:       <none>
CreationTimestamp:  Sun, 19 Jul 2020 15:59:37 +0200
Requesting User:    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper
Signer:            kubernetes.io/kube-apiserver-client-kubelet
Status:            Pending
Subject:
        Common Name:    system:node:ocp4-lk6b4-worker1.ocp4.example.org
        Serial Number:
        Organization:   system:nodes
Events:  <none>
```

5. CSR 情報が正しい場合は、要求を承認します。

```
$ oc adm certificate approve csr-m724n
```

6. インストールプロセスが完了するまで待機します。

```
$ openshift-install wait-for install-complete --dir $ASSETS_DIR --log-level debug
```

インストールが完了すると、コマンドラインには OpenShift Container Platform Web コンソールの URL と、管理者のユーザー名およびパスワードが表示されます。

13.5.27. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

13.5.28. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

13.6. RHV でのクラスターのアンインストール

OpenShift Container Platform クラスターを Red Hat Virtualization (RHV) から削除することができます。

13.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

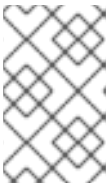
- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

13.6.2. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

クラスターの使用が完了したら、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターをクラウドから削除できます。

前提条件

- クラスターのインストールに使用した元の Playbook ファイル、アセットディレクトリーおよびファイル、および **$ASSETS_DIR** 環境変数が含まれます。通常、クラスターのインストール時に使用したのと同じコンピューターを使用してこれを実行できます。

手順

1. クラスターを削除するには、以下を入力します。

```
$ ansible-playbook -i inventory.yml \
    retire-bootstrap.yml \
    retire-masters.yml   \
    retire-workers.yml
```

2. DNS、ロードバランサー、およびこのクラスターの他のインフラストラクチャーに追加した設定を削除します。

第14章 VSPHERE へのインストール

14.1. VSPHERE へのインストールの準備

14.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターが必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。
- VMware プラットフォームのライセンスを確認している。Red Hat は VMware ライセンスに制限を設けていませんが、一部の VMware インフラストラクチャーコンポーネントにはライセンスが必要です。

14.1.2. vSphere に OpenShift Container Platform をインストールする方法の選択

インストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して vSphere に OpenShift Container Platform をインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform はユーザーが独自にプロビジョニングするインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

14.1.2.1. インストーラーでプロビジョニングされるインフラストラクチャーでの vSphere への OpenShift Container Platform のインストール

インストーラーでプロビジョニングされるインフラストラクチャーにより、インストールプログラムは OpenShift Container Platform で必要なリソースのプロビジョニングを事前に設定し、自動化することができます。

- **クラスターの vSphere へのインストール**: インストーラーでプロビジョニングされるインフラストラクチャーのインストールをカスタマイズせずに使用して、vSphere に OpenShift Container Platform をインストールできます。

- **カスタマイズによる vSphere へのクラスターのインストール**: インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトのカスタマイズオプションのインストールを使用して、vSphere に OpenShift Container Platform をインストールできます。
- **ネットワークのカスタマイズによる vSphere へのクラスターのインストール**: ネットワークのカスタマイズを使用して、インストーラーでプロビジョニングされる vSphere インフラストラクチャーに OpenShift Container Platform をインストールできます。インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。
- **ネットワークが制限された環境での vSphere へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを作成して、ネットワークが制限された環境で VMware vSphere インフラストラクチャーにクラスターをインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

14.1.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの vSphere への OpenShift Container Platform のインストール

ユーザーによってプロビジョニングされるインフラストラクチャーでは、ユーザーは OpenShift Container Platform に必要なすべてのリソースをプロビジョニングする必要があります。

- **ユーザーによってプロビジョニングされるインフラストラクチャーでの vSphere へのクラスターのインストール**: 独自にプロビジョニングする VMware vSphere インフラストラクチャーに OpenShift Container Platform をインストールできます。
- **カスタマイズされたネットワークを使用したユーザーによってプロビジョニングされるインフラストラクチャーの vSphere へのクラスターのインストール**: カスタマイズされたネットワーク設定オプションを使用して独自にプロビジョニングする VMware vSphere インフラストラクチャーに OpenShift Container Platform をインストールできます。
- **ネットワークが制限された環境でユーザーによってプロビジョニングされるインフラストラクチャーの vSphere へのクラスターのインストール**: ネットワークが制限された環境でプロビジョニングされる VMware vSphere インフラストラクチャーに、OpenShift Container Platform をインストールできます。

14.1.3. VMware vSphere インフラストラクチャーの要件

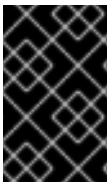
使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.1 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。

コンポーネント	サポートされる最小バージョン	説明
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.1.4. インストーラーでプロビジョニングされるインフラストラクチャーでの vSphere への OpenShift Container Platform のアンインストール

- **インストーラーでプロビジョニングされるインフラストラクチャーを使用する vSphere のクラスターのアンインストール**: インストーラーでプロビジョニングされるインフラストラクチャーを使用する VMware vSphere インフラストラクチャーにデプロイされたクラスターを削除できます。

14.2. クラスターの VSPHERE へのインストール

OpenShift Container Platform バージョン 4.8 では、インストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

14.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

14.2.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

14.2.3. VMware vSphere インフラストラクチャーの要件

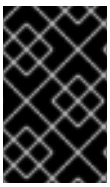
使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.2 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.2.4. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表14.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

14.2.5. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアク

セスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例14.1 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.Add**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		NewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.MarkAsTemplate VirtualMachine.Provisioning.DeployTemplate
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例14.2 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change

ロールの vSphere オブジェクト	必要になる場合	Configuration"."Add or remove device vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit

ロールの vSphere オブジェクト	必要になる場合	Inventory"."Create from existing vCenter GUI で必要な権限
		"Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例14.3 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.6 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

14.2.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

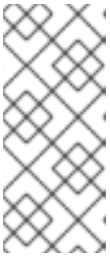
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

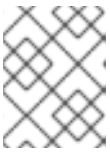
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

14.2.7. インストールプログラムの取得

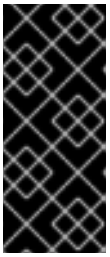
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

14.2.8. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

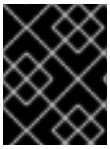
```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

14.2.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

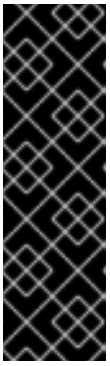
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

c. vCenter インスタンスの名前を指定します。

d. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。

e. 接続する vCenter インスタンスにあるデータセンターを選択します。

f. 使用するデフォルトの vCenter データストアを選択します。

注記

データストアとクラスター名は 60 文字を超えることができません。そのため、組み合わせた文字列の長さが 60 文字の制限を超えないようにしてください。

g. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

h. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

i. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

j. クラスター Ingress に設定した仮想 IP アドレスを入力します。

k. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

l. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

注記

データストアとクラスター名は 60 文字を超えることができません。そのため、組み合わせた文字列の長さが 60 文字の制限を超えないようにしてください。

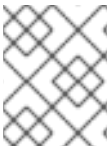
m. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

+ 出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

+

重要

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

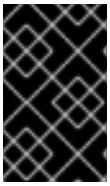
+

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

14.2.10. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。

PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

14.2.11. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

14.2.12. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

14.2.12.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

14.2.12.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.2.12.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

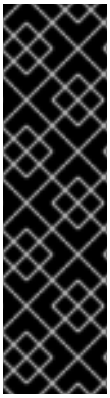
前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

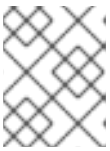
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

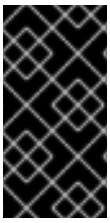
```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

14.2.12.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

3 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

4 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

14.2.13. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

14.2.14. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちま

す。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

14.2.15. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

14.2.16. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

14.3. カスタマイズによる VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、インストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

14.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

注記

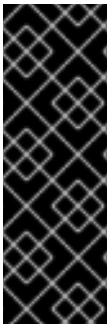
プロキシーを設定する場合は、このサイト一覧も確認してください。

14.3.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

14.3.3. VMware vSphere インフラストラクチャーの要件

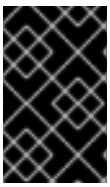
使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.7 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.3.4. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表14.8 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.9 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.10 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

14.3.5. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアク

セスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例14.4 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.Add**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		NewDisk **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例14.5 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Reset guest information" **"Virtual machine"."Change Configuration"."Change resource"** **"Virtual machine"."Change Configuration"."Change Settings"** **"Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility"** **"Virtual machine".Interaction."Guest operating system management by VIX API"** **"Virtual machine".Interaction."Power off"** **"Virtual machine".Interaction."Power on"** **"Virtual machine".Interaction.Reset** **"Virtual machine"."Edit Inventory"."Create new"** **"Virtual machine"."Edit Inventory"."Create from existing"** **"Virtual machine"."Edit Inventory"."Remove"** **"Virtual machine".Provisioning."Clone virtual machine"** **"Virtual machine".Provisioning."Mark as template"** **"Virtual machine".Provisioning."Deploy template"**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object"** **Resource."Assign virtual machine to resource pool"** **VApp.Import** **"Virtual machine"."Change Configuration"."Add existing disk"** **"Virtual machine"."Change Configuration"."Add new disk"** **"Virtual machine"."Change**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Inventory"."Create from existing "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例14.6 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.11 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

14.3.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

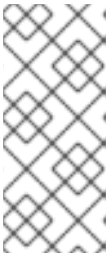
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

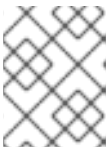
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

14.3.7. インストールプログラムの取得

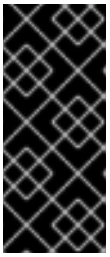
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

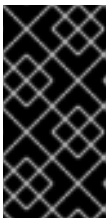
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

14.3.8. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates**をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

14.3.9. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

      iii. vCenter インスタンスの名前を指定します。

      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

      v. 接続する vCenter インスタンスにあるデータセンターを選択します。

      vi. 使用するデフォルトの vCenter データストアを選択します。

      vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

      viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xiii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

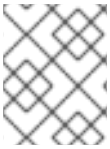
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

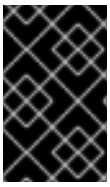
14.3.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

14.3.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表14.12 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

14.3.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表14.13 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

14.3.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表14.14 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列

パラメーター	説明	値
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

14.3.9.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表14.15 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列

パラメーター	説明	値
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

14.3.9.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表14.16 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数

パラメーター	説明	値
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

14.3.9.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
```

```
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name 9
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

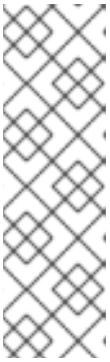
9 OpenShift Container Platform クラスターをインストールする vSphere クラスター。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

14.3.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

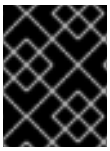
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

14.3.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

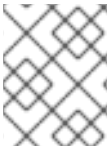
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

+

重要

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

+

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

14.3.11. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

14.3.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

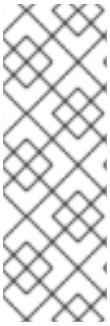
14.3.13. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

14.3.13.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

14.3.13.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

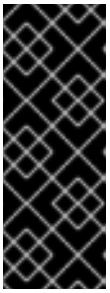
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.3.13.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

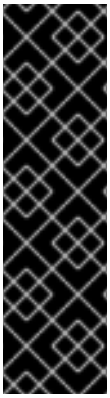
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



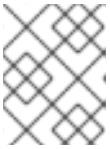
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

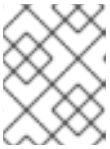
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

14.3.13.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

3 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

4 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

14.3.14. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

14.3.15. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちま

す。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

14.3.16. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

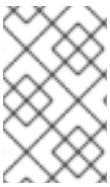
14.3.17. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

14.4. ネットワークのカスタマイズによる VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、カスタマイズされるネットワーク設定オプションと共にインストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。



注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

14.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

14.4.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

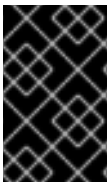
14.4.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.17 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.4.4. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表14.18 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-9999	ホストレベルのサービス。 ポート **9100-9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。

プロトコル	ポート	説明
	10250-10259	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-9999	ポート **9100-9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000-32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.19 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.20 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

14.4.5. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへ

の OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例14.7 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.Annotation **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.Adva**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		ncedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例14.8 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from

ロールの vSphere オブジェクト	必要になる場合	existing" vCenter GUI で必要な権限
		Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例14.9 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

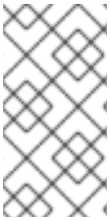
これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に

利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。



注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.21 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

14.4.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

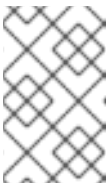
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

14.4.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

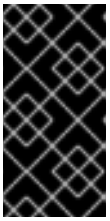
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

14.4.8. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

14.4.9. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

      iii. vCenter インスタンスの名前を指定します。

      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

      v. 接続する vCenter インスタンスにあるデータセンターを選択します。

      vi. 使用するデフォルトの vCenter データストアを選択します。

      vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

      viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

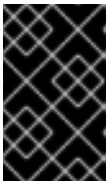
xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xiii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

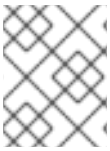


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

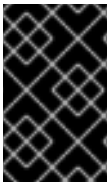
14.4.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

14.4.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表14.22 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

14.4.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表14.23 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

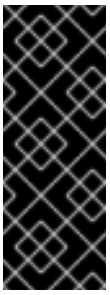
パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

14.4.9.1.3. オプションの設定パラメーター

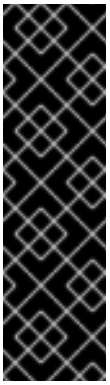
オプションのインストール設定パラメーターは、以下の表で説明されています。

表14.24 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列

パラメーター	説明	値
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

14.4.9.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表14.25 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列

パラメーター	説明	値
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

14.4.9.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表14.26 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数

パラメーター	説明	値
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

14.4.9.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
```

```
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name 9
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

9 OpenShift Container Platform クラスターをインストールする vSphere クラスター。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

14.4.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

14.4.10. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。

注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

14.4.11. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

14.4.12. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

14.4.12.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表14.27 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定

defaultNetwork オブジェクトの値は、以下の表で定義されます。

表14.28 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定

以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表14.29 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
```

```
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定
以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表14.30 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表14.31 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

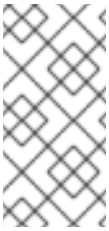
フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

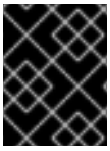
表14.32 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

14.4.13. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

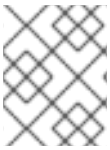
1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

   クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
```

```
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+



注記

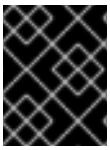
クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

+

重要

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

+



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

14.4.14. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

14.4.15. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

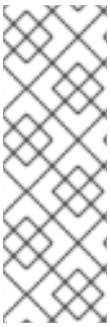
14.4.16. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

14.4.16.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないフラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

14.4.16.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

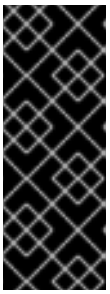
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.4.16.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

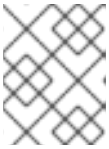
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

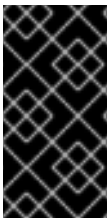
```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

14.4.16.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

3 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

4 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

14.4.17. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

14.4.18. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちま

す。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

14.4.19. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

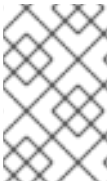
- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

14.4.20. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

14.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、プロビジョニングする VMware vSphere インフラストラクチャーにクラスターをインストールできます。



注記

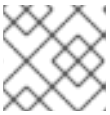
OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

14.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- インストールを完了するには、vSphere ホストに Red Hat Enterprise Linux CoreOS(RHCOS) OVA をアップロードする必要があります。このプロセスを完了するマシンには、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

14.5.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

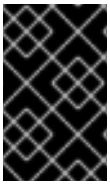
14.5.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.33 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

14.5.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表14.34 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

重要

すべての仮想マシンは、インストーラーと同じデータストアおよびフォルダーになければなりません。

14.5.4.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表14.35 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

14.5.4.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

14.5.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

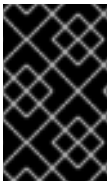
マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

14.5.4.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表14.36 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.37 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.38 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**

- **00:0c:29:00:00:00 - 00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00 - 00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00 - 00:50:56:FF:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

14.5.4.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

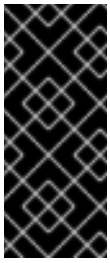
また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.39 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

14.5.4.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例14.10 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例14.11 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

14.5.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

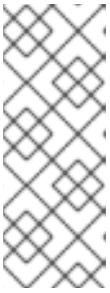
   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表14.40 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

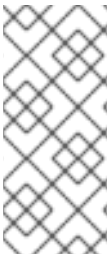
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表14.41 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

14.5.4.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例14.12 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

14.5.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

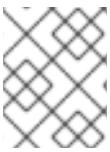
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

14.5.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

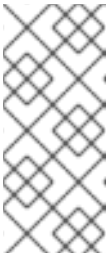
   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

14.5.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

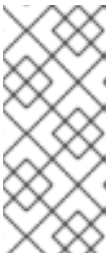
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

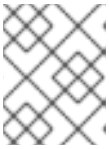
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

14.5.8. インストールプログラムの取得

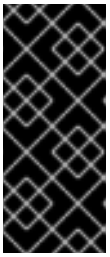
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

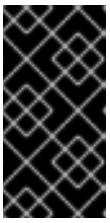
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

14.5.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

14.5.9.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths": ...}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

(4) **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロ

イする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 OpenShift Cluster Manager から取得したプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

14.5.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress

トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

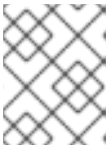
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

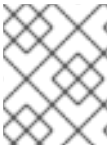


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

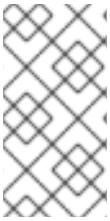
14.5.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

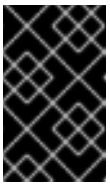
```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

14.5.11. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

14.5.12. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホ

ストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**

4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。

7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。

注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

- ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
- **install-config.yaml** ファイルで指定したデータストアを選択します。

f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。

重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。

i. 静的 IP 設定を行います。

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

コマンドの例

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

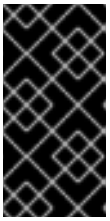
```
$ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
kargs=${IPCFG}"
```

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。VM の CPU とメモリーの予約に次の値があることを確認してください。
  - メモリー予約値は、設定されたメモリーサイズと同じである必要があります。
  - CPU 予約値は、低レイテンシー仮想 CPU の数に、実際に測定された物理 CPU 速度で乗算した数を最低でも指定する必要があります。
- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで、スチールクロックアカウンティング (**stealclock.enable**) に使用できるパラメーターのリストを検索します。利用可能な場合は、その値を **TRUE** に設定します。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。
- **設定パラメーターの追加** をクリックします。以下のパラメーター名および値を定義します。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

9. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

14.5.13. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。
   i. 設定を完了し、仮想マシンの電源をオンにします。
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

14.5.14. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。

重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成

一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

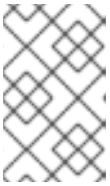
2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

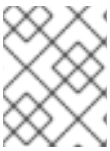
```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

14.5.15. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。



注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。

システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

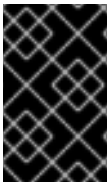
- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

14.5.16. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

14.5.17. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

14.5.18. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

14.5.19. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名

要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

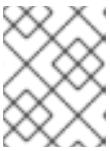
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

14.5.20. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

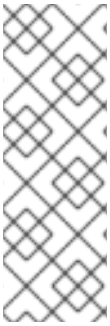
```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

14.5.20.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

14.5.20.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

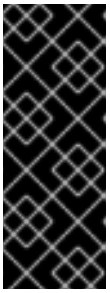
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.5.20.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

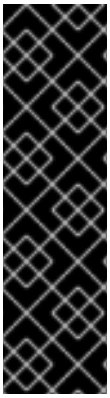
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

14.5.20.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

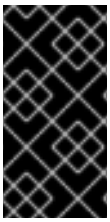
イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

14.5.20.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
   3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   4. 永続ボリューム要求 (PVC) のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

14.5.21. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
```

```
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                         READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                              1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                              1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                              1/1     Running     0
   ```

```
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

14.5.22. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

14.5.23. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを

追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

14.5.24. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

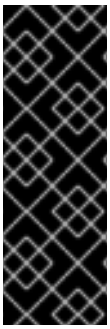
14.6. ネットワークのカスタマイズによる VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、カスタマイズされたネットワーク設定オプションでプロビジョニングする VMware vSphere インフラストラクチャーにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

14.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- インストールを完了するには、vSphere ホストに Red Hat Enterprise Linux CoreOS(RHCOS) OVA をアップロードする必要があります。このプロセスを完了するマシンには、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできる

ことを確認している。

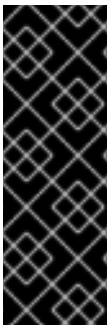
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

14.6.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

14.6.3. VMware vSphere インフラストラクチャーの要件

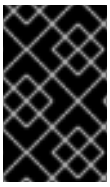
使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.42 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。

コンポーネント	サポートされる最小バージョン	説明
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.6.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

14.6.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表14.43 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。

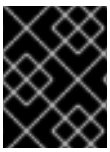
ホスト	説明
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。



重要

すべての仮想マシンは、インストーラーと同じデータストアおよびフォルダーになければなりません。

14.6.4.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表14.44 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

14.6.4.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認しま

す。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

14.6.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

14.6.4.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表14.45 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn

プロトコル	ポート	説明
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.46 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.47 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** - **00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00** - **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** - **00:50:56:FF:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイム

サーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

14.6.4.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

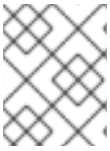
以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.48 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コンピュートマシン	**<worker><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

14.6.4.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例14.13 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
```

```
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例14.14 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
```

```
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

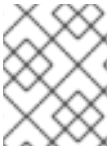
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

14.6.4.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表14.49 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。

- 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

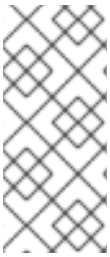
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

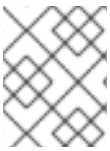
表14.50 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

14.6.4.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例14.15 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster ❶
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 ❷
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup ❸
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 ❹
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup ❺
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
```

```
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

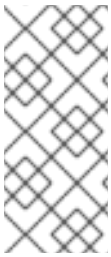
2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

14.6.5. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

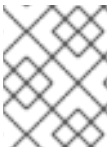
前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

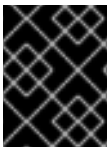


注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

14.6.6. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

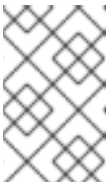
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

14.6.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定しま

す。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

14.6.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

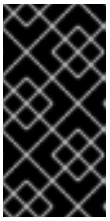
- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

14.6.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

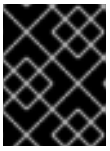
注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

14.6.9.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths": ...}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 OpenShift Cluster Manager から取得したプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

14.6.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

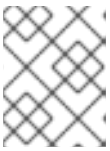
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定マップを生成して、追加の CA 証明書を保存します

user-ca-bundle という名前の設定マップを生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

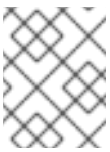
14.6.10. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

14.6.11. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

5. コントロールプレーンマシンおよび compute machineSets を定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- MachineSet ファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

14.6.12. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

14.6.12.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表14.51 Cluster Network Operator 設定オブジェクト

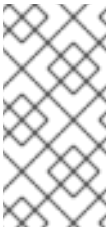
フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

フィールド	タイプ	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表14.52 defaultNetwork オブジェクト

フィールド	タイプ	説明

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表14.53 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	タイプ	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表14.54 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表14.55 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表14.56 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

14.6.13. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。

重要

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

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

14.6.14. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

14.6.15. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

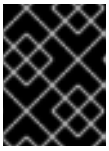
ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**
4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。

重要

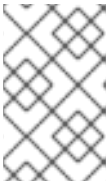
RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設

定に適した場所にあるストレージを動的にプロビジョニングします。

7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。



注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
      - **install-config.yaml** ファイルで指定したデータストアを選択します。

   f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

   g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。

重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。

g. **Customize hardware** タブで、**VM Options → Advanced** をクリックします。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。

  i. 静的 IP 設定を行います。

  ```
  $ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
  nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
  ```

  コマンドの例

  ```
  $ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
  nameserver=8.8.8.8"
  ```

  ii. vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

  ```
  $ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
  kargs=${IPCFG}"
  ```

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。VM の CPU とメモリーの予約に次の値があることを確認してください。

  - メモリー予約値は、設定されたメモリーサイズと同じである必要があります。
  - CPU 予約値は、低レイテンシー仮想 CPU の数に、実際に測定された物理 CPU 速度で乗算した数を最低でも指定する必要があります。

- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで、スチールクロックアカウンティング (**stealclock.enable**) に使用できるパラメーターのリストを検索します。利用可能な場合は、その値を **TRUE** に設定します。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。

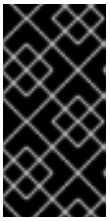
- **設定パラメーターの追加** をクリックします。以下のパラメーター名および値を定義します。

  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

9. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

14.6.16. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。
   g. **Customize hardware** タブで、**VM Options → Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

2. 継続してクラスター用の追加のコンピュートマシンを作成します。

14.6.17. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。

重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成

一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。

- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。

- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

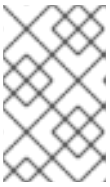
1 パーティションを設定する必要のあるディスクのストレージデバイス名。

2 データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小

さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

14.6.18. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。

注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

14.6.19. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

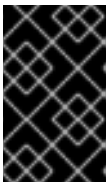
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

14.6.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

14.6.21. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.21.0
master-1 Ready    master  63m  v1.21.0
master-2 Ready    master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                 CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
```

```
master-2  Ready    master  74m  v1.21.0
worker-0  Ready    worker  11m  v1.21.0
worker-1  Ready    worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

14.6.21.1. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
```

```
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

14.6.21.2. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

14.6.21.3. イメージレジストリーストレージの設定

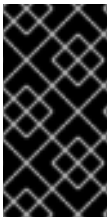
イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.6.21.3.1. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
   3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   4. 永続ボリューム要求 (PVC) のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

14.6.22. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
```

```
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

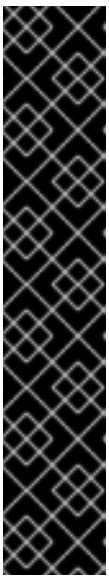
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                  NAME                                        READY   STATUS
```

```
RESTARTS   AGE
openshift-apiserver-operator      openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running     1          9m
openshift-apiserver               apiserver-67b9g                                 1/1     Running     0
3m
openshift-apiserver               apiserver-ljcmx                                 1/1     Running     0
1m
openshift-apiserver               apiserver-z25h4                                 1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

14.6.23. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

14.6.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

14.6.25. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

14.7. ネットワークが制限された環境での VSPHERE へのクラスターのインストール

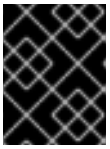
OpenShift Container Platform 4.8 では、インストールリリースコンテンツの内部ミラーを作成して、クラスターをネットワークが制限された環境で VMware vSphere インフラストラクチャーにインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

14.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- OpenShift Container Platform インストーラーは、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。
- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

14.7.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

14.7.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

14.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用し

ます。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

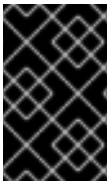
14.7.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.57 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.7.5. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表14.58 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.59 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.60 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

14.7.6. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例14.16 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.Add NewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.O bjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		stControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.MarkAsTemplate VirtualMachine.Provisioning.DeployTemplate
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例14.17 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource"

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例14.18 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。ネットワークが制限された環境の仮想マシンは、ノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理できるように vCenter にアクセスできる必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。

- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.61 必要な DNS レコード

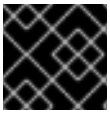
コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

14.7.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

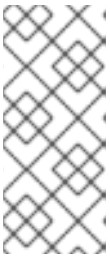
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

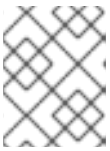
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

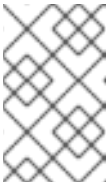
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

14.7.8. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
```

```
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

14.7.9. ネットワークが制限されたインストール用の RHCOS イメージの作成

Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された VMware vSphere 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリースト上に置かれます。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.8 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere**イメージをダウンロードします。
4. ダウンロードしたイメージを、bastion サーバーからアクセス可能な場所にアップロードします。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

14.7.10. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

前提条件

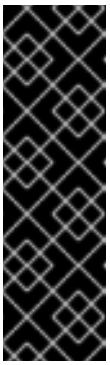
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得する。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、これをアクセス可能な場所にアップロードする。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。
      iii. vCenter インスタンスの名前を指定します。
      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。

インストールプログラムは vCenter インスタンスに接続します。

v. 接続する vCenter インスタンスにあるデータセンターを選択します。

vi. 使用するデフォルトの vCenter データストアを選択します。

vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xiii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルで **platform.vsphere.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
  vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
vmware.x86_64.ova?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

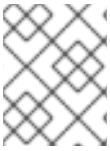
5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

14.7.10.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

14.7.10.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表14.62 必須パラメーター

パラメーター	説明	値

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

14.7.10.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表14.63 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
	
	複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

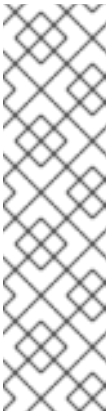
14.7.10.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表14.64 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

14.7.10.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表14.65 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列

パラメーター	説明	値
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

14.7.10.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表14.66 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

14.7.10.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
```

```
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster 8
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name 9
    apiVIP: api_vip
    ingressVIP: ingress_vip
    clusterOSImage: http://mirror.example.com/images/rhcos-48.83.202103221318-0-
vmware.x86_64.ova 10
fips: false
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 11
sshKey: 'ssh-ed25519 AAAA...'
additionalTrustBundle: | 12
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 13
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

9 OpenShift Container Platform クラスターをインストールする vSphere クラスター。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

10 bastion サーバーからアクセス可能な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所。

11 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

12 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

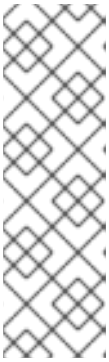
13 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

14.7.10.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
```

```
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

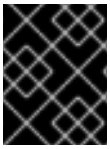
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

14.7.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

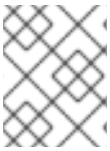
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

+



注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

+

重要

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

+

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

14.7.12. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。

PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

14.7.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

14.7.14. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

14.7.15. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

14.7.15.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

14.7.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

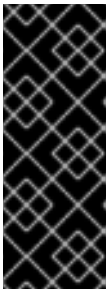
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.7.15.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

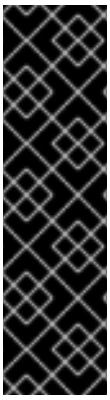
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



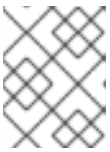
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

14.7.16. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

14.7.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

14.7.18. 次のステップ

- クラスターをカスタマイズします。

- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。

14.8. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを制限されたネットワークでプロビジョニングする VMware vSphere インフラストラクチャーにインストールできます。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

14.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

  重要

  インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- クラスターの 永続ストレージ をプロビジョニングしている。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- インストールを完了するには、vSphere ホストに Red Hat Enterprise Linux CoreOS(RHCOS) OVA をアップロードする必要があります。このプロセスを完了するマシンには、vCenter および ESXi ホストのポート 443 にアクセスできる必要があります。ポート 443 にアクセスできることを確認している。

- ファイアウォールを使用する場合は、ポート 443 にアクセスできることを管理者に確認している。インストールを成功させるには、コントロールプレーンノードがポート 443 で vCenter および ESXi ホストに到達できる必要があります。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

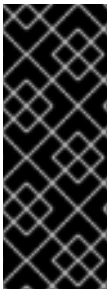
プロキシーを設定する場合は、このサイト一覧も確認してください。

14.8.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

14.8.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

14.8.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

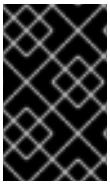
14.8.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表14.67 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。
オプション: Networking(NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) は OpenShift Container Platform 4.6 および NSX-T 3.x+ で認定されています。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

14.8.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

14.8.5.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表14.68 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

重要

すべての仮想マシンは、インストーラーと同じデータストアおよびフォルダーになければなりません。

14.8.5.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表14.69 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

14.8.5.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

14.8.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

14.8.5.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表14.70 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表14.71 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表14.72 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** - **00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00** - **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** - **00:50:56:FF:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

14.8.5.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

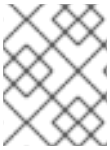
以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定す

るベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表14.73 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>. <base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>. <base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps. <cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>. <base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>. <cluster_name>. <base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コンピュートマシン	**<worker><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

14.8.5.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例14.19 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
```

```
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例14.20 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
```

```
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

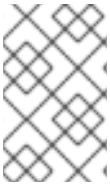
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

14.8.5.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

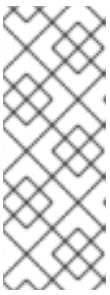
注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表14.74 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。

- 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表14.75 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

14.8.5.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例14.21 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 (4)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
```

```
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

14.8.6. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

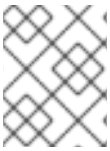
前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

14.8.7. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

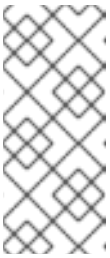
   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```



注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

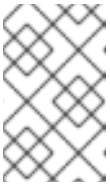
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

14.8.8. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

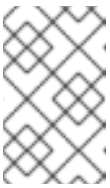
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/.ssh/id_ed25519 などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

14.8.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

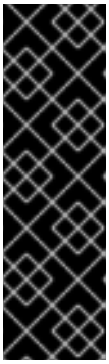
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得する。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

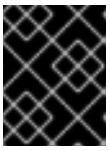
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

14.8.9.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
```

```
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
additionalTrustBundle: | 18
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 19
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

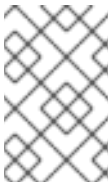
15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

19 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

14.8.9.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対す

る呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

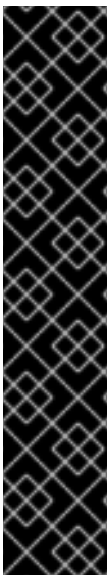
注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

14.8.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

14.8.11. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定する必要があります。

手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-worker-chrony 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 3
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst 4
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

1 2 コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を

使用します。

3 マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc <mc-name> -o yaml** で YAML ファイルを確認できます。

4 DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

- クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
- クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-chrony.yaml
```

14.8.12. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

14.8.13. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**
4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

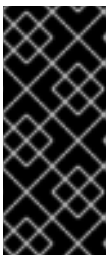
```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設

定に適した場所にあるストレージを動的にプロビジョニングします。

7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。

注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

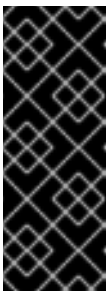
   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
      - **install-config.yaml** ファイルで指定したデータストアを選択します。

   f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

   g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。

g. **Customize hardware** タブで、**VM Options → Advanced** をクリックします。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。

  i. 静的 IP 設定を行います。

  ```
  $ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
  nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
  ```

  コマンドの例

  ```
  $ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
  nameserver=8.8.8.8"
  ```

  ii. vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

  ```
  $ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
  kargs=${IPCFG}"
  ```

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。VM の CPU とメモリーの予約に次の値があることを確認してください。

  - メモリー予約値は、設定されたメモリーサイズと同じである必要があります。
  - CPU 予約値は、低レイテンシー仮想 CPU の数に、実際に測定された物理 CPU 速度で乗算した数を最低でも指定する必要があります。

- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで、スチールクロックアカウンティング (**stealclock.enable**) に使用できるパラメーターのリストを検索します。利用可能な場合は、その値を **TRUE** に設定します。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。

- **設定パラメーターの追加** をクリックします。以下のパラメーター名および値を定義します。

  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

9. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

14.8.14. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware** を選択します。
   g. **Customize hardware** タブで、**VM Options → Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

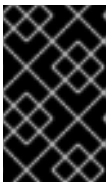
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

14.8.15. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。



重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成

一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。

- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。

- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小

さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

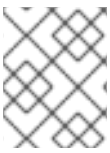
```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

14.8.16. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。



注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

14.8.17. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにイ

ンストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

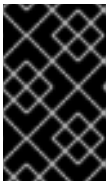
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

14.8.18. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラス

ターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

14.8.19. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

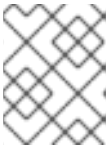
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

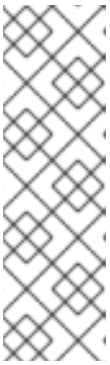
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                 CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

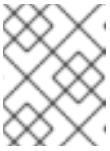
```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
-
```

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.21.0
master-1  Ready     master  73m  v1.21.0
master-2  Ready     master  74m  v1.21.0
worker-0  Ready     worker  11m  v1.21.0
worker-1  Ready     worker  11m  v1.21.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

14.8.20. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
```

```
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

14.8.20.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

14.8.20.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

14.8.20.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。

- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

14.8.20.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

14.8.20.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
   3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   4. 永続ボリューム要求 (PVC) のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

14.8.21. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
```

```
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

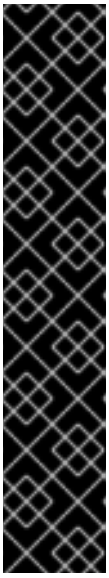
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running   0
3m
openshift-apiserver             apiserver-ljcmx                             1/1     Running   0
1m
openshift-apiserver             apiserver-z25h4                             1/1     Running   0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running   0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```
(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

4. Cluster registration ページでクラスターを登録します。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

14.8.22. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。

2. 永続ボリュームのクローンを作成します。

3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

14.8.23. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

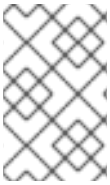
- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

14.8.24. 次のステップ

- クラスターをカスタマイズします。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

14.9. インストーラーでプロビジョニングされるインフラストラクチャーを使用する VSPHERE のクラスターのアンインストール

インストーラーでプロビジョニングされるインフラストラクチャーを使用して、VMware vSphere インスタンスにデプロイしたクラスターを削除できます。

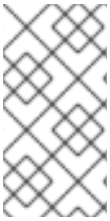


注記

openshift-install destroy cluster コマンドを実行して OpenShift Container Platform をアンインストールしても、vSphere ボリュームは自動的に削除されません。クラスター管理者は、vSphere ボリュームを手動で検索し、それらを削除する必要があります。

14.9.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

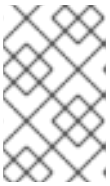
- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

14.10. VSPHERE PROBLEM DETECTOR OPERATOR の使用

14.10.1. vSphere Problem Detector Operator について

vSphere Problem Detector Operator は、一般的なインストールおよびストレージに関連する正しくない設定の問題について vSphere にデプロイされたクラスターをチェックします。

Operator は **openshift-cluster-storage-operator** namespace で実行され、Cluster Storage Operator がクラスターが vSphere にデプロイされたことを検知すると Cluster Storage Operator によって起動します。vSphere Problem Detector Operator は vSphere vCenter Server と通信して、クラスター内の仮想マシン、デフォルトのデータストア、および vSphere vCenter Server 設定についての他の情報を判別します。Operator は Cloud Credential Operator からの認証情報を使用して vSphere に接続します。

Operator は以下のスケジュールに基づいてチェックを実行します。

- チェックは 8 時間ごとに実行されます。

- チェックに失敗すると、Operator は 1 分、2 分、4 分、8 分などの間隔でチェックを再び実行します。Operator は、8 時間を最大の間隔とし、その範囲内で間隔を 2 倍にします。
- すべてのチェックに合格すると、スケジュールは 8 時間の間隔に戻ります。

Operator は、障害の発生後にチェックの頻度を増加させ、Operator が障害状態が修復された直後に正常な状態を報告できるようにします。Operator を手動で実行し、トラブルシューティングについての情報をすぐに確認できます。

14.10.2. vSphere Problem Detector Operator チェックの実行

vSphere Problem Detector Operator のチェックを実行するスケジュールを上書きし、チェックを即時に実行できます。

vSphere Problem Detector Operator は 8 時間ごとにチェックを自動的に実行します。ただし、Operator が起動すると、チェックがすぐに実行されます。Operator は、Cluster Storage Operator の起動時に Cluster Storage Operator によって起動し、クラスターが vSphere で実行されているかどうかを判別します。チェックをすぐに実行するには、vSphere Problem Detector Operator を **0** にスケーリングしてから、**1** に戻し、vSphere Problem Detector Operator が再起動できるようにします。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. Operator を **0** にスケーリングします。

```
$ oc scale deployment/vsphere-problem-detector-operator --replicas=0 \
    -n openshift-cluster-storage-operator
```

   デプロイメントがすぐにゼロにスケーリングされない場合、以下のコマンドを実行して Pod の終了を待機します。

```
$ oc wait pods -l name=vsphere-problem-detector-operator \
    --for=delete --timeout=5m -n openshift-cluster-storage-operator
```

2. Operator を **1** にスケーリングします。

```
$ oc scale deployment/vsphere-problem-detector-operator --replicas=1 \
    -n openshift-cluster-storage-operator
```

3. 古いリーダーロックを削除し、Cluster Storage Operator の新規リーダー選択を加速します。

```
$ oc delete -n openshift-cluster-storage-operator \
    cm vsphere-problem-detector-lock
```

検証

- vSphere Problem Detector Operator によって生成されるイベントまたはログを表示します。イベントまたはログに最新のタイムスタンプがあることを確認します。

14.10.3. vSphere Problem Detector Operator からのイベントの表示

vSphere Problem Detector Operator が設定チェックを実行した後に、コマンドラインまたは OpenShift Container Platform Web コンソールから表示できるイベントを作成します。

手順

- コマンドラインを使用してイベントを表示するには、以下のコマンドを実行します。

```
$ oc get event -n openshift-cluster-storage-operator \
    --sort-by={.metadata.creationTimestamp}
```

出力例

```
16m     Normal   Started                pod/vsphere-problem-detector-operator-xxxxx      Started
container vsphere-problem-detector
16m     Normal   Created                pod/vsphere-problem-detector-operator-xxxxx      Created
container vsphere-problem-detector
16m     Normal   LeaderElection         configmap/vsphere-problem-detector-lock  vsphere-
problem-detector-operator-xxxxx became leader
```

- OpenShift Container Platform Web コンソールを使用してイベントを表示するには、**Home → Events** に移動し、**Project** メニューから **openshift-cluster-storage-operator** を選択します。

14.10.4. vSphere Problem Detector Operator からのログの表示

vSphere Problem Detector Operator が設定チェックを実行した後に、コマンドラインまたは OpenShift Container Platform Web コンソールから表示できるログレコードを作成します。

手順

- コマンドラインを使用してログを表示するには、以下のコマンドを実行します。

```
$ oc logs deployment/vsphere-problem-detector-operator \
    -n openshift-cluster-storage-operator
```

出力例

```
I0108 08:32:28.445696       1 operator.go:209] ClusterInfo passed
I0108 08:32:28.451029       1 datastore.go:57] CheckStorageClasses checked 1 storage
classes, 0 problems found
I0108 08:32:28.451047       1 operator.go:209] CheckStorageClasses passed
I0108 08:32:28.452160       1 operator.go:209] CheckDefaultDatastore passed
I0108 08:32:28.480648       1 operator.go:271] CheckNodeDiskUUID:<host_name> passed
I0108 08:32:28.480685       1 operator.go:271] CheckNodeProviderID:<host_name> passed
```

- OpenShift Container Platform Web コンソールで Operator ログを表示するには、以下の手順を実行します。

  a. **Workloads → Pods** に移動します。

  b. **Projects** メニューから **openshift-cluster-storage-operator** を選択します。from the

  c. **vsphere-problem-detector-operator** Pod のリンクをクリックします。

  d. **Pod details** ページの **Logs** タブをクリックしてログを表示します。

14.10.5. vSphere Problem Detector Operator によって実行される設定チェック

以下の表は、vSphere Problem Detector Operator が実行する設定チェックを特定します。一部のチェックでは、クラスターの設定を確認します。他のチェックは、クラスター内の各ノードの設定を確認します。

表14.76 クラスター設定チェック

名前	説明
CheckDefaultDatastore	vSphere 設定のデフォルトのデータストア名が動的プロビジョニングで使用できる程度の短い名前であることを確認します。 このチェックに失敗した場合は、以下が予想されます。 ● **systemd** は、**Failed to set up mount unit: Invalid argument** などのエラーのログをジャーナルに記録します。 ● **systemd** は、仮想マシンがシャットダウンされていないか、ノードから すべての Pod をドレイン (解放) せずに再起動されている場合はボリュームをアンマウントしません。 このチェックに失敗した場合は、デフォルトのデータストアのより短い名前で vSphere を再設定します。
CheckFolderPermissions	デフォルトのデータストアでボリュームを一覧表示するパーミッションを検証します。このパーミッションは、ボリュームの作成に必要です。Operator は、/ および **/kubevols** ディレクトリーを一覧表示してパーミッションを検証します。ルートディレクトリーが存在する必要があります。これは、チェックの実行時に **/kubevols ディレクトリー** が存在しない場合に許可されます。**/kubevols** ディレクトリーは、このディレクトリーが存在しない場合に、データストアが動的プロビジョニングで使用される際に作成されます。 このチェックに失敗した場合は、OpenShift Container Platform のインストール時に指定された vCenter アカウントに必要なパーミッションを確認します。
CheckStorageClasses	以下を確認してください。 ● このストレージクラスによってプロビジョニングされる各永続ボリュームへの完全修飾パスは 255 文字未満です。 ● ストレージクラスがストレージポリシーを使用する場合、ストレージクラスは 1 つのポリシーのみを使用し、そのポリシーを定義する必要があります。
CheckTaskPermissions	最新のタスクおよびデータストアを一覧表示するパーミッションを検証します。
ClusterInfo	vSphere vCenter からクラスターバージョンおよび UUID を収集します。

表14.77 ノード設定チェック

名前	説明
CheckNodeDiskUUID	すべての vSphere 仮想マシンが **disk.enableUUID=TRUE** で設定されていることを確認します。 このチェックに失敗した場合は、Red Hat ナレッジベースソリューションの How to check 'disk.EnableUUID' parameter from VM in vSphere を参照してください。
CheckNodeProviderID	すべてのノードが vSphere vCenter の **ProviderID** で設定されていることを確認します。以下のコマンドからの出力に各ノードのプロバイダー ID が含まれていない場合に、このチェックに失敗します。 `$ oc get nodes -o custom-columns=NAME:.metadata.name,PROVIDER_ID:.spec.providerID,UUID:.status.nodeInfo.systemUUID` このチェックに失敗した場合は、クラスター内の各ノードのプロバイダー ID の設定方法について、vSphere の製品ドキュメントを参照してください。
CollectNodeESXiVersion	ノードを実行する ESXi ホストのバージョンを報告します。
CollectNodeHWVersion	ノードの仮想マシンのハードウェアバージョンを報告します。

14.10.6. ストレージクラス設定チェックについて

vSphere ストレージを使用する永続ボリュームの名前は、データストア名とクラスター ID に関連します。

永続ボリュームが作成されると、**systemd** は永続ボリュームのマウントユニットを作成します。**systemd** プロセスには、永続ボリュームに使用される VDMK ファイルへの完全修飾パスの長さについて 255 文字の制限があります。

完全修飾パスは、**systemd** および vSphere の命名規則に基づいています。命名規則では、以下のパターンを使用します。

```
/var/lib/kubelet/plugins/kubernetes.io/vsphere-volume/mounts/[<datastore>] 00000000-0000-0000-0000-000000000000/<cluster_id>-dynamic-pvc-00000000-0000-0000-0000-000000000000.vmdk
```

- 命名規則では、255 文字制限の内の 205 文字が必要です。
- データストア名とクラスター ID はデプロイメントで判別されます。
- データストア名とクラスター ID は前述のパターンに代入されます。次に、パスは特殊文字をエスケープできるように **systemd-escape** コマンドで処理されます。たとえば、ハイフン文字ではエスケープ後に 4 文字を使用します。エスケープされた値は **\x2d** になります。
- **systemd-escape** で処理した後に、**systemd** が VDMK ファイルへの完全修飾パスにアクセスできるようにするには、パスの長さが 255 文字未満である必要があります。

14.10.7. vSphere Problem Detector Operator のメトリクス

vSphere Problem Detectorーリングスタックで使用される以下のメトリクスを公開します。

表14.78 vSphere Problem Detector Operator によって公開されるメトリクス

名前	説明
vsphere_cluster_check_total	vSphere Problem Detector Operator が実行したクラスターレベルのチェックの累積数です。この数には、成功と失敗の両方が含まれます。
vsphere_cluster_check_errors	vSphere Problem Detector Operator が実行したクラスターレベルのチェックの失敗したチェック数です。たとえば、値 **1** は 1 つのクラスターレベルのチェックが失敗したことを示します。
vsphere_esxi_version_total	特定のバージョンを持つ ESXi ホストの数ホストが複数のノードを実行する場合は、ホストが 1 回のみカウントされることに注意してください。
vsphere_node_check_total	vSphere Problem Detector Operator が実行したノードレベルのチェックの累積数です。この数には、成功と失敗の両方が含まれます。
vsphere_node_check_errors	vSphere Problem Detector Operator が実行したノードレベルのチェックの失敗したチェック数です。たとえば、値 **1** は 1 つのノードレベルのチェックが失敗したことを示します。
vsphere_node_hw_version_total	特定のハードウェアバージョンを持つ vSphere ノードの数。
vsphere_vcenter_info	vSphere vCenter サーバーに関する情報

14.10.8. 関連情報

- モニターリングの概要

第15章 VMC へのインストール

15.1. VMC へのインストールの準備

15.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- クラスターが必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

15.1.2. VMC に OpenShift Container Platform をインストールする方法の選択

インストーラーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して、VMC に OpenShift Container Platform をインストールすることができます。デフォルトのインストールタイプは、インストーラーでプロビジョニングされるインフラストラクチャーを使用します。この場合、インストールプログラムがクラスターの基礎となるインフラストラクチャーをプロビジョニングします。OpenShift Container Platform はユーザーが独自にプロビジョニングするインフラストラクチャーにインストールすることもできます。インストールプログラムがプロビジョニングするインフラストラクチャーを使用しない場合は、クラスターリソースをユーザー自身で管理し、維持する必要があります。

インストーラーおよびユーザーによってプロビジョニングされるインストールプロセスの詳細は、インストールプロセス を参照してください。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自に提供するインフラストラクチャーでクラスターをインストールするには、VMC プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

15.1.2.1. インストーラーでプロビジョニングされるインフラストラクチャーでの VMC への OpenShift Container Platform のインストール

インストーラーでプロビジョニングされるインフラストラクチャーにより、インストールプログラムは OpenShift Container Platform で必要なリソースのプロビジョニングを事前に設定し、自動化することができます。

- **クラスターの VMC へのインストール**: インストーラーでプロビジョニングされるインフラストラクチャーのインストールをカスタマイズせずに使用して、VMC に OpenShift Container Platform をインストールできます。
- **カスタマイズによる VMC へのクラスターのインストール**: インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトのカスタマイズオプションのインストールを使用して、VMC に OpenShift Container Platform をインストールできます。
- **ネットワークのカスタマイズによる VMC へのクラスターのインストール**: ネットワークのカスタマイズを使用して、インストーラーでプロビジョニングされる VMC インフラストラク

チャーに OpenShift Container Platform をインストールできます。インストール時に OpenShift Container Platform ネットワーク設定をカスタマイズすることで、クラスターが既存の IP アドレスの割り当てと共存でき、ネットワーク要件に準拠することができます。

- **ネットワークが制限された環境での VMC へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを作成して、ネットワークが制限された環境で VMC インフラストラクチャーにクラスターをインストールできます。この方法を使用して、インターネット上に表示されない内部ネットワークに OpenShift Container Platform をデプロイすることができます。

15.1.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの VMC への OpenShift Container Platform のインストール

ユーザーによってプロビジョニングされるインフラストラクチャーでは、ユーザーは OpenShift Container Platform に必要なすべてのリソースをプロビジョニングする必要があります。

- **ユーザーによってプロビジョニングされるインフラストラクチャーでの VMC へのクラスターのインストール**: 独自にプロビジョニングする VMC インフラストラクチャーに OpenShift Container Platform をインストールできます。
- **カスタマイズされたネットワークを使用したユーザーによってプロビジョニングされるインフラストラクチャーでの VMC へのクラスターのインストール**: カスタマイズされたネットワーク設定オプションを使用して独自にプロビジョニングする VMC インフラストラクチャーに OpenShift Container Platform をインストールできます。
- **ネットワークが制限された環境でユーザーによってプロビジョニングされるインフラストラクチャーでの VMC へのクラスターのインストール**: ネットワークが制限された環境で独自にプロビジョニングする VMC インフラストラクチャーに、OpenShift Container Platform をインストールできます。

15.1.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.1 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.1.4. インストーラーでプロビジョニングされるインフラストラクチャーでの VMC への OpenShift Container Platform のアンインストール

- **インストーラーでプロビジョニングされるインフラストラクチャーを使用した VMC のクラスターのアンインストール**: インストーラーでプロビジョニングされるインフラストラクチャーを使用した VMC インフラストラクチャーにデプロイされたクラスターを削除できます。

15.2. クラスターの VMC へのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイし、VMware vSphere にインストールできます。

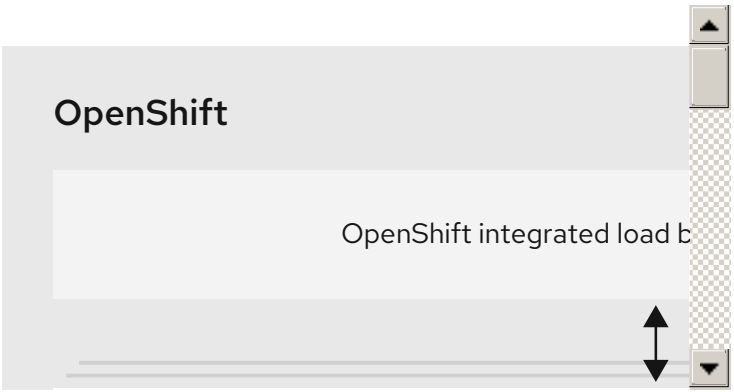
OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要なリソースのデプロイおよび管理プロセスを自動化します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.2.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



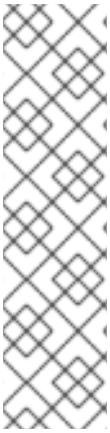
OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- DHCP 範囲外にある 2 つの IP アドレスを割り当て、それらを逆引き DNS レコードで設定します。
  - 割り当てられた IP アドレスをポイントする **api.<cluster_name>.<base_domain>** の DNS レコード。
  - 割り当てられた IP アドレスをポイントする ***.apps.<cluster_name>.<base_domain>** の DNS レコード。
- 以下のファイアウォールルールを設定します。
  - OpenShift Container Platform コンピュートネットワークとインターネット間の ANY:ANY ファイアウォールルール。これは、コンテナーイメージをダウンロードするためにノードおよびアプリケーションによって使用されます。
  - ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
  - OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。
- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
  - OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。
  - ベース DNS 名 (**companyname.com** など)。
  - デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。
  - 以下の vCenter 情報:
    - vCenter ホスト名、ユーザー名、およびパスワード
    - データセンター名 (**SDDC-Datacenter** など)
    - クラスター名 (**Cluster-1** など)
    - ネットワーク名
    - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール



注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.2.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノードを使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ
- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.2.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。

- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

15.2.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

15.2.4. VMware vSphere インフラストラクチャーの要件

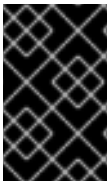
使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.2 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.2.5. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表15.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)

プロトコル	ポート	説明
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

15.2.6. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

> **例15.1 vSphere API でのインストールに必要なロールと権限**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.Edit**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		Device VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.MarkAsTemplate VirtualMachine.Provisioning.DeployTemplate
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例15.2 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Configuration"."Add or remove device "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit

ロールの vSphere オブジェクト	必要になる場合	Inventory"."Create from existing vCenter GUI で必要な権限 "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例15.3 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.6 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

15.2.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

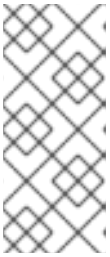
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

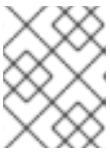
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

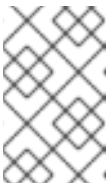
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

15.2.8. インストールプログラムの取得

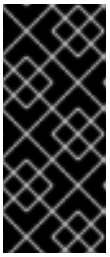
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

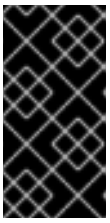
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.2.9. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。
2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

15.2.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

c. vCenter インスタンスの名前を指定します。

d. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。

e. 接続する vCenter インスタンスにあるデータセンターを選択します。

f. 使用するデフォルトの vCenter データストアを選択します。

注記

データストアとクラスター名は 60 文字を超えることができません。そのため、組み合わせた文字列の長さが 60 文字の制限を超えないようにしてください。

g. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

h. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

i. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

j. クラスター Ingress に設定した仮想 IP アドレスを入力します。

k. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

l. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

注記

データストアとクラスター名は 60 文字を超えることができません。そのため、組み合わせた文字列の長さが 60 文字の制限を超えないようにしてください。

m. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

重要

VMC 環境でホストされる bastion からの **openshift-install** コマンドを使用します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

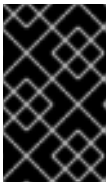
```
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

15.2.11. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

15.2.12. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.2.13. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

15.2.13.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

15.2.13.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.2.13.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

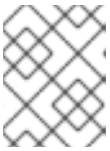
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

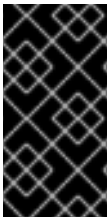
```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

15.2.13.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

3 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

4 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

15.2.14. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

15.2.15. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちま

す。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

15.2.16. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.2.17. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

15.3. カスタマイズによる VMC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイして、インストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。

OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要なリソースのデプロイおよび管理プロセスを自動化します。

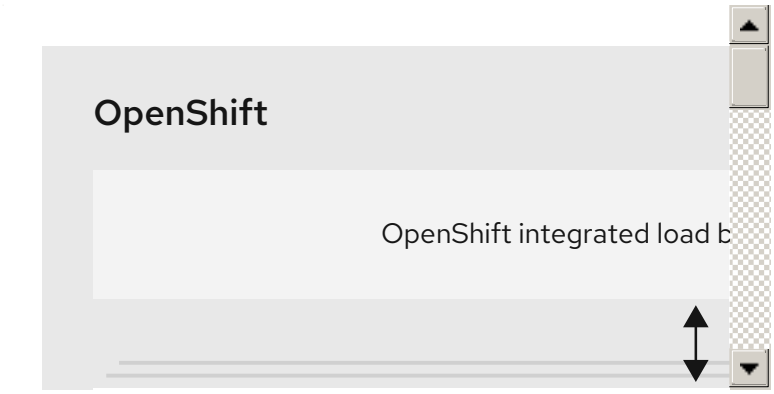
OpenShift Container Platform インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.3.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- DHCP 範囲外にある 2 つの IP アドレスを割り当て、それらを逆引き DNS レコードで設定します。
    - 割り当てられた IP アドレスをポイントする **api.<cluster_name>.<base_domain>** の DNS レコード。
    - 割り当てられた IP アドレスをポイントする ***.apps.<cluster_name>.<base_domain>** の DNS レコード。
- 以下のファイアウォールルールを設定します。
    - OpenShift Container Platform コンピュートネットワークとインターネット間の ANY:ANY ファイアウォールルール。これは、コンテナーイメージをダウンロードするためにノードおよびアプリケーションによって使用されます。
    - ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
    - OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。
- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
    - OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。
    - ベース DNS 名 (**companyname.com** など)。
    - デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。

- 以下の vCenter 情報:
  - vCenter ホスト名、ユーザー名、およびパスワード
  - データセンター名 (**SDDC-Datacenter** など)
  - クラスター名 (**Cluster-1** など)
  - ネットワーク名
  - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール

注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.3.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノードを使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ

- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.3.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

15.3.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

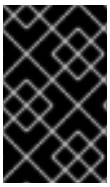
15.3.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.7 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.3.5. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表15.8 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.9 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.10 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

15.3.6. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限

OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例15.4 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.AddExistingDisk **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例15.5 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new"

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例15.6 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.11 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

15.3.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

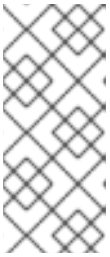
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

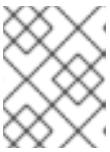
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

15.3.8. インストールプログラムの取得

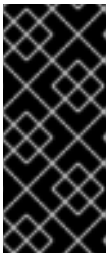
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.3.9. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

15.3.10. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

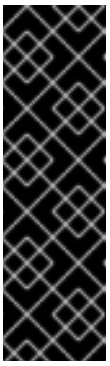
手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

      iii. vCenter インスタンスの名前を指定します。

      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

      v. 接続する vCenter インスタンスにあるデータセンターを選択します。

      vi. 使用するデフォルトの vCenter データストアを選択します。

      vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

      viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xiii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

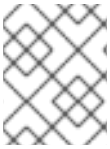
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

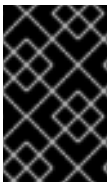
15.3.10.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

15.3.10.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表15.12 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.3.10.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表15.13 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

15.3.10.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表15.14 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列

パラメーター	説明	値
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

15.3.10.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表15.15 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列

パラメーター	説明	値
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

15.3.10.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表15.16 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数

パラメーター	説明	値
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

15.3.10.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
```

```
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name 9
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

9 OpenShift Container Platform クラスターをインストールする vSphere クラスター。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

15.3.10.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

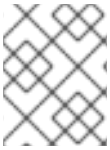
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

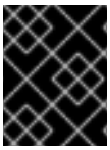


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

15.3.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

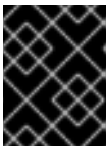
手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

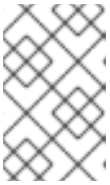
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



重要

VMC 環境でホストされる bastion からの **openshift-install** コマンドを使用します。



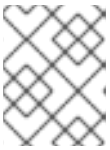
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

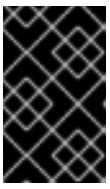


注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。

重要

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



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

15.3.12. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

15.3.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

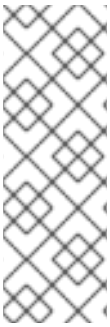
15.3.14. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

15.3.14.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

15.3.14.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.3.14.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

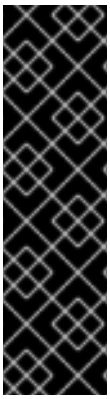
前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

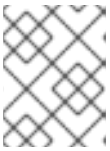
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります

す。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry    4.7                                 True        False         False      6h50m
```

15.3.14.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
4. 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

15.3.15. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

15.3.16. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

15.3.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.3.18. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

15.4. ネットワークのカスタマイズによる VMC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイし、カスタマイズされるネットワーク設定オプションと共にインストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。

OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要なリソースのデプロイおよび管理プロセスを自動化します。

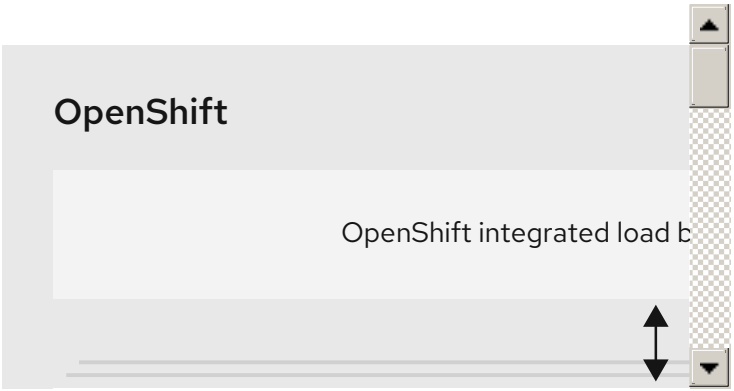
OpenShift Container Platform ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.4.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

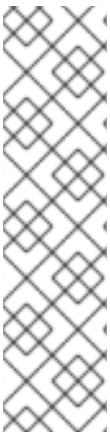
- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- DHCP 範囲外にある 2 つの IP アドレスを割り当て、それらを逆引き DNS レコードで設定します。
  - 割り当てられた IP アドレスをポイントする **api.<cluster_name>.<base_domain>** の DNS レコード。
  - 割り当てられた IP アドレスをポイントする ***.apps.<cluster_name>.<base_domain>** の DNS レコード。
- 以下のファイアウォールルールを設定します。
  - OpenShift Container Platform コンピュートネットワークとインターネット間の ANY:ANY ファイアウォールルール。これは、コンテナーイメージをダウンロードするためにノードおよびアプリケーションによって使用されます。
  - ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
  - OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。
- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
  - OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。

- ベース DNS 名 (**companyname.com** など)。
- デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。
- 以下の vCenter 情報:
  - vCenter ホスト名、ユーザー名、およびパスワード
  - データセンター名 (**SDDC-Datacenter** など)
  - クラスター名 (**Cluster-1** など)
  - ネットワーク名
  - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール



注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.4.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on

AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノードを使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ
- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.4.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

15.4.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。

- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

15.4.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.17 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.4.5. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表15.18 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.19 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.20 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

15.4.6. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限

OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例15.7 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.AddExistingDisk **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.EditDevice** **VirtualMachine.Config.Memory** **VirtualMachine.Config.RemoveDisk** **VirtualMachine.Config.Rename** **VirtualMachine.Config.ResetGuestInfo** **VirtualMachine.Config.Resource** **VirtualMachine.Config.Settings** **VirtualMachine.Config.UpgradeVirtualHardware** **VirtualMachine.Interact.GuestControl** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Interact.Reset** **VirtualMachine.Inventory.Create** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone** **VirtualMachine.Provisioning.MarkAsTemplate** **VirtualMachine.Provisioning.DeployTemplate**
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限

例15.8 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"

vSphere vCenter Datacenter ロールの vSphere オブジェクト	インストールプログラムが仮想マシンフォルダーを作成する場合	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Gues

ロールの vSphere オブジェクト	必要になる場合	t operating system management by VIX API" vCenter GUI で必要な権限
		"Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例15.9 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:

- 1 テンプレート
- 1 一時的ブートストラップノード
- 3 コントロールプレーンノード
- 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.21 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

15.4.7. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

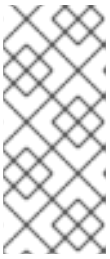
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

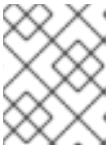
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

15.4.8. インストールプログラムの取得

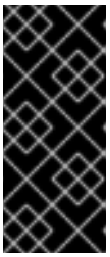
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

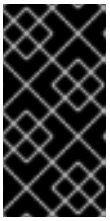
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.4.9. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

15.4.10. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。

   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

      iii. vCenter インスタンスの名前を指定します。

      iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

      v. 接続する vCenter インスタンスにあるデータセンターを選択します。

      vi. 使用するデフォルトの vCenter データストアを選択します。

      vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

      viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

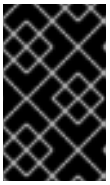
xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xiii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細は、インストール設定パラメーターのセクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

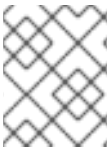


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

15.4.10.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

15.4.10.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表15.22 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.4.10.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表15.23 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

15.4.10.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表15.24 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列

パラメーター	説明	値
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

15.4.10.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表15.25 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列

パラメーター	説明	値
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

15.4.10.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表15.26 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数

パラメーター	説明	値
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

15.4.10.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
```

```
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name 9
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

9 OpenShift Container Platform クラスターをインストールする vSphere クラスター。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

15.4.10.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

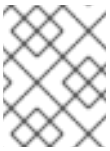
15.4.11. ネットワーク設定フェーズ

OpenShift Container Platform をインストールする前に、ネットワーク設定をカスタマイズできる 2 つのフェーズがあります。

フェーズ 1

マニフェストファイルを作成する前に、**install-config.yaml** ファイルで以下のネットワーク関連のフィールドをカスタマイズできます。

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  これらのフィールドの詳細は、**インストール設定パラメーター** を参照してください。



注記

優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

フェーズ 2

openshift-install create manifests を実行してマニフェストファイルを作成した後に、変更するフィールドのみでカスタマイズされた Cluster Network Operator マニフェストを定義できます。マニフェストを使用して、高度なネットワーク設定を指定できます。

フェーズ 2 で、**install-config.yaml** ファイルのフェーズ 1 で指定した値を上書きすることはできません。ただし、フェーズ 2 ではクラスターネットワークプロバイダーをさらにカスタマイズできます。

15.4.12. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

15.4.13. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

15.4.13.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表15.27 Cluster Network Operator 設定オブジェクト

フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	タイプ	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表15.28 defaultNetwork オブジェクト

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定

以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表15.29 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
```

```
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定
以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表15.30 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表15.31 policyAuditConfig object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。

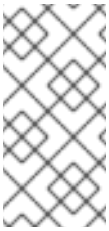
フィールド	タイプ	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表15.32 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。  **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

フィールド	タイプ	説明
proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値: kubeProxyConfig: proxyArguments: iptables-min-sync-period: - 0s

15.4.14. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

VMC 環境でホストされる bastion からの **openshift-install** コマンドを使用します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

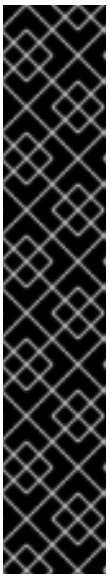
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

15.4.15. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

■

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

15.4.16. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.4.17. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

15.4.17.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

15.4.17.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

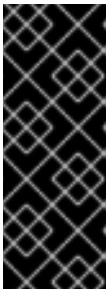
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.4.17.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

15.4.17.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
```

```
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
3. 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
4. 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1. カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

15.4.18. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。

3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

15.4.19. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

15.4.20. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.4.21. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

15.5. ネットワークが制限された環境での VMC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイし、これを制限されたネットワークの VMware vSphere インフラストラクチャーにインストールできます。

OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要な

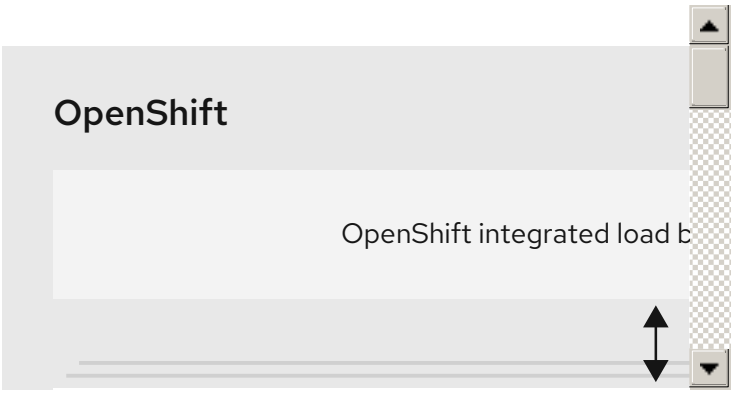
リソースのデプロイおよび管理プロセスを自動化します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.5.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- DHCP 範囲外にある 2 つの IP アドレスを割り当て、それらを逆引き DNS レコードで設定します。
  - 割り当てられた IP アドレスをポイントする **api.<cluster_name>.<base_domain>** の DNS レコード。
  - 割り当てられた IP アドレスをポイントする ***.apps.<cluster_name>.<base_domain>** の DNS レコード。
- 以下のファイアウォールルールを設定します。
  - ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
  - OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。
- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
  - OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。

- ベース DNS 名 (**companyname.com** など)。
- デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。
- 以下の vCenter 情報:
  - vCenter ホスト名、ユーザー名、およびパスワード
  - データセンター名 (**SDDC-Datacenter** など)
  - クラスター名 (**Cluster-1** など)
  - ネットワーク名
  - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール

注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.5.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノード

を使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

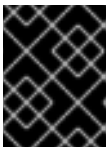
これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ
- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.5.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

15.5.3. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インス

トーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

15.5.3.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

15.5.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

15.5.5. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.33 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.5.6. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。

必要なネットワークポートに関する次の詳細を確認してください。

表15.34 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	仮想拡張可能 LAN (VXLAN)

プロトコル	ポート	説明
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.35 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.36 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

15.5.7. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例15.10 vSphere API でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
vSphere vCenter	Always	**CNS.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**
仮想マシンフォルダー	Always	**InventoryService.Tagging.ObjectAttachable** **Resource.AssignVMToPool** **VApp.Import** **VirtualMachine.Config.AddExistingDisk** **VirtualMachine.Config.AddNewDisk** **VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.DiskLease** **VirtualMachine.Config.Edit**

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		Device VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.MarkAsTemplate VirtualMachine.Provisioning.DeployTemplate
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	InventoryService.Tagging.ObjectAttachable Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease

ロールの vSphere オブジェクト	必要になる場合	vSphere API で必要な権限
		VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone VirtualMachine.Provisioning.DeployTemplate VirtualMachine.Provisioning.MarkAsTemplate Folder.Create Folder.Delete

例15.11 vCenter グラフィカルユーザーインターフェイス (GUI) でのインストールに必要なロールと権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere vCenter	Always	**Cns.Searchable "vSphere Tagging"."Assign or Unassign vSphere Tag" "vSphere Tagging"."Create vSphere Tag Category" "vSphere Tagging"."Create vSphere Tag" vSphere Tagging"."Delete vSphere Tag Category" "vSphere Tagging"."Delete vSphere Tag" "vSphere Tagging"."Edit vSphere Tag Category" "vSphere Tagging"."Edit vSphere Tag" Sessions."Validate session" "Profile-driven storage"."Profile-driven storage update" "Profile-driven storage"."Profile-driven storage view"**
vSphere vCenter Cluster	仮想マシンがクラスタールートに作成される場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**
vSphere vCenter リソースプール	既存のリソースプールが提供されている場合	**Host.Configuration."Storage partition configuration" Resource."Assign virtual machine to resource pool" VApp."Assign resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add new disk"**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
vSphere Datastore	Always	**Datastore."Allocate space" Datastore."Browse datastore" Datastore."Low level file operations" "vSphere Tagging"."Assign or Unassign vSphere Tag on Object"**
vSphere ポートグループ	Always	**Network."Assign network"**
仮想マシンフォルダー	Always	**"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename**

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		"Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit Inventory"."Create from existing" "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Mark as template" "Virtual machine".Provisioning."Deploy template"
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	"vSphere Tagging"."Assign or Unassign vSphere Tag on Object" Resource."Assign virtual machine to resource pool" VApp.Import "Virtual machine"."Change Configuration"."Add existing disk" "Virtual machine"."Change Configuration"."Add new disk" "Virtual machine"."Change

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Configuration"."Add or remove device" "Virtual machine"."Change Configuration"."Advanced configuration" "Virtual machine"."Change Configuration"."Set annotation" "Virtual machine"."Change Configuration"."Change CPU count" "Virtual machine"."Change Configuration"."Extend virtual disk" "Virtual machine"."Change Configuration"."Acquire disk lease" "Virtual machine"."Change Configuration"."Modify device settings" "Virtual machine"."Change Configuration"."Change Memory" "Virtual machine"."Change Configuration"."Remove disk" "Virtual machine"."Change Configuration".Rename "Virtual machine"."Change Configuration"."Reset guest information" "Virtual machine"."Change Configuration"."Change resource" "Virtual machine"."Change Configuration"."Change Settings" "Virtual machine"."Change Configuration"."Upgrade virtual machine compatibility" "Virtual machine".Interaction."Guest operating system management by VIX API" "Virtual machine".Interaction."Power off" "Virtual machine".Interaction."Power on" "Virtual machine".Interaction.Reset "Virtual machine"."Edit Inventory"."Create new" "Virtual machine"."Edit

ロールの vSphere オブジェクト	必要になる場合	vCenter GUI で必要な権限
		Inventory"."Create from existing "Virtual machine"."Edit Inventory"."Remove" "Virtual machine".Provisioning."Clone virtual machine" "Virtual machine".Provisioning."Deploy template" "Virtual machine".Provisioning."Mark as template" Folder."Create folder" Folder."Delete folder"

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例15.12 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用
vSphere 環境で vMotion を使用する場合は、OpenShift Container Platform クラスターをインストールする前に以下を考慮してください。

- OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。
  コンピュートおよびコントロールプレーンノードのアップタイムを確保するには、vMotion の VMware のベストプラクティスに従うことが推奨されます。また、VMware 非アフィニティールールを使用して、メンテナンス中またはハードウェアの問題時に OpenShift Container Platform の可用性を向上させることも推奨します。

  vMotion および anti-affinity ルールの詳細は、vMotion ネットワーク要件 および VM の非アフィニティールール に関する VMware vSphere のドキュメントを参照してください。

- Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

- 同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスをクラスターマシンに提供するように設定されていることを確認する必要があります。すべてのノードが同じ VLAN にある必要があります。2 日目の操作として 2 番目の VLAN を使用してクラスターをスケーリングすることはできません。ネットワークが制限された環境の仮想マシンは、ノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理できるように vCenter にアクセスできる必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

注記

クラスターの各 OpenShift Container Platform ノードは、DHCP を使用して検出可能な Network Time Protocol (NTP) サーバーにアクセスできることが推奨されます。NTP サーバーなしでインストールが可能です。ただし、非同期のサーバークロックによりエラーが発生しますが、NTP サーバーはこのエラーを阻止します。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.37 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

15.5.8. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

15.5.9. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

15.5.10. ネットワークが制限されたインストール用の RHCOS イメージの作成

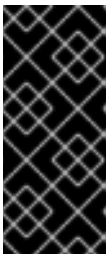
Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された VMware vSphere 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリースト上に置かれます。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.8 の最新リリースを選択します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere**イメージをダウンロードします。
4. ダウンロードしたイメージを、bastion サーバーからアクセス可能な場所にアップロードします。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

15.5.11. インストール設定ファイルの作成

VMware vSphere にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得する。
- Red Hat Enterprise Linux CoreOS (RHCOS) イメージを取得し、これをアクセス可能な場所にアップロードする。
- サブスクリプションレベルでサービスプリンシパルのパーミッションを取得する。

手順

1. **install-config.yaml** ファイルを作成します。
   a. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

iii. vCenter インスタンスの名前を指定します。

iv. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。

v. 接続する vCenter インスタンスにあるデータセンターを選択します。

vi. 使用するデフォルトの vCenter データストアを選択します。

vii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

viii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

ix. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

x. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xiii. Red Hat OpenShift Cluster Manager からプルシークレット を貼り付けます。

2. **install-config.yaml** ファイルで **platform.vsphere.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
```

```
 vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
vmware.x86_64.ova?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

   a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

   ```
   pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <mirror_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

   b. **additionalTrustBundle** パラメーターおよび値を追加します。

   ```
   additionalTrustBundle: |
    -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    -----END CERTIFICATE-----
   ```

   この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

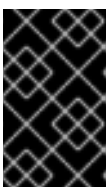
   c. 以下のようなイメージコンテンツリソースを追加します。

   ```
   imageContentSources:
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: quay.example.com/openshift-release-dev/ocp-release
   - mirrors:
     - <mirror_host_name>:5000/<repo_name>/release
     source: registry.example.com/ocp/release
   ```

   これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

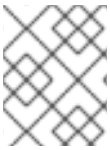


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

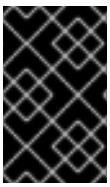
15.5.11.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

15.5.11.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表15.38 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	小文字いちぶハイフン (**-**) の文字列 (**dev** など)。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

pullSecret	Red Hat OpenShift Cluster Manager からプルシークレット を取得して、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージをダウンロードすることを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

15.5.11.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表15.39 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

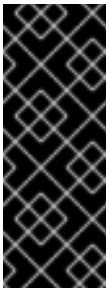
15.5.11.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表15.40 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **s390x** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。

パラメーター	説明	値
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このフィールドを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。
sshKey	クラスターマシンへのアクセスを認証するための単一または複数の SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	1 つ以上のキー。以下に例を示します。 sshKey: <key1> <key2> <key3>

15.5.11.1.4. 追加の VMware vSphere 設定パラメーター

追加の VMware vSphere 設定パラメーターは以下の表で説明されています。

表15.41 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列

パラメーター	説明	値
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

15.5.11.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表15.42 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数

パラメーター	説明	値
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの仮想ソケットの数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

15.5.11.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster (8)
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
```

```
    cluster: vsphere_cluster_name 9
    apiVIP: api_vip
    ingressVIP: ingress_vip
    clusterOSImage: http://mirror.example.com/images/rhcos-48.83.202103221318-0-
vmware.x86_64.ova 10
fips: false
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 11
sshKey: 'ssh-ed25519 AAAA...'
additionalTrustBundle: | 12
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 13
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

9 OpenShift Container Platform クラスターをインストールする vSphere クラスター。インストールプログラムは、vSphere クラスターの root リソースプールをデフォルトのリソースプールとして使用します。

10 bastion サーバーからアクセス可能な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所。

11 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコー

ドされたユーザー名およびパスワードを指定します。

12 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

13 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

15.5.11.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

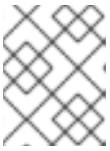
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。

3 プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。

4 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

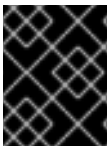
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

15.5.12. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターのデプロイメントを初期化します。

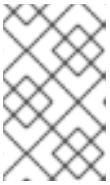
```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

VMC 環境でホストされる bastion からの **openshift-install** コマンドを使用します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

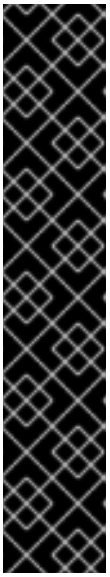
クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

出力例

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

注記

クラスターアクセスおよび認証情報の情報は、インストールが正常に実行される際に **<installation_directory>/.openshift_install.log** に出力されます。



重要

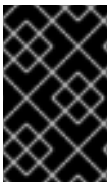
- インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。
- 24 時間証明書はクラ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

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

15.5.13. バイナリーのダウンロードによる OpenShift CLI のインストール

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

15.5.14. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.5.15. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

15.5.16. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

15.5.16.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

15.5.16.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.5.16.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

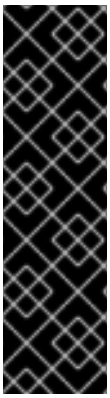
前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



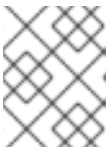
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

15.5.17. スチールクロックアカウンティング

デフォルトでは、インストールプログラムは、スチールクロックアカウンティングパラメーター (**stealclock.enabled**) を有効にせずに、クラスターの仮想マシンをプロビジョニングします。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。クラスターをデプロイメントしたら、vSphere Client を使用して、各仮想マシンでこのパラメーターを有効にします。

詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

15.5.18. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.5.19. 次のステップ

- クラスターをカスタマイズします。

- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。

15.6. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した VMC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイし、これをプロビジョニングされる VMware vSphere インフラストラクチャーにインストールできます。

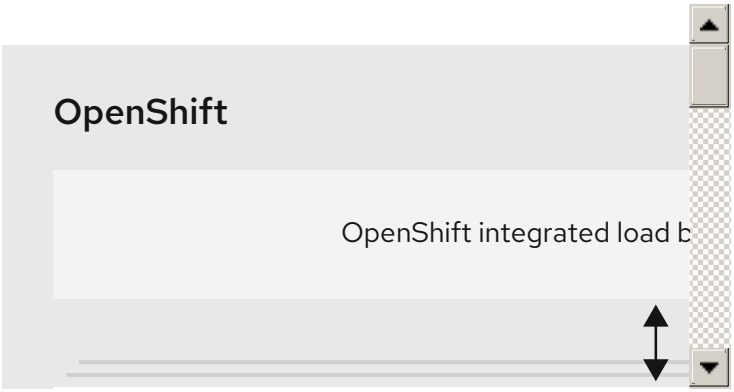
OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要なリソースのデプロイおよび管理プロセスを自動化します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.6.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- 以下のファイアウォールルールを設定します。
  - OpenShift Container Platform コンピュートネットワークとインターネット間の ANY:ANY ファイアウォールルール。これは、コンテナーイメージをダウンロードするためにノード

およびアプリケーションによって使用されます。

- ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
- OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。

- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
  - OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。
  - ベース DNS 名 (**companyname.com** など)。
  - デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。
  - 以下の vCenter 情報:
    - vCenter ホスト名、ユーザー名、およびパスワード
    - データセンター名 (**SDDC-Datacenter** など)
    - クラスター名 (**Cluster-1** など)
    - ネットワーク名
    - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール

注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.6.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノードを使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ
- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.6.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

15.6.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

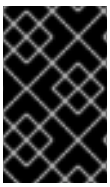
15.6.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.43 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.6.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

15.6.5.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表15.44 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

15.6.5.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表15.45 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

15.6.5.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

15.6.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

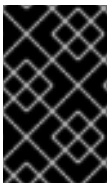
マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

15.6.5.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。



重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表15.46 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.47 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.48 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

15.6.5.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

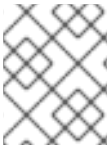
DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.49 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

15.6.5.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例15.13 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例15.14 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
```

```
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

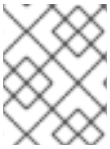
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

15.6.5.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.50 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

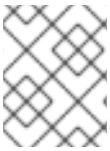
ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.51 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

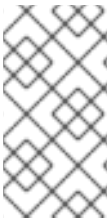


注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

15.6.5.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例15.15 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

15.6.6. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

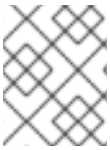
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

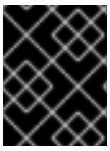


注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

15.6.7. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

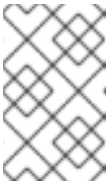
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

15.6.8. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

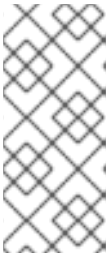
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

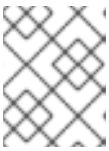
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

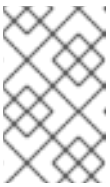
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

15.6.9. インストールプログラムの取得

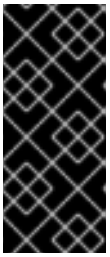
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

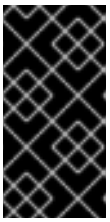
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.6.10. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

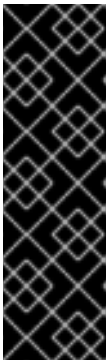
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



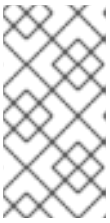
重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

15.6.10.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths": ...}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

(4) **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロ

イする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 OpenShift Cluster Manager から取得したプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

15.6.10.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress

トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

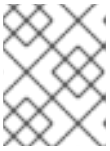
Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。**** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

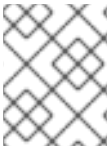


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

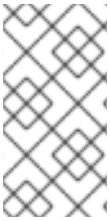
15.6.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

15.6.12. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware Cloud on AWS でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

15.6.13. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホ

ストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。
- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**

4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。

7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。

注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

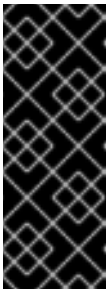
d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

- ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
- **install-config.yaml** ファイルで指定したデータストアを選択します。

f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。

i. 静的 IP 設定を行います。

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

コマンドの例

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

```
$ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
kargs=${IPCFG}"
```

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。VM の CPU とメモリーの予約に次の値があることを確認してください。
  - メモリー予約値は、設定されたメモリーサイズと同じである必要があります。
  - CPU 予約値は、低レイテンシー仮想 CPU の数に、実際に測定された物理 CPU 速度で乗算した数を最低でも指定する必要があります。
- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで、スチールクロックアカウンティング (**stealclock.enable**) に使用できるパラメーターのリストを検索します。利用可能な場合は、その値を **TRUE** に設定します。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。
- **設定パラメーターの追加** をクリックします。以下のパラメーター名および値を定義します。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

9. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

15.6.14. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

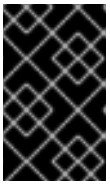
1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。
   i. 設定を完了し、仮想マシンの電源をオンにします。
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

15.6.15. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして作成する場合にのみ正式にサポートされます。



重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成

一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

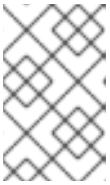
2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、 **manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

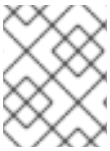
```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

15.6.16. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。



注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。

システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

15.6.17. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

15.6.18. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

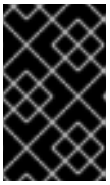
1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

15.6.19. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.6.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名

要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

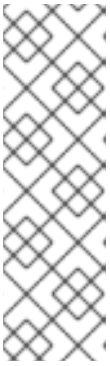
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

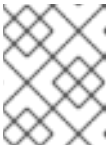
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

15.6.21. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

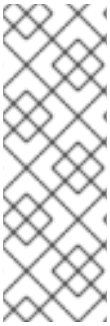
```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

15.6.21.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

15.6.21.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

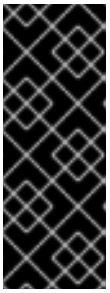
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.6.21.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

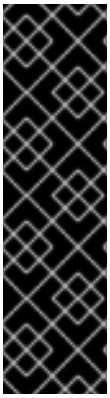
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



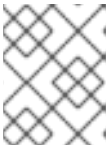
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

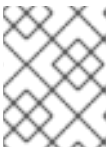
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

15.6.21.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

15.6.21.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

   (3) 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (4) 永続ボリューム要求 (PVC) のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

15.6.22. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
```

```
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

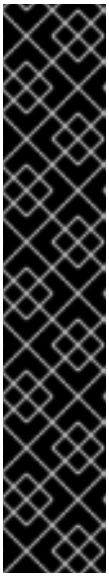
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver        apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver        apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver        apiserver-z25h4                             1/1     Running    0
```

```
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

15.6.23. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

15.6.24. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを

追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.6.25. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

15.7. ユーザーによってプロビジョニングされるインフラストラクチャーおよびネットワークのカスタマイズを使用した VMC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイし、これをカスタマイズされるネットワーク設定オプションでプロビジョニングされるインフラストラクチャーを使用して VMware vSphere インスタンスにインストールできます。

OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要なリソースのデプロイおよび管理プロセスを自動化します。

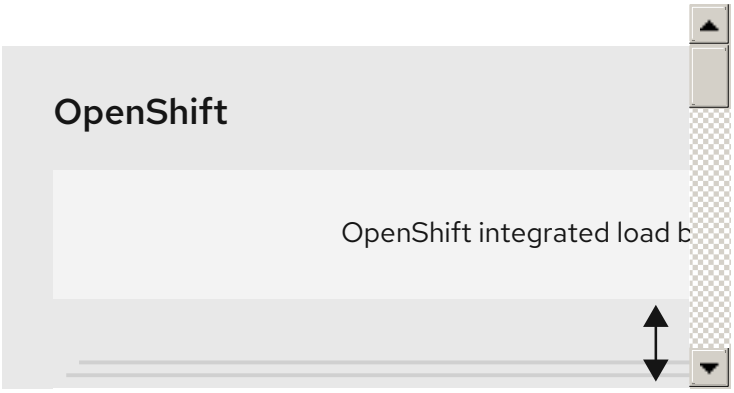
ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の VXLAN 設定と統合できます。大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.7.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



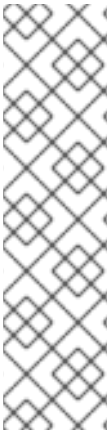
OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- 以下のファイアウォールルールを設定します。
  - OpenShift Container Platform コンピュートネットワークとインターネット間の ANY:ANY ファイアウォールルール。これは、コンテナーイメージをダウンロードするためにノードおよびアプリケーションによって使用されます。
  - ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
  - OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。
- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
  - OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。
  - ベース DNS 名 (**companyname.com** など)。
  - デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。
  - 以下の vCenter 情報:
    - vCenter ホスト名、ユーザー名、およびパスワード
    - データセンター名 (**SDDC-Datacenter** など)
    - クラスター名 (**Cluster-1** など)
    - ネットワーク名
    - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール



注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.7.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノードを使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ
- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.7.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用する場合)。

15.7.3. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

15.7.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.52 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.7.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

15.7.5.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表15.53 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。

ホスト	説明
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

15.7.5.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表15.54 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

15.7.5.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

15.7.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

15.7.5.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表15.55 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve

プロトコル	ポート	説明
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.56 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.57 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

15.7.5.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは

重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.58 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
コントロールプレーンマシン	**<master><n>. <cluster_name>. <base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>. <cluster_name>. <base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

15.7.5.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例15.16 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
```

```
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

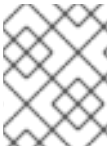
以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

-

例15.17 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。
2. Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。
3. ブートストラップマシンの逆引き DNS 解決を提供します。
4. 5. 6. コントロールプレーンマシンの逆引き DNS 解決を提供します。
7. 8. コンピュートマシンの逆引き DNS 解決を提供します。

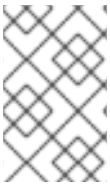


注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

15.7.5.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



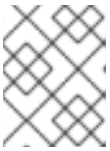
注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.59 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

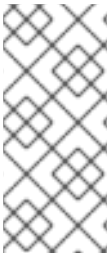
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表15.60 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

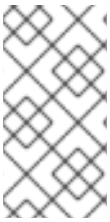
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

15.7.5.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例15.18 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster (1)
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 (2)
```

```
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 4
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 6
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 7
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1. この例では、クラスター名は **ocp4** です。
2. ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。
3. 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。
4. ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。
6. ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。
7. ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

15.7.6. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。

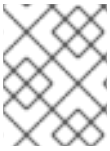
OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。

6. DNS 設定を検証します。

   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

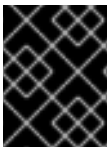


注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

15.7.7. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。



重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

出力例

```
api.ocp4.example.com.  0 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  0 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  0 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

15.7.8. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (~/**.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

(1) ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

15.7.9. インストールプログラムの取得

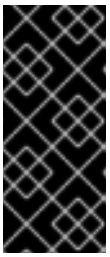
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

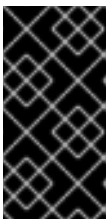
手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15.7.10. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

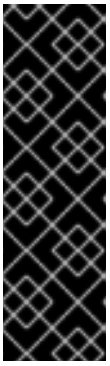
前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

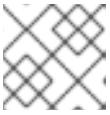
```
$ mkdir <installation_directory>
```



重要

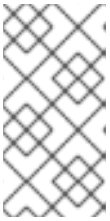
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。



注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

15.7.10.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
```

```
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
fips: false 15
pullSecret: '{"auths": ...}' 16
sshKey: 'ssh-ed25519 AAAA...' 17
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

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

インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 OpenShift Cluster Manager から取得したプルシークレット。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

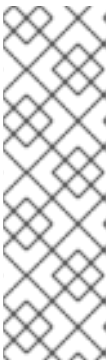
17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

15.7.10.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

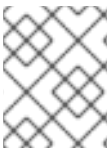
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

15.7.11. 高度なネットワーク設定の指定

クラスターネットワークプロバイダーに高度なネットワーク設定を使用し、クラスターを既存のネットワーク環境に統合することができます。高度なネットワーク設定は、クラスターのインストール前にのみ指定することができます。

重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルを変更してネットワーク設定をカスタマイズすることは、サポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了している。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、マニフェストを作成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前の、高度なネットワーク設定用のスタブマニフェストファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. 以下の例のように、**cluster-network-03-config.yml** ファイルで、クラスターの高度なネットワーク設定を指定します。

OpenShift SDN ネットワークプロバイダーに異なる VXLAN ポートを指定します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

OVN-Kubernetes ネットワークプロバイダーの IPsec を有効にします。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
```

```
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、Ignition 設定ファイルの作成時に **manifests/** ディレクトリーを使用します。

5. コントロールプレーンマシンおよび compute machineSets を定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- MachineSet ファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

15.7.12. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN または OVN-Kubernetes などのクラスターネットワークプロバイダー。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプロバイダー設定を指定できます。

15.7.12.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表15.61 Cluster Network Operator 設定オブジェクト

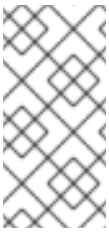
フィールド	タイプ	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

フィールド	タイプ	説明
spec.clusterNetwork	**array**	Pod ID アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定する一覧です。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes Container Network Interface (CNI) ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークの Container Network Interface (CNI) ネットワークプロバイダーを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表15.62 defaultNetwork オブジェクト

フィールド	タイプ	説明

フィールド	タイプ	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。クラスターネットワークプロバイダーはインストール時に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform はデフォルトで、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーを使用します。
openshiftSDNConfig	**object**	このオブジェクトは OpenShift SDN クラスターネットワークプロバイダーにのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは OVN-Kubernetes クラスターネットワークプロバイダーにのみ有効です。

OpenShift SDN CNI クラスターネットワークプロバイダーの設定
以下の表は、OpenShift SDN Container Network Interface (CNI) クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表15.63 openshiftSDNConfig オブジェクト

フィールド	タイプ	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	タイプ	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OpenShift SDN 設定の例

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定

以下の表は OVN-Kubernetes CNI クラスターネットワークプロバイダーの設定フィールドについて説明しています。

表15.64 ovnKubernetesConfig object

フィールド	タイプ	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。 この値は、クラスターのインストール後は変更できません。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

表15.65 **policyAuditConfig** object

フィールド	タイプ	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** または 50MB です。
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。

フィールド	タイプ	説明
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

OVN-Kubernetes 設定の例

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

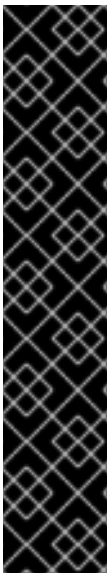
表15.66 kubeProxyConfig オブジェクト

フィールド	タイプ	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値:
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

15.7.13. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。



重要

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

前提条件

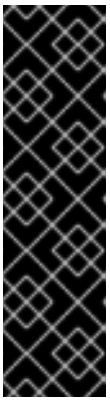
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得する。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

15.7.14. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware Cloud on AWS でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

15.7.15. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。

- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。

2. ブートストラップノードの以下の二次的な Ignition 設定ファイル を、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。

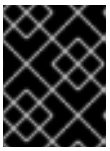
   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**

4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

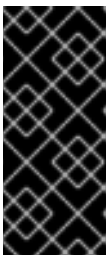
```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。
7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。

注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

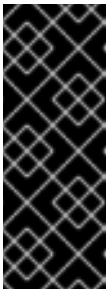
   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。
   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。
   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。
   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

- ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
- **install-config.yaml** ファイルで指定したデータストアを選択します。

f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。

i. 静的 IP 設定を行います。

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

コマンドの例

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

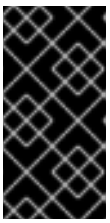
```
$ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
kargs=${IPCFG}"
```

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。VM の CPU とメモリーの予約に次の値があることを確認してください。
  - メモリー予約値は、設定されたメモリーサイズと同じである必要があります。
  - CPU 予約値は、低レイテンシー仮想 CPU の数に、実際に測定された物理 CPU 速度で乗算した数を最低でも指定する必要があります。
- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで、スチールクロックアカウンティング (**stealclock.enable**) に使用できるパラメーターのリストを検索します。利用可能な場合は、その値を **TRUE** に設定します。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。
- **設定パラメーターの追加** をクリックします。以下のパラメーター名および値を定義します。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

9. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

15.7.16. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。

- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。
   i. 設定を完了し、仮想マシンの電源をオンにします。
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

15.7.17. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして

作成する場合にのみ正式にサポートされます。

重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成

一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
```

```
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、**manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

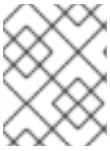
```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

15.7.18. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。



注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

15.7.19. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

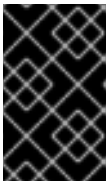
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

15.7.20. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.7.21. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

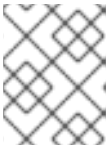
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

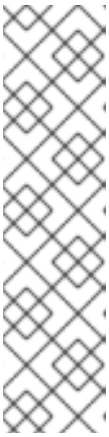
出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                    CONDITION
```

```
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

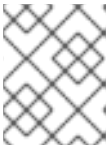
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

15.7.22. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

15.7.22.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

15.7.22.2. イメージレジストリーストレージの設定

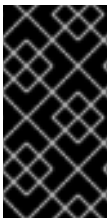
イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.7.22.2.1. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
```

```
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

(1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

(2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

(3) 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

(4) 永続ボリューム要求 (PVC) のサイズ。

b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

15.7.23. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

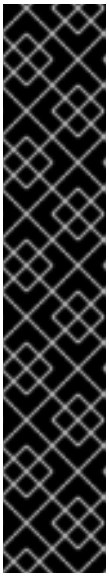
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                               1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                               1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running    0          5m
...
```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```

(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

15.7.24. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

15.7.25. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.7.26. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

15.8. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での VMC へのクラスターのインストール

OpenShift Container Platform バージョン 4.8 では、クラスターを VMware Cloud (VMC) on AWS にデプロイし、これを制限されたネットワークでプロビジョニングする VMware vSphere インフラストラクチャーにインストールできます。

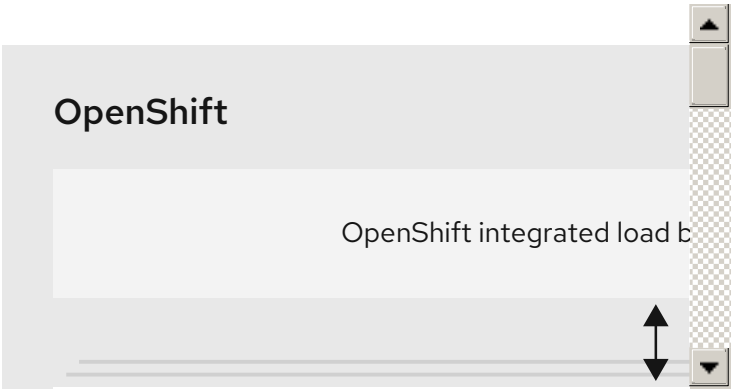
OpenShift Container Platform デプロイメント用に VMC 環境を設定した後に、VMC 環境に併設された bastion 管理ホストの OpenShift Container Platform インストールプログラムを使用します。インストールプログラムおよびコントロールプレーンは、OpenShift Container Platform クラスターに必要なリソースのデプロイおよび管理プロセスを自動化します。

注記

OpenShift Container Platform は、単一の VMware vCenter へのクラスターのデプロイのみをサポートします。複数の vCenter にマシン/マシンセットを含むクラスターをデプロイすることはサポートされていません。

15.8.1. vSphere 用の VMC の設定

OpenShift Container Platform を VMware Cloud (VMC) on AWS でホストされた vSphere クラスターにインストールし、アプリケーションをオンプレミスおよびオンプレミスの両方でハイブリッドクラウド全体にデプロイし、管理することができます。



OpenShift Container Platform を VMware vSphere にインストールする前に、複数のオプションを VMC 環境で設定する必要があります。VMC 環境が以下の前提条件を満たしていることを確認します。

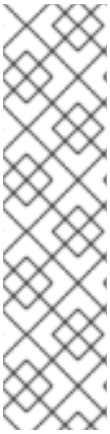
- 排他的ではない、 DHCP 対応の NSX-T ネットワークセグメントおよびサブネットを作成します。他の仮想マシン (VM) をサブネットでホストできますが、OpenShift Container Platform デプロイメントには 8 つ以上の IP アドレスが利用可能でなければなりません。
- 以下のファイアウォールルールを設定します。
  - ポート 443 上のインストールホストと、ソフトウェア定義データセンター (SDDC) 管理ネットワーク間の ANY:ANY ファイアウォールルール。これにより、デプロイメント時に Red Hat Enterprise Linux CoreOS (RHCOS) OVA をアップロードできます。
  - OpenShift Container Platform コンピュートネットワークと vCenter 間の HTTPS ファイアウォールルール。この接続により、OpenShift Container Platform はノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理するために vCenter と通信できます。
- OpenShift Container Platform をデプロイするには、以下の情報が必要です。
  - OpenShift Container Platform クラスター の名前 (**vmc-prod-1** など)

- OpenShift Container Platform クラスターの名前 (**vmc-prod-1** など)。
- ベース DNS 名 (**companyname.com** など)。
- デフォルトを使用しない場合、Pod ネットワーク CIDR およびサービスネットワーク CIDR を特定する必要があります。これはデフォルトで **10.128.0.0/14** および **172.30.0.0/16** にそれぞれ設定されます。これらの CIDR は Pod 間および Pod とサービス間の通信に使用され、外部からアクセスすることはできません。ただし、それらは組織内の既存のサブネットと重複することができません。
- 以下の vCenter 情報:
  - vCenter ホスト名、ユーザー名、およびパスワード
  - データセンター名 (**SDDC-Datacenter** など)
  - クラスター名 (**Cluster-1** など)
  - ネットワーク名
  - データストア名 (**WorkloadDatastore** など)

注記

クラスターのインストールの完了後に、vSphere クラスターを VMC **Compute-ResourcePool** リソースプールに移動することが推奨されます。

- bastion として VMC にデプロイされる Linux ベースのホスト。
  - bastion ホストには Red Hat Enterprise Linux (RHEL) または他の Linux ベースのホストを使用できます。インターネット接続と OVA を ESXi ホストにアップロードする機能が必要です。
  - OpenShift CLI ツールをダウンロードし、 bastion ホストにインストールします。
    - **openshift-install** インストールプログラム
    - OpenShift CLI (**oc**) ツール



注記

VMware NSX Container Plugin for Kubernetes (NCP) は使用できないため、NSX は OpenShift SDN として使用されません。VMC で現在利用できる NSX のバージョンは、OpenShift Container Platform で認定されている NCP のバージョンとは互換性がありません。

ただし、NSX DHCP サービスは、フルスタックの自動化 OpenShift Container Platform デプロイメントおよびマシン API の vSphere への統合によって手動または自動でプロビジョニングされたノードと共に仮想マシンの IP 管理に使用されます。さらに、NSX ファイアウォールルールは、OpenShift Container Platform クラスターの a アクセス、および bastion ホストと VMC vSphere ホスト間のアクセスを有効にするために作成されます。

15.8.1.1. VMC Sizer ツール

VMware Cloud on AWS は AWS ベアメタルインフラストラクチャー上に構築されます。これは、AWS ネイティブサービスを実行するベアメタルインフラストラクチャーと同じです。VMware cloud on

AWSS のソフトウェア定義データセンター (SDDC) がデプロイされると、これらの物理サーバーノードを使用し、単一のテナント方式で VMware ESXi ハイパーバイザーを実行します。つまり、物理インフラストラクチャーは、 VMC を使用して他のユーザーがアクセスすることはできません。仮想インフラストラクチャーをホストするために必要な物理ホストの数を考慮することが重要です。

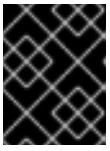
これを判別できるように、VMware は VMC on AWS Sizer を提供しています。このツールを使用して、VMC でホストするリソースを定義できます。

- ワークロードのタイプ
- 仮想マシンの合計数
- 仕様情報 (以下を含む)。
  - ストレージ要件
  - vCPU
  - vRAM
  - オーバーコミットの比率

これらの詳細情報により、Sizer ツールは VMware のベストプラクティスに基づいてレポートを生成し、クラスター設定および必要なホスト数について推奨します。

15.8.2. vSphere 要件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ミラーホストでレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- ブロックレジストリーストレージ をプロビジョニングしている。永続ストレージの詳細は、永続ストレージについて を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

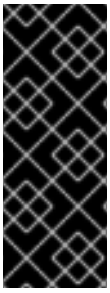
15.8.3. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.8 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インス

トーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の Route 53 DNS や IAM サービスなどの一部のクラウド機能には、インターネットアクセスが必要です。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

15.8.3.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

15.8.4. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

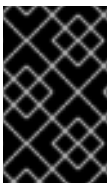
15.8.5. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 または 7 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表15.67 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 以降 (HW バージョン 13)	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U3 または 7.0 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

15.8.6. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

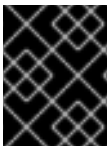
このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

15.8.6.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表15.68 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

15.8.6.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表15.69 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

15.8.6.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

15.8.6.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

15.8.6.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表15.70 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表15.71 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表15.72 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

15.8.6.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表15.73 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。 **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

15.8.6.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例15.19 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例15.20 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

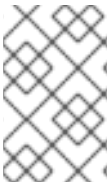
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

15.8.6.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

   表15.74 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

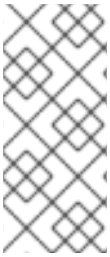
ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

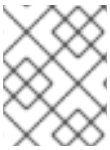
表15.75 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック



注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

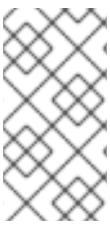


注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

15.8.6.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例15.21 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
```

```
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
frontend stats
  bind *:1936
  mode            http
  log             global
  maxconn 10
  stats enable
  stats hide-version
  stats refresh 30s
  stats show-node
  stats show-desc Stats for ocp4 cluster 1
  stats auth admin:ocp4
  stats uri /stats
listen api-server-6443 2
  bind *:6443
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup 3
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 4
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 5
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 6
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 7
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 この例では、クラスター名は **ocp4** です。

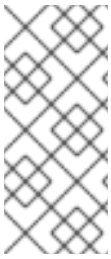
2 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

3 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

4 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

6 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

7 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

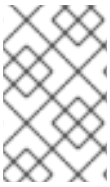


注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

15.8.7. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

15.8.8. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

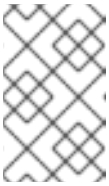
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

15.8.9. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。

重要

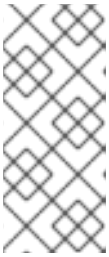
障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

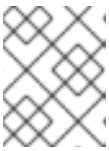
2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。



注記

一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

15.8.10. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得する。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

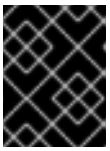
この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

15.8.10.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
additionalTrustBundle: | (18)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (19)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
```

```
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1. クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5. **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6. 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4. **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7. クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8. DNS レコードに指定したクラスター名。

9. vCenter サーバーの完全修飾ホスト名または IP アドレス。

10. サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11. vSphere ユーザーに関連付けられたパスワード。

12. vSphere データセンター。

13. 使用するデフォルトの vSphere データストア。

14. オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

16 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

18 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

19 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

15.8.10.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

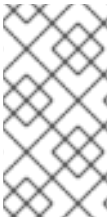
15.8.11. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

2. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

> **警告**
>
> 3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

> **重要**
>
> コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

3. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。
   c. ファイルを保存し、終了します。
4. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

15.8.12. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware Cloud on AWS でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得している。
- クラスターの Ignition 設定ファイルを生成している。
- **jq** パッケージをインストールしている。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

15.8.13. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を VMware vSphere のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) を vSphere ホストにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

前提条件

- クラスターの Ignition 設定ファイルを取得している。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセス権がある。

- vSphere クラスター を作成している。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。

2. ブートストラップノードの以下の二次的な Ignition 設定ファイル を、**<installation_directory>/merge-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "merge": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "3.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. インストールプログラムにより作成された次の Ignition 設定ファイルを見つけます。

   - **<installation_directory>/master.ign**
   - **<installation_directory>/worker.ign**
   - **<installation_directory>/merge-bootstrap.ign**

4. Ignition 設定ファイルを Base64 エンコーディングに変換します。この手順の後半で、これらのファイルを VM の追加の設定パラメーター **guestinfo.ignition.config.data** に追加する必要があります。
   たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

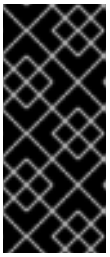
```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/merge-bootstrap.ign > <installation_directory>/merge-bootstrap.64
```

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。

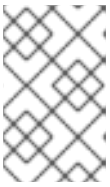


重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

6. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。このフォルダー名を使用すると、vCenter はその Workspace 設定に適した場所にあるストレージを動的にプロビジョニングします。
7. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。



注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

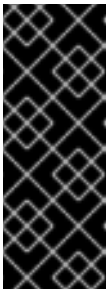
   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。
   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。
   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。
   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

- ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
- **install-config.yaml** ファイルで指定したデータストアを選択します。

f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

8. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。

- オプション: vSphere でデフォルトの DHCP ネットワークを上書きします。静的 IP ネットワークを有効にするには、以下を実行します。

i. 静的 IP 設定を行います。

```
$ export IPCFG="ip=<ip>::<gateway>:<netmask>:<hostname>:<iface>:none
nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]"
```

コマンドの例

```
$ export IPCFG="ip=192.168.100.101::192.168.100.254:255.255.255.0:::none
nameserver=8.8.8.8"
```

ii. vSphere で OVA から仮想マシンを起動する前に、**guestinfo.afterburn.initrd.network-kargs** プロパティーを設定します。

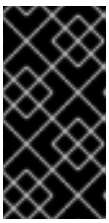
```
$ govc vm.change -vm "<vm_name>" -e "guestinfo.afterburn.initrd.network-
kargs=${IPCFG}"
```

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。VM の CPU とメモリーの予約に次の値があることを確認してください。
  - メモリー予約値は、設定されたメモリーサイズと同じである必要があります。
  - CPU 予約値は、低レイテンシー仮想 CPU の数に、実際に測定された物理 CPU 速度で乗算した数を最低でも指定する必要があります。
- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで、スチールクロックアカウンティング (**stealclock.enable**) に使用できるパラメーターのリストを検索します。利用可能な場合は、その値を **TRUE** に設定します。スチールクロックアカウンティングを有効にすると、クラスターの問題のトラブルシューティングに役立ちます。
- **設定パラメーターの追加** をクリックします。以下のパラメーター名および値を定義します。
  - **guestinfo.ignition.config.data**: この手順で先程作成した、base-64 でエンコードされたファイルを見つけて、このマシンタイプに関する base-64 でエンコードされた Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
  - **steelclock.enable**: このパラメーターが定義されていない場合は、追加して **TRUE** を指定します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

9. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

15.8.14. vSphere でのコンピュートマシンのクラスターへの追加

コンピュートマシンを VMware vSphere のユーザーがプロビジョニングした OpenShift Container Platform クラスターに追加することができます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。

- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。
   i. 設定を完了し、仮想マシンの電源をオンにします。
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

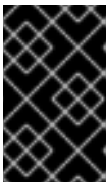
15.8.15. ディスクパーティション設定

ほとんどの場合、データパーティションは、最初に別のオペレーティングシステムをインストールするのではなく、RHCOS をインストールして作成されます。この場合、OpenShift Container Platform インストーラーでは、ディスクパーティションの設定が許可されます。

ただし、以下は、OpenShift Container Platform ノードのインストール時に、デフォルトのパーティション設定を上書きするために介入が必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションとして

作成する場合にのみ正式にサポートされます。



重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

個別の /var パーティションの作成

一般的に、OpenShift Container Platform のディスクパーティション設定は、インストーラーに任せる必要があります。ただし、拡張予定のファイルシステムの一部に個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** パーティションまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var は、Red Hat Enterprise Linux CoreOS (RHCOS) の新規インストール前に有効にする必要があるため、以下の手順では OpenShift Container Platform インストールの **openshift-install** の準備フェーズで挿入されるマシン設定マニフェストを作成して、別の **/var** パーティションを設定します。

手順

1. OpenShift Container Platform インストールファイルを保存するディレクトリーを作成します。

```
$ mkdir $HOME/clusterconfig
```

2. **openshift-install** を実行して、**manifest** および **openshift** のサブディレクトリーにファイルのセットを作成します。プロンプトが表示されたら、システムの質問に回答します。

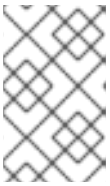
```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
```

```
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、ワーカーノードに異なるインスタンスタイプを使用することはできません。

4. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

5. **openshift-install** を再度実行し、**manifest** および **openshift** のサブディレクトリー内のファイルセットから、Ignition 設定を作成します。

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Ignition 設定ファイルを Red Hat Enterprise Linux CoreOS (RHCOS) システムをインストールために vSphere インストール手順への入力として使用できます。

15.8.16. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。

注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

```
# bootupctl status
```

出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
  Update: At latest version
```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
```

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

15.8.17. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

15.8.18. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

15.8.19. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

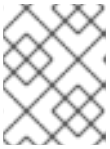
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

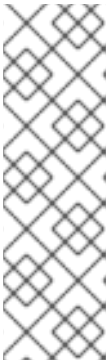
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

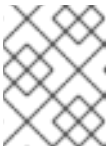
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE    REQUESTOR                                    CONDITION
```

```
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

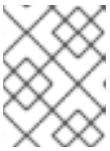
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

15.8.20. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

15.8.20.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

15.8.20.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

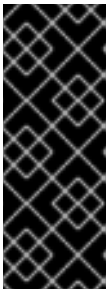
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

15.8.20.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

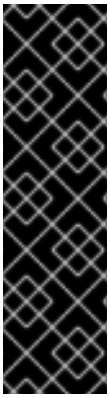
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

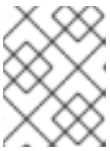
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False      6h50m
```

15.8.20.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

15.8.20.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

   (3) 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (4) 永続ボリューム要求 (PVC) のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、VMware vSphere のレジストリーの設定 について参照してください。

15.8.21. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
```

```
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

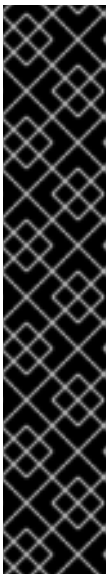
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running    0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running    0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running    0
```

```
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

4. Cluster registration ページでクラスターを登録します。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

15.8.22. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

15.8.23. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または

OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

15.8.24. 次のステップ

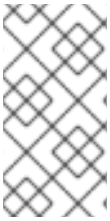
- クラスターをカスタマイズします。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、追加の信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- オプション: vSphere Problem Detector Operator からのイベントを表示 し、クラスターにパーミッションまたはストレージ設定の問題があるかどうかを判別します。

15.9. VMC のクラスターのアンインストール

インストーラーでプロビジョニングされるインフラストラクチャーを使用して、VMware Cloud (VMC) on AWS にデプロイし、VMware vSphere インフラストラクチャーにインストールされたクラスターを削除できます。

15.9.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

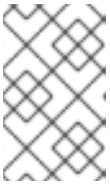
手順

1. クラスターをインストールするために使用したコンピューターのインストールプログラムが含まれるディレクトリーから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir <installation_directory> --log-level info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

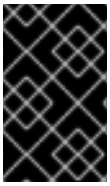
クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第16章 任意のプラットフォームへのインストール

16.1. クラスターの任意のプラットフォームへのインストール

OpenShift Container Platform バージョン 4.8 では、仮想化およびクラウド環境を含む、プロビジョニングする任意のインフラストラクチャーにクラスターをインストールできます。



重要

仮想化またはクラウド環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

16.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターインストール方法の選択およびそのユーザー向けの準備 のドキュメント内容を確認している。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要がある。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

16.1.2. OpenShift Container Platform のインターネットアクセス

OpenShift Container Platform 4.8 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

- OpenShift Cluster Manager にアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

16.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイする要件について説明します。

16.1.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表16.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。ただし、コンピュートマシンは Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) 7.9 のいずれかを選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

16.1.3.2. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表16.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	IOPS
ブートストラップ	RHCOS	4	16 GB	100 GB	該当なし
コントロールプレーン	RHCOS	4	16 GB	100 GB	該当なし
コンピュート	RHCOS	2	8 GB	100 GB	該当なし

1. 1 つの物理コア (IFL) は、SMT-2 が有効な場合に 2 つの論理コア (スレッド) を提供します。ハイパーバイザーは、2 つ以上の vCPU を提供できます。

16.1.3.3. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

16.1.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには HTTP または HTTPS サーバーが必要になります。

マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。マシンに永続 IP アドレスおよびホスト名があることを確認します。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

16.1.3.4.1. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

本セクションでは、必要なポートの詳細を説明します。

重要

接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するために、すべてのノードにインターネットへのアクセスが必要です。

表16.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表16.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表16.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ユーザーによってプロビジョニングされるインフラストラクチャーの NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、またはクラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

関連情報

- chrony タイムサービスの設定

16.1.3.5. ユーザーによってプロビジョニングされる DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- The Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされる OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表16.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<master><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノード (別名マスターノード) の各マシンを識別するための DNS A/AAAA または CNAME レコードと DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<worker><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。

16.1.3.5.1. ユーザーによってプロビジョニングされるクラスターの DNS 設定の例

このセクションでは、ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

ユーザーによってプロビジョニングされるクラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、ユーザーによってプロビジョニングされるクラスターの名前解決の A レコードの例を示しています。

例16.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
master0.ocp4.example.com. IN A 192.168.1.97 5
master1.ocp4.example.com. IN A 192.168.1.98 6
master2.ocp4.example.com. IN A 192.168.1.99 7
;
```

```
worker0.ocp4.example.com. IN A 192.168.1.11 8
worker1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。

2 Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。

3 ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

ユーザーによってプロビジョニングされるクラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、ユーザーによってプロビジョニングされるクラスターの逆引き名前解決の PTR レコードの例を示しています。

例16.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 4
```

```
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

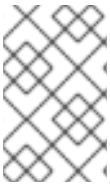
7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

16.1.3.6. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。

   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

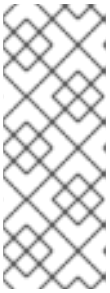
注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表16.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が健全な状態になり、3 つの要求が不健全な状態になります。これらは十分にテストされた値になります。

2. **アプリケーション Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表16.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック
1936	デフォルトでは、Ingress コントローラー Pod を実行するワーカーノード。入力ヘルスチェックプローブの **/healthz/ready** エンドポイントを設定する必要があります。	X	X	HTTP トラフィック

注記

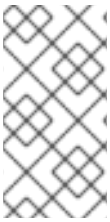
ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

16.1.3.6.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働シナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイできるため、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

例16.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
```

```
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    daemon
  defaults
    mode                    http
    log                     global
    option                  dontlognull
    option http-server-close
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
  frontend stats
    bind *:1936
    mode            http
    log             global
    maxconn 10
    stats enable
    stats hide-version
    stats refresh 30s
    stats show-node
    stats show-desc Stats for ocp4 cluster (1)
    stats auth admin:ocp4
    stats uri /stats
  listen api-server-6443 (2)
    bind *:6443
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:6443 check inter 1s backup (3)
    server master0 master0.ocp4.example.com:6443 check inter 1s
    server master1 master1.ocp4.example.com:6443 check inter 1s
    server master2 master2.ocp4.example.com:6443 check inter 1s
  listen machine-config-server-22623 (4)
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (5)
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
  listen ingress-router-443 (6)
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
  listen ingress-router-80 (7)
    bind *:80
    mode tcp
```

```
balance source
server worker0 worker0.ocp4.example.com:80 check inter 1s
server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1. この例では、クラスター名は **ocp4** です。
2. ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。
3. 5 ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。
4. ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。
6. ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。
7. ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

16.1.4. ユーザーによってプロビジョニングされるインフラストラクチャーの準備

ユーザーによってプロビジョニングされるインフラストラクチャーに OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要について説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続の設定、Ignition ファイルの Web サーバーの準備、ファイアウォール経由での必要なポートの有効化、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

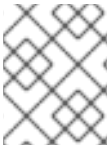
準備後、クラスターインフラストラクチャーは、**ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x のテスト済みインテグレーション を確認している。
- **ユーザーによってプロビジョニングされるインフラストラクチャーを使用したクラスターの要件**で説明されているインフラストラクチャーの要件を確認している。

手順

1. 静的 IP アドレスをセットアップします。
2. HTTP または HTTPS サーバーを設定し、Ignition ファイルをクラスターノードに提供します。
3. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
4. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件**のセクションを参照してください。
5. クラスターに必要な DNS インフラストラクチャーを設定します。
    a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**ユーザーによってプロビジョニングされる DNS 要件**のセクションを参照してください。
6. DNS 設定を検証します。
    a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。
    b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
    DNS 検証手順の詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証**のセクションを参照してください。
7. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。



注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

16.1.5. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 解決の検証

OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする前に、DNS 設定を検証できます。

重要

本セクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーに必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  0 IN A 192.168.1.5
   ```

   c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

   ```
   $ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   random.apps.ocp4.example.com.  0 IN A 192.168.1.5
   ```

-

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 0 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  0 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.
```

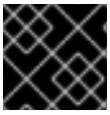
c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

16.1.6. クラスターノードの SSH アクセス用のキーペアの生成

OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定できます。キーは、Ignition 設定ファイルを介して Red Hat Enterprise Linux CoreOS (RHCOS) ノードに渡され、ノードへの SSH アクセスを認証するために使用されます。このキーは各ノードの **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加され、パスワードなしの認証が可能になります。

キーがノードに渡されると、キーペアを使用して RHCOS ノードにユーザー **core** として SSH を実行できます。SSH 経由でノードにアクセスするには、秘密鍵のアイデンティティーをローカルユーザーの SSH で管理する必要があります。

インストールのデバッグまたは障害復旧を実行するためにクラスターノードに対して SSH を実行する場合は、インストールプロセスの間に SSH 公開鍵を指定する必要があります。 **/openshift-install gather** コマンドでは、SSH 公開鍵がクラスターノードに配置されている必要もあります。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

手順

1. クラスターノードへの認証に使用するローカルマシンに既存の SSH キーペアがない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name> 1
```

1 新しい SSH キーのパスとファイル名 (**~/.ssh/id_ed25519** など) を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. 公開 SSH キーを表示します。

```
$ cat <path>/<file_name>.pub
```

たとえば、次のコマンドを実行して ~/**.ssh/id_ed25519.pub** 公開鍵を表示します。

```
$ cat ~/.ssh/id_ed25519.pub
```

3. ローカルユーザーの SSH エージェントに SSH 秘密鍵 ID が追加されていない場合は、それを追加します。キーの SSH エージェント管理は、クラスターノードへのパスワードなしの SSH 認証、または **./openshift-install gather** コマンドを使用する場合は必要になります。

注記

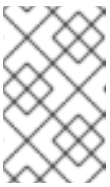
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

a. **ssh-agent** プロセスがローカルユーザーに対して実行されていない場合は、バックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```



注記

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

4. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

1 ~/**.ssh/id_ed25519** などの、SSH プライベートキーのパスおよびファイル名を指定します。

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

16.1.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシン (例: 500 MB のローカルディスク領域のある Red Hat Enterprise Linux 8) が必要です。

手順

1. OpenShift Cluster Manager サイトの インフラストラクチャープロバイダー ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. インフラストラクチャープロバイダーを選択します。
3. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターのインストール完了後は、インストールプログラムおよびインストールプログラムが作成するファイルを保持する必要があります。ファイルはいずれもクラスターを削除するために必要になります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。クラスターを削除するには、特定のクラウドプロバイダー用の OpenShift Container Platform のアンインストール手順を実行します。

4. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Red Hat OpenShift Cluster Manager からインストールプルシークレット をダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

16.1.8. バイナリーのダウンロードによる OpenShift CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.8 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.8 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

16.1.9. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされる OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- ローカルマシンには、インストールプログラムに提供する SSH 公開鍵があります。このキーは、デバッグおよび障害復旧のためにクラスターノードへの SSH 認証に使用されます。
- OpenShift Container Platform インストールプログラムおよびクラスターのプルシークレットを取得しています。

手順

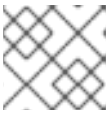
1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 提供されるサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

注記

一部のプラットフォームタイプでは、代わりに **./openshift-install create install-config --dir <installation_directory>** を実行して **install-config.yaml** ファイルを生成することができます。プロンプト時にクラスター設定の詳細を指定できます。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

16.1.9.1. IBM Z のサンプル install-config.yaml ファイル

16.1.9.2. 他のプラットフォーム用のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

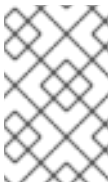
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



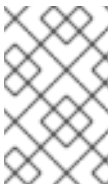
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする場合は、この値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。ユーザーによってプロビジョニングされるインストールでは、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

警告

Red Hat Virtualization は現在、oVirt プラットフォーム上にあるユーザーによってプロビジョニングされるインフラストラクチャーでのインストールをサポートしていません。そのため、プラットフォームを **none** に設定し、OpenShift Container Platform が各ノードをベアメタルノードとして、およびクラスターをベアメタルクラスターとして識別できるようにします。これは、任意のプラットフォームにクラスターをインストールする のと同じであり、次の制限があります。

1. クラスタープロバイダーがないため、各マシンを手動で追加する必要があり、ノードスケーリング機能はありません。
2. oVirt CSI ドライバーはインストールされず、CSI 機能はありません。

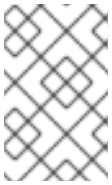
13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

14 Red Hat OpenShift Cluster Manager からのプルシークレット 。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16.1.9.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーをバイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
```

```
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合には、インストールプログラムは、**openshift-config** namespace に **user-ca-bundle** という名前の設定魔府を生成して、追加の CA 証明書を保存します。**additionalTrustBundle** と少なくとも 1 つのプロキシー設定を指定した場合には、**Proxy** オブジェクトは **trusted CA** フィールドで **user-ca-bundle** 設定マップを参照するように設定されます。その後、Cluster Network Operator は、**trustedCA** パラメーターに指定されたコンテンツを RHCOS トラストバンドルにマージする **trusted-ca-bundle** 設定マップを作成します。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

16.1.9.4. 3 ノードクラスターの設定

オプションで、ゼロ (0) コンピュートマシンを 3 つのコントロールプレーンマシンのみで設定されるベアメタルクラスターにデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

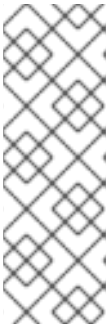
前提条件

- 既存の **install-config.yaml** ファイルがある。

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform をユーザーによってプロビジョニングされるインフラストラクチャーにインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。インストーラーでプロビジョニングされるインストールでは、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、ユーザーによってプロビジョニングされるインストールには適用されません。

注記

コントロールプレーンノードの推奨リソースは 6 vCPU および 21 GB です。コントロールプレーンノードが 3 つの場合には、これは最小の 5 ノードクラスターと同等のメモリー + vCPU です。3 つのノードをバックする必要があります。それぞれに、SMT2 が有効な IFL が 3 つ含まれる 120 GB ディスクにインストールします。各コントロールプレーンノードのテスト済みの最小設定とは、120 GB ディスクに 3 つの vCPU および 10 GB が指定された設定です。

3 ノードのクラスターのインストールについては、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件**のセクションを参照してください。

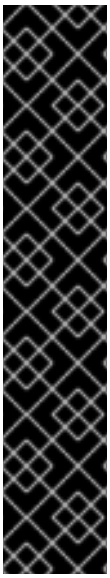
- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

16.1.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

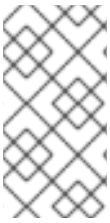
一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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



注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。インストールプログラムの Linux バージョンは s390x でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがワーカーノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

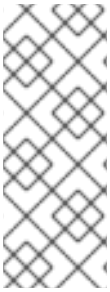
```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

16.1.11. RHCOS のインストールおよび OpenShift Container Platform ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングするベアメタルインフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift

Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

RHCOS をマシンにインストールするには、ISO イメージまたはネットワーク PXE ブートを使用する手順のいずれかを実行します。



注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4 の今後のリリースで削除される予定です。

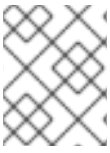
以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブインストール起動プロセスを中断してカーネル引数を追加できます。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。



注記

OpenShift Container Platform 4.6 の時点で、RHCOS ISO およびその他のインストールアーティファクトは、4K セクターのディスクへのインストールをサポートします。

16.1.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

   コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-
live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

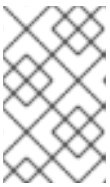
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。



注記

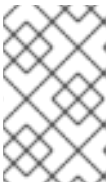
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **コア** ユーザーにはインストールを実行するために必要な root 権限がないため、**sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



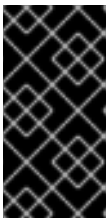
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する必要がある場合は、**coreos-installer** を実行する前に内部認証局 (CA) をシステム信頼ストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

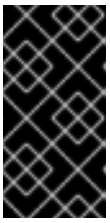


重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

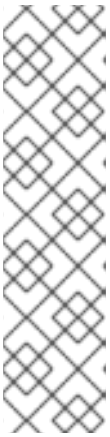
10. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

16.1.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE または iPXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定**のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

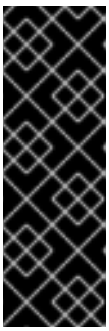
コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.8/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、 **initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**

- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. 使用する起動方法に必要な追加ファイルをアップロードします。

   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーと **rootfs** ファイルを HTTP サーバーにアップロードします。

   - iPXE の場合、**kernel**、**initramfs**、および **rootfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE または iPXE インストールを設定し、インストールを開始します。ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

- iPXE の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

1. HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。
2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。
3. HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. PXE UEFI を使用する場合は、以下の操作を実行します。
   a. システムの起動に必要な **shimx64.efi** and **grubx64.efi** EFI バイナリーと **grub.cfg** ファイルを指定します。
      - ホストに RHCOS ISO をマウントしてから、**images/efiboot.img** ファイルをマウントし、必要な EFI バイナリーを展開します。

```
$ mkdir -p /mnt/iso
```

```
$ mkdir -p /mnt/efiboot
```

```
$ mount -o loop rhcos-installer.x86_64.iso /mnt/iso
```

```
$ mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
```

- **efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
$ cp /mnt/efiboot/EFI/redhat/shimx64.efi .
```

```
$ cp /mnt/efiboot/EFI/redhat/grubx64.efi .
```

```
$ umount /mnt/efiboot
```

```
$ umount /mnt/iso
```

- RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

b. **grub.cfg** ファイルを編集し、以下のような引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linuxefi rhcos-<version>-live-kernel-<architecture> coreos.inst.install_dev=/dev/sda
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
 initrdefi rhcos-<version>-live-initramfs.<architecture>.img
}
```

詳細は以下のようになります。

rhcos-<version>-live-kernel-<architecture>

TFTP サーバーにアップロードした **kernel** ファイルを指定します。

http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img

HTTP サーバーにアップロードしたライブ rootfs イメージの場所を指定します。

http://<HTTP_server>/bootstrap.ign

HTTP サーバーにアップロードしたブートストラップ Ignition 設定ファイルの場所を指定します。

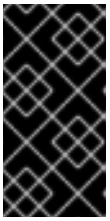
rhcos-<version>-live-initramfs.<architecture>.img

TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

UEFI ブート用に PXE サーバーを設定する方法は、Red Hat ナレッジベースの記事 How to configure/setup a PXE server for Red Hat Enterprise Linux? を参照してください。

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

16.1.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。本セクションでは、以下のような手法で実行できるいくつかの設定について説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ISO への Ignition 設定の埋め込み

本セクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

16.1.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

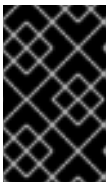
- 詳細の RHCOS インストールリファレンスの表を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

手順

1. ISO インストーラーを起動します。
2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。
3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

16.1.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。特定のアーキテクチャーの各 RHCOS ノードは、デフォルトのパーティション設定が上書きされない限り、同じパーティションレイアウトを使用します。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。

以下は、OpenShift Container Platform クラスターノードへの RHCOS のインストール時に、デフォルトのパーティション設定の上書きが必要と思われる 2 つのケースになります。

- 別個のパーティションの作成: 空のディスクへのグリーンフィールドインストールの場合は、別のストレージをパーティションに追加する必要がある場合があります。これは、**/var** または **/var/lib/etcd** などの **/var** のサブディレクトリー (両方ではない) を個別のパーティションにマウントする場合にのみ正式にサポートされます。

重要

Kubernetes は 2 つのファイルシステムパーティションのみをサポートします。元の設定に複数のパーティションを追加すると、Kubernetes はそれらをすべて監視できません。

- 既存のパーティションの保持: ブラウンフィールドインストールで、既存のノードに OpenShift Container Platform を再インストールし、以前のオペレーティングシステムからのデータパーティションを維持する必要がある場合、既存のデータパーティションを保持できる **coreos-installer** へのブート引数とオプションの両方があります。

警告

カスタムパーティションを使用すると、これらのパーティションが OpenShift Container Platform によって監視されないか、またはアラートが通知される可能性があります。デフォルトのパーティションを上書きする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

16.1.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

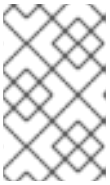
1. インストールホストで、OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ openshift-install create manifests --dir <installation_directory>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.9.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。
3. データパーティションのサイズ (メビバイト単位)。
4. コンテナーストレージに使用されるファイルシステムでは、 **prjquota** マウントオプションを有効にする必要があります。



注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、 **clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ openshift-install create ignition-configs --dir <installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

16.1.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、 **coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。

注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

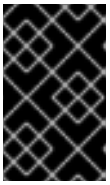
この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

16.1.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。



重要

これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは手動で作成され、Red Hat でサポートされていないため、可能な場合は使用しないようにする必要があります。この方法では、Ignition 設定はライブインストールメディアに渡され、起動時にすぐに実行され、RHCOS システムのディスクへのインストール前後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。
  PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、 Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

16.1.11.3.3.1. RHCOS ISO へのライブインストール Ignition 設定の埋め込み

ライブインストール Ignition 設定を RHCOS ISO イメージに直接埋め込むことができます。ISO イメージが起動すると、埋め込み設定が自動的に適用されます。

手順

1. 以下のイメージミラーページから **coreos-installer** バイナリーをダウンロードします:
   https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/
2. RHCOS ISO イメージおよび Ignition 設定ファイルを取得し、それらを **/mnt** などのアクセス可能なディレクトリーにコピーします。

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. 以下のコマンドを実行して Ignition 設定を ISO に埋め込みます。

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

   その ISO を使用して、指定されたライブインストール Ignition 設定を使用して RHCOS をインストールできるようになります。

重要

coreos-installer iso ignition embed を使用して、**openshift-installer** によって生成されるファイル (例: **bootstrap.ign**、**master.ign** および **worker.ign**) を埋め込むことはサポートされておらず、かつ推奨されていません。

4. 埋め込み Ignition 設定の内容を表示し、これをファイルに転送するには、以下を実行します。

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

出力例

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. Ignition 設定を削除し、再利用できるように ISO をその初期状態に戻すには、以下を実行します。

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

別の Ignition 設定を ISO に埋め込むか、または ISO を初期状態で使用することができるようになりました。

16.1.11.3.4. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションについて説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

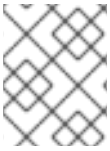
16.1.11.3.4.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法について説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、man ページの **dracut.cmdline** を参照してください。

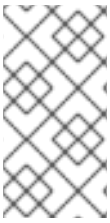
次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 IP アドレスの設定

IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない IP アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1つまたは複数のネットワークを設定する場合、1つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの DHCP の無効化
2 つ以上のネットワークインターフェイスがあり、1つのインターフェイスのみが使用される場合などに、1つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、DHCP は使用されない **enp2s0** について無効にされます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 IP 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの VLAN の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の DNS サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、network_interfaces は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。options はボンディングオプションのコンマ区切りの一覧です。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

  ```
  bond=bond0:em1,em2:mode=active-backup
  ip=bond0:dhcp
  ```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

共有 OSA/RoCE カードが使用されている場合の問題を回避するために、常にアクティブバックアップモードでオプション **fail_over_mac=1** を設定してください。

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team= name [:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、network_interfacesは物理 (イーサネット) インターフェイス (**em1、em2**) のコンマ区切りリストを表します。

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、Red Hat ナレッジベースアーティクル libvirt-lxc を使用した Linux コンテナー (廃止) を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

16.1.11.4. bootupd を使用したブートローダーの更新

bootupd を使用してブートローダーを更新するには、RHCOS マシンに **bootupd** を手動でインストールするか、または有効にされた **systemd** ユニットでマシン設定を指定する必要があります。**grubby** またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。

bootupd のインストール後に、これを OpenShift Container Platform クラスターからリモート管理できます。

注記

BootHole の脆弱性からの保護などを目的として、**bootupd** は、ベアメタルまたは仮想化ハイパーバイザーのインストールでのみ使用することが推奨されます。

手動のインストール方法

bootctl コマンドラインツールを使用して、**bootupd** を手動でインストールできます。

1. システムのステータスを検査します。

   ```
   # bootupctl status
   ```

   出力例

   ```
   Component EFI
     Installed: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
     Update: At latest version
   ```

2. インストールされている **bootupd** なしで作成された RHCOS イメージには、明示的な導入フェーズが必要になります。
   システムのステータスが **Adoptable** の場合に、導入を実行します。

   ```
   # bootupctl adopt-and-update
   ```

   出力例

   ```
   Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
   ```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

   ```
   # bootupctl update
   ```

   出力例

   ```
   Updated: grub2-efi-x64-1:2.04-31.fc33.x86_64,shim-x64-15-8.x86_64
   ```

-

マシン設定方法

bootupd を有効にするもう 1 つの方法としては、マシン設定を指定する方法があります。

- 以下の例のように、有効にされた **systemd** ユニットでマシン設定ファイルを指定します。

  #### 出力例

```
variant: rhcos
version: 1.1.0
systemd:
  units:
    - name: custom-bootupd-auto.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

16.1.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

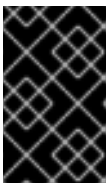
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.21.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

16.1.13. CLI の使用によるクラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイしていること。
- **oc** CLI をインストールしていること。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

16.1.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.21.0
master-1  Ready     master  63m  v1.21.0
master-2  Ready     master  64m  v1.21.0
```

出力には作成したすべてのマシンが一覧表示されます。

注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

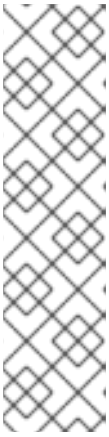
出力例

```
NAME        AGE   REQUESTOR                                   CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他のユーザーによってプロビジョニングされるインフラストラクチャーなどのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、 **oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードのアイデンティティーを確認します。

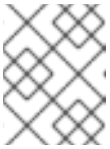
- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
```

```
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.21.0
   master-1  Ready     master  73m  v1.21.0
   master-2  Ready     master  74m  v1.21.0
   worker-0  Ready     worker  11m  v1.21.0
   worker-1  Ready     worker  11m  v1.21.0
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

16.1.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.2     True        False         False      19m
baremetal                                  4.8.2     True        False         False      37m
cloud-credential                           4.8.2     True        False         False      40m
cluster-autoscaler                         4.8.2     True        False         False      37m
config-operator                            4.8.2     True        False         False      38m
console                                    4.8.2     True        False         False      26m
csi-snapshot-controller                    4.8.2     True        False         False      37m
dns                                        4.8.2     True        False         False      37m
etcd                                       4.8.2     True        False         False      36m
image-registry                             4.8.2     True        False         False      31m
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

2. 利用不可の Operator を設定します。

16.1.15.1. デフォルトの OperatorHub ソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

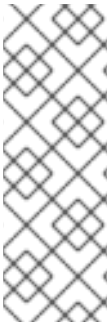
ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Global Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成し、削除し、無効にし、有効にすることができます。

16.1.15.2. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

16.1.15.3. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

16.1.15.3.1. IBM Z の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Z にクラスターがある。
- クラスターの永続ストレージをプロビジョニングしている。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

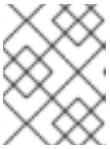
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME           VERSION                       AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                         True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

16.1.15.3.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

16.1.15.3.3. ブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

16.1.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication             4.8.2     True        False         False      19m
baremetal                  4.8.2     True        False         False      37m
cloud-credential           4.8.2     True        False         False      40m
cluster-autoscaler         4.8.2     True        False         False      37m
config-operator            4.8.2     True        False         False      38m
console                    4.8.2     True        False         False      26m
csi-snapshot-controller    4.8.2     True        False         False      37m
dns                        4.8.2     True        False         False      37m
etcd                       4.8.2     True        False         False      36m
image-registry             4.8.2     True        False         False      31m
```

```
ingress                                    4.8.2     True        False         False      30m
insights                                   4.8.2     True        False         False      31m
kube-apiserver                             4.8.2     True        False         False      26m
kube-controller-manager                    4.8.2     True        False         False      36m
kube-scheduler                             4.8.2     True        False         False      36m
kube-storage-version-migrator              4.8.2     True        False         False      37m
machine-api                                4.8.2     True        False         False      29m
machine-approver                           4.8.2     True        False         False      37m
machine-config                             4.8.2     True        False         False      36m
marketplace                                4.8.2     True        False         False      37m
monitoring                                 4.8.2     True        False         False      29m
network                                    4.8.2     True        False         False      38m
node-tuning                                4.8.2     True        False         False      37m
openshift-apiserver                        4.8.2     True        False         False      32m
openshift-controller-manager               4.8.2     True        False         False      30m
openshift-samples                          4.8.2     True        False         False      32m
operator-lifecycle-manager                 4.8.2     True        False         False      37m
operator-lifecycle-manager-catalog         4.8.2     True        False         False      37m
operator-lifecycle-manager-packageserver   4.8.2     True        False         False      32m
service-ca                                 4.8.2     True        False         False      38m
storage                                    4.8.2     True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

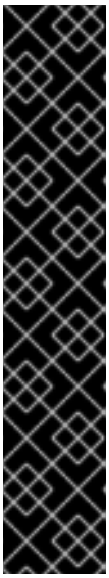
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                         READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

16.1.17. OpenShift Container Platform の Telemetry アクセス

OpenShift Container Platform 4.8 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager に登録されます。

OpenShift Cluster Manager インベントリーが正常である (Telemetry によって自動的に維持、または OpenShift Cluster Manager を使用して手動で維持) ことを確認した後に、subscription watch を使用 して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

関連情報

- Telemetry サービスの詳細は、リモートヘルスモニターリングについて を参照してください。

16.1.18. 次のステップ

- クラスターをカスタマイズします。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定します。

第17章 インストール設定

17.1. ノードのカスタマイズ

OpenShift Container Platform ノードへの直接の変更は推奨されませんが、必要とされる低レベルのセキュリティー、冗長性、ネットワーク、またはパフォーマンス機能を実装することが必要になる場合があります。OpenShift Container Platform ノードへの直接の変更は、以下によって実行できます。

- **openshift-install** の実行時にクラスターを起動するためにマニフェストファイルに組み込まれるマシン設定を作成します。
- Machine Config Operator を使用して実行中の OpenShift Container Platform ノードに渡されるマシン設定を作成します。
- ベアメタルノードのインストール時に **coreos-installer** に渡される Ignition 設定を作成します。

以下のセクションでは、この方法でノード上で設定する必要が生じる可能性のある機能について説明します。

17.1.1. Butane でのマシン設定の作成

マシン設定は、ユーザーおよびファイルシステムの作成、ネットワークの設定、systemd ユニットのインストールなどを行う方法をマシンに指示することで、コントロールプレーンマシンおよびワーカーマシンを設定するために使用されます。

マシン設定の変更は困難である可能性があるため、Butane 設定を使用してマシン設定を作成することができます。これにより、ノードの設定がより容易になります。

17.1.1.1. Butane について

Butane は、OpenShift Container Platform が使用するコマンドラインユーティリティーで、マシン設定を作成するための便利で簡略化した構文を提供したり、マシン設定の追加検証を実行したりします。Butane が受け入れる Butane 設定ファイルの形式は、OpenShift Butane config spec で定義されています。

17.1.1.2. Butane のインストール

Butane ツール (**butane**) をインストールして、コマンドラインインターフェイスから OpenShift Container Platform マシン設定を作成できます。対応するバイナリーファイルをダウンロードし、Linux、Windows、または macOS に **butane** をインストールできます。

ヒント

Butane リリースは、古いリリースと、Fedora CoreOS Config Transpiler (FCCT) との後方互換性があります。

手順

1. Butane イメージのダウンロードページ (https://mirror.openshift.com/pub/openshift-v4/clients/butane/) に移動してください。
2. **butane** バイナリーを取得します。

a. 最新バージョンの Butane の場合は、最新の **butane** イメージを現在のディレクトリーに保存します。

```
$ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane --output butane
```

b. オプション: aarch64 や ppc64le など、Butane をインストールする特定のタイプのアーキテクチャーの場合は、適切な URL を指定してください。以下に例を示します。

```
$ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane-aarch64 --output butane
```

3. ダウンロード済みのバイナリーファイルを実行可能にします。

```
$ chmod +x butane
```

4. **butane** バイナリーファイルを **PATH** にあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証手順

- **butane** コマンドを実行して、Butane ツールを使用できるようになりました。

```
$ butane <butane_file>
```

17.1.1.3. Butane を使用した MachineConfig オブジェクトの作成

Butane を使用して **MachineConfig** オブジェクトを作成できるため、インストール時に、または Machine Config Operator を使用して、ワーカーノードまたはコントロールプレーンノードを設定できます。

前提条件

- **butane** ユーティリティーをインストールしている。

手順

1. Butane 設定ファイルを作成します。以下の例では、**99-worker-custom.bu** という名前のファイルを作成します。このファイルは、カーネルデバッグメッセージを表示するようにシステムコンソールを設定し、chrony タイムサービスのカスタム設定を指定します。

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-worker-custom
  labels:
    machineconfiguration.openshift.io/role: worker
openshift:
  kernel_arguments:
    - loglevel=7
storage:
```

```
files:
  - path: /etc/chrony.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

注記

99-worker-custom.bu ファイルは、ワーカーノードのマシン設定を作成するように設定されます。コントロールプレーンノードにデプロイするには、ロールを **worker** から **master** に変更します。どちらの方法でも、デプロイメントの種類ごとに異なるファイル名を使用して手順全体を繰り返すことができます。

2. 直前の手順で作成したファイルを Butane に指定して **MachineConfig** オブジェクトを作成します。

```
$ butane 99-worker-custom.bu -o ./99-worker-custom.yaml
```

MachineConfig オブジェクト YAML ファイルは、マシンの設定を終了するために作成されます。

3. 将来的に **MachineConfig** オブジェクトを更新する必要がある場合に備えて、Butane 設定を保存します。

4. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、**MachineConfig** オブジェクト YAML ファイルを **openshift** ディレクトリーに追加します。クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

```
$ oc create -f 99-worker-custom.yaml
```

関連情報

- カーネルモジュールのノードへの追加
- インストール時のディスクの暗号化およびミラーリング

17.1.2. day-1 カーネル引数の追加

多くの場合、カーネル引数を day-2 アクティビティーとして変更することが推奨されますが、初期クラスターのインストール時にすべてのマスターまたはワーカーノードにカーネル引数を追加することができます。以下は、クラスターのインストール時にカーネル引数を追加して、システムの初回起動前に有効にする必要が生じる可能性のある理由です。

- SELinux などの機能を無効にし、初回起動時にシステムに影響を与えないようにする必要がある場合。

警告

RHCOS での SELinux の無効化はサポートされていません。

- システムの起動前に、低レベルのネットワーク設定を実行する必要がある場合。

カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

起動時に RHEL 8 カーネルに渡すことのできる引数の一覧については、Kernel.org カーネルパラメーター を参照してください。カーネル引数が OpenShift Container Platform の初回インストールを完了するために必要な場合は、この手順でカーネル引数のみを追加することが推奨されます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. カーネル引数をワーカーまたはコントロールプレーンノード (別名マスターノード) に追加するかどうかを決定します。

3. **openshift** ディレクトリーでファイル (例: **99-openshift-machineconfig-master-kargs.yaml**) を作成し、カーネル設定を追加するために **MachineConfig** オブジェクトを定義します。この例では、**loglevel=7** カーネル引数をコントロールプレーンノードに追加します。

```
$ cat << EOF > 99-openshift-machineconfig-master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
EOF
```

カーネル引数をワーカーノードに追加する場合は、**master** を **worker** に切り替えます。マスターおよびワーカーノードの両方に追加するために別々の YAML ファイルを作成します。

クラスターの作成を継続できます。

17.1.3. カーネルモジュールのノードへの追加

大半の一般的なハードウェアの場合、Linux カーネルには、コンピューターの起動時にそのハードウェアを使用するために必要となるデバイスドライバーモジュールが含まれます。ただし、一部のハードウェアの場合、Linux でモジュールを利用できません。したがって、各ホストコンピューターにこれら

のモジュールを提供する方法を確保する必要があります。この手順では、OpenShift Container Platform クラスターのノードについてこれを実行する方法を説明します。

この手順に従ってカーネルモジュールを最初にデプロイする際、モジュールは現行のカーネルに対して利用可能になります。新規カーネルがインストールされると、kmods-via-containers ソフトウェアはモジュールを再ビルドし、デプロイしてそのモジュールの新規カーネルと互換性のあるバージョンが利用可能になるようにします。

この機能によって各ノードでモジュールが最新の状態に保てるようにするために、以下が実行されます。

- 新規カーネルがインストールされているかどうかを検出するために、システムの起動時に起動する各ノードに systemd サービスを追加します。
- 新規カーネルが検出されると、サービスはモジュールを再ビルドし、これをカーネルにインストールします。

この手順に必要なソフトウェアの詳細については、kmods-via-containers github サイトを参照してください。

以下の重要な点に留意してください。

- この手順はテクノロジープレビューです。
- ソフトウェアのツールおよびサンプルは公式の RPM 形式で利用できず、現時点ではこの手順に記載されている非公式の **github.com** サイトからしか取得できません。
- この手順で追加する必要がある可能性のあるサードパーティーのカーネルモジュールについては Red Hat はサポートしません。
- この手順では、カーネルモジュールのビルドに必要なソフトウェアは RHEL 8 コンテナーにデプロイされます。モジュールは、ノードが新規カーネルを取得する際に各ノードで自動的に再ビルドされることに注意してください。このため、各ノードには、モジュールの再ビルドに必要なカーネルと関連パッケージを含む **yum** リポジトリーへのアクセスが必要です。このコンテンツは、有効な RHEL サブスクリプションを使用して効果的に利用できます。

17.1.3.1. カーネルモジュールコンテナーのビルドおよびテスト

カーネルモジュールを OpenShift Container Platform クラスターにデプロイする前に、プロセスを別の RHEL システムでテストできます。カーネルモジュールのソースコード、KVC フレームワーク、および kmod-via-containers ソフトウェアを収集します。次にモジュールをビルドし、テストします。RHEL 8 システムでこれを行うには、以下を実行します。

手順

1. RHEL 8 システムを登録します。

```
# subscription-manager register
```

2. RHEL 8 システムにサブスクリプションを割り当てます。

```
# subscription-manager attach --auto
```

3. ソフトウェアとコンテナーのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

4. **kmod-via-containers** リポジトリーのクローンを作成します。

   a. リポジトリーのフォルダーを作成します。

   ```
   $ mkdir kmods; cd kmods
   ```

   b. リポジトリーのクローンを作成します。

   ```
   $ git clone https://github.com/kmods-via-containers/kmods-via-containers
   ```

5. RHEL 8 ビルドホストに KVC フレームワークインスタンスをインストールし、モジュールをテストします。これにより、**kmods-via-container** systemd サービスが追加され、読み込まれます。

   a. **kmod-via-containers** ディレクトリーに移動します。

   ```
   $ cd kmods-via-containers/
   ```

   b. KVC フレームワークインスタンスをインストールします。

   ```
   $ sudo make install
   ```

   c. systemd マネージャー設定を再読み込みします。

   ```
   $ sudo systemctl daemon-reload
   ```

6. カーネルモジュールのソースコードを取得します。ソースコードは、制御下になく、他から提供されるサードパーティーモジュールをビルドするために使用される可能性があります。システムに対してクローン作成できる以下の **kvc-simple-kmod** サンプルのコンテンツと同様のコンテンツが必要になります。

   ```
   $ cd .. ; git clone https://github.com/kmods-via-containers/kvc-simple-kmod
   ```

7. この例では、設定ファイル **simple-kmod.conf** を編集し、Dockerfile の名前を **Dockerfile.rhel** に変更します。

   a. **kvc-simple-kmod** ディレクトリーに移動します。

   ```
   $ cd kvc-simple-kmod
   ```

   b. Dockerfile の名前を変更します。

   ```
   $ cat simple-kmod.conf
   ```

   Dockerfile の例

   ```
   KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-simple-kmod.git"
   KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
   ```

```
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

8. この例ではカーネルモジュール **simple-kmod** の **kmods-via-containers@.service** のインスタンスを作成します。

```
$ sudo make install
```

9. **kmods-via-containers@.service** インスタンスを有効にします。

```
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

10. systemd サービスを有効にし、起動します。

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service --now
```

a. サービスのステータスを確認します。

```
$ sudo systemctl status kmods-via-containers@simple-kmod.service
```

出力例

```
● kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
       enabled; vendor preset: disabled)
Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

11. カーネルモジュールがロードされていることを確認するには、**lsmod** コマンドを使用してモジュールを一覧表示します。

```
$ lsmod | grep simple_
```

出力例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

12. オプション。他の方法を使用して **simple-kmod** のサンプルが機能していることを確認します。

- **dmesg** を使ってカーネルリングバッファーで Hello world メッセージを探します。

```
$ dmesg | grep 'Hello world'
```

出力例

```
[ 6420.761332] Hello world from simple_kmod.
```

- /**proc** で **simple-procfs-kmod** の値を確認します。

```
$ sudo cat /proc/simple-procfs-kmod
```

出力例

```
simple-procfs-kmod number = 0
```

- **spkut** コマンドを実行して、モジュールの詳細情報を取得します。

```
$ sudo spkut 44
```

出力例

```
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
+ podman run -i --rm --privileged
  simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

その後は、システムの起動時に、このサービスは新規カーネルが実行中であるかどうかをチェックします。新規カーネルがある場合は、サービスは新規バージョンのカーネルモジュールをビルドし、これをロードします。モジュールがすでにビルドされている場合は、これをロードします。

17.1.3.2. カーネルモジュールの OpenShift Container Platform へのプロビジョニング

OpenShift Container Platform クラスターの初回起動時にカーネルモジュールを有効にする必要があるかどうかに応じて、以下のいずれかの方法でデプロイするようにカーネルモジュールを設定できます。

- **クラスターインストール時のカーネルモジュールのプロビジョニング (day-1)**: コンテンツを **MachineConfig** として作成し、これをマニフェストファイルのセットと共に組み込み、これを **openshift-install** に提供できます。
- **Machine Config Operator によるカーネルモジュールのプロビジョニング (day-2)**: カーネルモジュールを追加する際にクラスターが稼働するまで待機できる場合、Machine Config Operator (MCO) を使用してカーネルモジュールソフトウェアをデプロイできます。

いずれの場合も、各ノードは、新規カーネルの検出時にカーネルパッケージと関連ソフトウェアパッケージを取得できる必要があります。該当するコンテンツを取得できるように各ノードをセットアップする方法はいくつかあります。

- 各ノードに RHEL エンタイトルメントを提供します。
- **/etc/pki/entitlement** ディレクトリーから、既存 RHEL ホストの RHEL エンタイトルメントを取得し、それらを Ignition 設定の作成時に提供する他のファイルと同じ場所にコピーします。
- Dockerfile 内で、カーネルおよびその他のパッケージを含む **yum** リポジトリーへのポインターを追加します。これには、新たにインストールされたカーネルと一致させる必要があるため、新規のカーネルパッケージが含まれている必要があります。

17.1.3.2.1. MachineConfig オブジェクトを介したカーネルモジュールのプロビジョニング

MachineConfig オブジェクト でカーネルモジュールソフトウェアをパッケージ化することで、そのソフトウェアをインストール時に、または Machine Config Operator を使用して、ワーカーノードまたはコントロールプレーンノードに配信できます。

手順

1. RHEL 8 システムを登録します。

```
# subscription-manager register
```

2. RHEL 8 システムにサブスクリプションを割り当てます。

```
# subscription-manager attach --auto
```

3. ソフトウェアのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

4. カーネルモジュールおよびツールをホストするディレクトリーを作成します。

```
$ mkdir kmods; cd kmods
```

5. **kmods-via-containers** ソフトウェアを取得します。
   a. **kmods-via-containers** リポジトリーのクローンを作成します。

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

   b. **kvc-simple-kmod** リポジトリーのクローンを作成します。

```
$ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

6. モジュールソフトウェアを取得します。この例では、**kvc-simple-kmod** が使用されます。
7. fakeroot ディレクトリーを作成し、先にクローン作成したリポジトリーを使用して Ignition で配信するファイルを使用してこれを設定します。
   a. ディレクトリーを作成します。

```
$ FAKEROOT=$(mktemp -d)
```

   b. **kmod-via-containers** ディレクトリーに移動します。

```
$ cd kmods-via-containers
```

   c. KVC フレームワークインスタンスをインストールします。

```
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

   d. **kvc-simple-kmod** ディレクトリーに移動します。

```
$ cd ../kvc-simple-kmod
```

   e. インスタンスを作成します。

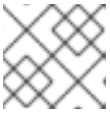
```
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

8. fakeroot ディレクトリーのクローンを作成し、以下のコマンドを実行してシンボリックリンク

8. fakeroot ディレクトリーのクローンを作成し、以下のコマンドを実行してシンボリックリンクをターゲットのコピーに置き換えます。

```
$ cd .. && rm -rf kmod-tree && cp -Lpr ${FAKEROOT} kmod-tree
```

9. カーネルモジュールツリーを埋め込む Butane 設定ファイル (**99-simple-kmod.bu**) を作成し、systemd サービスを有効にします。



注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-simple-kmod
  labels:
    machineconfiguration.openshift.io/role: worker 1
storage:
  trees:
    - local: kmod-tree
systemd:
  units:
    - name: kmods-via-containers@simple-kmod.service
      enabled: true
```

1 コントロールプレーンノードでデプロイするには、**worker** を **master** に変更します。コントロールプレーンおよびワーカーノードの両方にデプロイするには、それぞれのノードのタイプに対してこれらの残りの手順を 1 回ずつ実行します。

10. Butane を使用して、配信されるファイルおよび設定を含むマシン設定 YAML ファイルの **99-simple-kmod.yaml** を生成します。

```
$ butane 99-simple-kmod.bu --files-dir . -o 99-simple-kmod.yaml
```

11. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、そのファイルを **openshift** ディレクトリーに追加します。クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

```
$ oc create -f 99-simple-kmod.yaml
```

ノードは **kmods-via-containers@simple-kmod.service** サービスを起動し、カーネルモジュールがロードされます。

12. カーネルモジュールがロードされていることを確認するには、ノードにログインすることができます (**oc debug node/<openshift-node>** を使用してから **chroot /host** を使用します)。モジュールを一覧表示するには、**lsmod** コマンドを使用します。

```
$ lsmod | grep simple_
```

出力例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

17.1.4. インストール時のディスクの暗号化およびミラーリング

OpenShift Container Platform のインストール時に、クラスターノードでブートディスクの暗号化およびミラーリングを有効にできます。

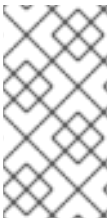
17.1.4.1. ディスクの暗号化について

インストール時に、コントロールプレーンおよびコンピュートノードのブートディスクの暗号化を有効できます。OpenShift Container Platform は Trusted Platform Module (TPM) v2 および Tang 暗号化モードをサポートします。

- TPM v2: これは優先されるモードです。TPM v2 は、サーバー内に含まれる安全な暗号プロセッサーにパスフレーズを保存します。このモードを使用すると、ディスクがサーバーから削除された場合にクラスターノードのブートディスクデータが復号化されないようにできます。
- tang: Tang および Clevis は、ネットワークバインドディスク暗号化 (NBDE) を有効にするサーバーおよびクライアントコンポーネントです。クラスターノードのブートディスクデータを 1 つまたは複数の Tang サーバーにバインドできます。これにより、ノードが Tang サーバーにアクセスできるセキュアなネットワーク上にある場合を除き、データの復号化ができなくなります。Clevis は、クライアント側の復号化の実装に使用される自動復号化フレームワークです。

重要

Tang 暗号化モードを使用したディスクの暗号化は、ユーザーによってプロビジョニングされるインフラストラクチャーでのベアメタルおよび vSphere インストールでのみサポートされます。



注記

以前のバージョンの Red Hat Enterprise Linux CoreOS (RHCOS) では、ディスク暗号化は Ignition 設定で **/etc/clevis.json** を指定して設定されました。このファイルは、OpenShift Container Platform 4.7 以降で作成されたクラスターではサポートされず、ディスクの暗号化は以下の手順で設定される必要があります。

TPM v2 または Tang 暗号化モードを有効にすると、RHCOS ブートディスクは LUKS2 形式を使用して暗号化されます。

この機能には以下の特徴があります。

- インストーラーでプロビジョニングされるインフラストラクチャーおよびユーザーによってプロビジョニングされるインフラストラクチャーのデプロイメントで利用可能である。
- Red Hat Enterprise Linux CoreOS (RHCOS) システムのみでサポートされる。
- マニフェストのインストールフェーズでディスク暗号化が設定される。これにより、初回起動時からディスクに書き込まれたすべてのデータが暗号化されます。
- パスフレーズを提供するのにユーザーの介入を必要としない。
- FIPS モードが有効な場合は、AES-256-XTS 暗号化、または AES-256-CBC を使用します。

17.1.4.1.1. 暗号化しきい値の設定

OpenShift Container Platform では、複数の Tang サーバーの要件を指定できます。TPM v2 および Tang 暗号化モードを同時に設定して、TPM のセキュアな暗号プロセッサーが存在し、安全なネットワーク上で Tang サーバーにアクセスできる場合にのみ、ブートディスクデータを復号化できます。

Butane 設定の **threshold** 属性を使用して、復号化できるようにするために必要な TPM v2 および Tang 暗号化条件の最小数を定義できます。宣言条件の組み合わせで指定値に到達した場合に、しきい値が満たされます。たとえば、2 台の Tang サーバーにアクセスするか、TPM のセキュアな暗号プロセッサーおよび Tang サーバーの 1 つにアクセスすることで、以下の設定の **2 しきい値** に到達できます。

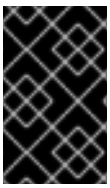
ディスク暗号化の Butane 設定例

```
variant: openshift
version: 4.8.0
metadata:
  name: worker-storage
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  layout: x86_64
  luks:
    tpm2: true (1)
    tang: (2)
      - url: http://tang1.example.com:7500
        thumbprint: jwGN5tRFK-kF6pIX89ssF3khxxX
      - url: http://tang2.example.com:7500
        thumbprint: VCJsvZFjBSIHSldw78rOrq7h2ZF
    threshold: 2 (3)
openshift:
  fips: true
```

(1) Trusted Platform Module (TPM) を使用してルートファイルシステムを暗号化する場合は、このフィールドを追加してください。

(2) 1 台以上の Tang サーバーを使用する必要がある場合は、このセクションを追加してください。

(3) 復号化に実行に必要な TPM v2 および Tang 暗号化条件の最小数を指定します。



重要

デフォルトの **しきい値** は **1** です。設定に複数の暗号化条件を追加するにも拘らず、しきい値を指定しない場合は、条件のいずれかが満たされている場合に復号化を実行できます。

注記

復号化に TPM v2 と Tang の両方が必要な場合には、**しきい値** 属性の値は、指定された Tang サーバーの合計数と 1 つの値である必要があります。**しきい値** が低い場合には、暗号化モードのいずれかのみを使用してしきい値に到達できます。たとえば、**tpm2** を **true** に設定して、2 台の Tang サーバーを指定すると、TPM の安全な暗号プロセッサーが利用できない場合でも 2 台の Tang サーバーにアクセスして、しきい値を **2** つ満たすことができます。

17.1.4.2. ディスクのミラーリングについて

コントロールプレーンおよびワーカーノードでの OpenShift Container Platform のインストール時に、ブートおよびその他のディスクの 2 つ以上の冗長ストレージデバイスへのミラーリングを有効にできます。ノードは、1 つのデバイスが利用可能な状態である限り、ストレージデバイスに障害が発生した後も引き続き機能します。

ミラーリングは、障害の発生したディスクの置き換えをサポートしません。ミラーを元の低下していない状態に復元するには、ノードを再プロビジョニングします。

注記

ミラーリングは、RHCOS システムのユーザーによってプロビジョニングされるインフラストラクチャーのデプロイメントでのみ利用できます。ミラーリングのサポートは、BIOS または UEFI で起動した x86_64 ノード、および ppc64le ノードで利用できます。

17.1.4.3. ディスク暗号化およびミラーリングの設定

OpenShift Container Platform のインストール時に暗号化およびミラーリングを有効にし、設定することができます。

前提条件

- インストールノードで OpenShift Container Platform インストールプログラムをダウンロードしている。
- インストールノードに Butane がインストールされている。

注記

Butane は、OpenShift Container Platform が使用するコマンドラインユーティリティーで、マシン設定を作成するための便利で簡略化した構文を提供したり、マシン設定の追加検証を実行したりします。詳細は、**Butane を使用したマシン設定の作成** セクションを参照してください。

- Tang 交換キーのサムプリントの生成に使用できる Red Hat Enterprise Linux (RHEL) 8 マシンにアクセスできる。

手順

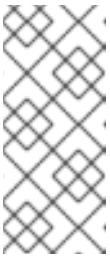
1. TPM v2 を使用してクラスターを暗号化する必要がある場合、TPM v2 暗号化を各ノードの BIOS で有効にする必要があるかどうかを確認します。これは、ほとんどの Dell システムで必要になります。コンピューターのマニュアルを確認してください。
2. Tang を使用してクラスターを暗号化する必要がある場合は、以下の準備段階の手順に従います。
   a. Tang サーバーを設定するか、または既存のサーバーにアクセスします。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。
   b. RHEL 8 マシンに **clevis** パッケージがインストールされていない場合はインストールします。

```
$ sudo yum install clevis
```

c. RHEL 8 マシンで以下のコマンドを実行し、交換キーのサムプリントを生成します。**http://tang.example.com:7500** を Tang サーバーの URL に置き換えます。

```
$ clevis-encrypt-tang '{"url":"http://tang.example.com:7500"}' < /dev/null > /dev/null (1)
```

(1) この例では、**tangd.socket** は Tang サーバーのポート **7500** でリッスンしています。



注記

このステップでは、**clevis-encrypt-tang** コマンドを使用して、交換キーのサムプリントを生成します。この時点で暗号化用のデータがコマンドに渡されないため、**/dev/null** はプレーンテキストではなく、インプットとして提供されます。この手順には必要ないため、暗号化された出力は **/dev/null** に送信されます。

出力例

```
The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9 (1)
```

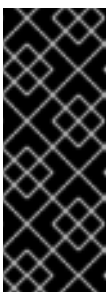
(1) エクスチェンジキーのサムプリント。

Do you want to trust these keys? [ynYN] のプロンプトが表示されたら、**Y** と入力します。

注記

RHEL 8 には、Clevis バージョン 15 が同梱されており、SHA-1 ハッシュアルゴリズムを使用してサムプリントを生成します。その他のディストリビューションには、Clevis バージョン 17 以降があり、サムプリントに SHA-256 ハッシュアルゴリズムを使用します。サムプリントを作成するために SHA-1 を使用する Clevis バージョンを使用し、OpenShift Container Platform クラスターノードに Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に Clevis バインディングの問題を防ぐ必要があります。

d. ノードが静的 IP アドレス指定で設定されている場合は、RHCOS ノードのインストール時に **coreos-installer --append-karg** オプションを使用してインストール済みシステムの IP アドレスを設定します。ネットワークに必要な **ip=** およびその他の引数を追加します。



重要

一部の静的 IP の設定方法は、初回のブート後に initramfs に影響を与えず、Tang 暗号化では機能しない場合があります。これらには、**coreos-installer --copy-network** オプションが含まれ、さらにインストール時に **ip=** 引数をライブ ISO または PXE イメージのカーネルコマンドラインに追加することも含まれます。静的 IP 設定が間違っていると、ノードの 2 回目のブートが失敗します。

3. インストールノードで、インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 **<installation_directory>** は、インストールファイルを保存するディレクトリーへのパスに置き換えます。

4. ディスクの暗号化、ミラーリング、またはそれら両方を設定する Butane 設定を作成します。たとえば、コンピュートノードのストレージを設定するには、**$HOME/clusterconfig/worker-storage.bu** ファイルを作成します。

起動デバイスの Butane 設定例

```
variant: openshift
version: 4.8.0
metadata:
  name: worker-storage 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
boot_device:
  layout: x86_64 3
  luks: 4
    tpm2: true 5
    tang: 6
      - url: http://tang.example.com:7500 7
        thumbprint: PLjNyRdGw03zlRoGjQYMahSZGu9 8
    threshold: 1 9
  mirror: 10
    devices: 11
      - /dev/sda
      - /dev/sdb
openshift:
  fips: true 12
```

1 2 コントロールプレーンの設定については、これらの両方の場所で **worker** を **master** に置き換えます。

3 ppc64le ノードで、このフィールドを **ppc64le** に設定します。その他のすべてのノードで、このフィールドは省略できます。

4 ルートファイルシステムを暗号化する必要がある場合は、このセクションを追加してください。詳細は、**ディスク暗号化について** のセクションを参照してください。

5 Trusted Platform Module (TPM) を使用してルートファイルシステムを暗号化する場合は、このフィールドを追加してください。

6 1 台以上の Tang サーバーを使用する必要がある場合は、このセクションを追加してください。

7 Tang サーバーの URL を指定します。この例では、**tangd.socket** は Tang サーバーのポート **7500** でリッスンしています。

8 前述のステップで生成された Exchange キーサムプリントを指定します。

9 復号化に実行に必要な TPM v2 および Tang 暗号化条件の最小数を指定します。デフォル

10 ブートディスクをミラーリングする必要がある場合は、このセクションを追加してください。詳細は、**ディスクのミラーリングについて** を参照してください。

11 RHCOS がインストールされるディスクを含む、ブートディスクミラーに含まれる必要があるすべてのディスクデバイスを一覧表示します。

12 クラスターで FIPS モードを有効にするためにこのディレクティブを追加します。

重要

ノードをディスク暗号化とミラーリングの両方を使用するように設定する場合、両方の機能を同じ Butane 設定に設定する必要があります。さらに、FIPS モードが有効にされたノードでディスク暗号化を設定する場合、FIPS モードが別のマニフェストで有効化されている場合でも、同じ Butane 設定に **fips** ディレクティブを追加する必要があります。

5. 対応する Butane 設定からコントロールプレーンまたはコンピュートノードのマニフェストを作成し、**<installation_directory>/openshift** ディレクトリーに保存します。たとえば、コンピュートノードのマニフェストを作成するには、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/worker-storage.bu -o <installation_directory>/openshift/99-worker-storage.yaml
```

ディスクの暗号化またはミラーリングを必要とするノード種別ごとに、この手順を繰り返します。

6. 今後マニフェストを更新する必要がある場合には、Butane 設定を保存します。

7. 残りの OpenShift Container Platform インストールを継続します。

ヒント

インストール時に、ディスク暗号化またはミラーリングに関連するエラーメッセージがないか、RHCOS ノードでコンソールログをモニタリングできます。

重要

追加のデータパーティションを設定する場合、暗号化が明示的に要求されない限り、それらは暗号化されません。

検証

OpenShift Container Platform のインストール後に、ブートディスクの暗号化またはミラーリングがクラスターノードで有効にされているかどうかを確認できます。

1. インストールホストから、デバッグ Pod を使用してクラスターノードにアクセスします。

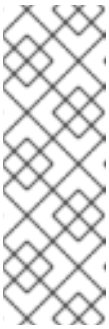
   a. ノードのデバッグ Pod を起動します。以下の例では、**compute-1** ノードのデバッグ Pod を起動します。

```
$ oc debug node/compute-1
```

   b. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、

Pod 内の **/host** にノードのルートファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ノードの実行可能パスに含まれるバイナリーを実行できます。

```
# chroot /host
```



注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、**kubelet** がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

2. ブートディスクの暗号化を設定している場合は、有効であるかどうかを確認します。

   a. デバッグシェルで、ノードでのルートマッピングのステータスを確認します。

```
# cryptsetup status root
```

出力例

```
/dev/mapper/root is active and is in use.
  type:    LUKS2 (1)
  cipher:  aes-xts-plain64 (2)
  keysize: 512 bits
  key location: keyring
  device:  /dev/sda4 (3)
  sector size:  512
  offset:  32768 sectors
  size:    15683456 sectors
  mode:    read/write
```

(1) 暗号化形式。TPM v2 または Tang 暗号化モードを有効にすると、RHCOS ブートディスクは LUKS2 形式を使用して暗号化されます。

(2) LUKS2 ボリュームの暗号化に使用される暗号化アルゴリズム。FIPS モードが有効な場合には 、**aes-cbc-essiv:sha256** 暗号が使用されます。

(3) 暗号化した LUKS2 ボリュームを含むデバイス。ミラーリングを有効にすると、値は **/dev/md126** などのソフトウェアミラーデバイスを表します。

   b. 暗号化されたデバイスにバインドされる Clevis プラグインを一覧表示します。

```
# clevis luks list -d /dev/sda4 (1)
```

(1) 前述のステップの出力の **device** フィールドに一覧表示されるデバイスを指定します。

出力例

```
1: sss '{"t":1,"pins":{"tang":[{"url":"http://tang.example.com:7500"}]}}' 1
```

1 この出力例では、Tang プラグインは、**/dev/sda4** デバイスの Shamir の Secret Sharing (SSS) Clevis プラグインにより使用されます。

3. ミラーリングを設定している場合は、有効かどうかを確認します。

a. デバッグシェルから、ノードにあるソフトウェアの RAID デバイスの一覧を表示します。

```
# cat /proc/mdstat
```

出力例

```
Personalities : [raid1]
md126 : active raid1 sdb3[1] sda3[0] 1
  393152 blocks super 1.0 [2/2] [UU]

md127 : active raid1 sda4[0] sdb4[1] 2
  51869632 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```

1 この例では、**/dev/md126** ソフトウェア RAID ミラーデバイスは、クラスターノードの **/dev/sda3** および **/dev/sdb3** ディスクデバイスを使用します。

2 この例では、**/dev/md127** ソフトウェア RAID ミラーデバイスは、クラスターノードの **/dev/sda4** および **/dev/sdb4** ディスクデバイスを使用します。

b. 上記のコマンドの出力に記載されている各ソフトウェア RAID デバイスの詳細を確認してください。以下の例は、**/dev/md126** デバイスの詳細を示しています。

```
# mdadm --detail /dev/md126
```

出力例

```
/dev/md126:
           Version : 1.0
     Creation Time : Wed Jul  7 11:07:36 2021
        Raid Level : raid1 1
        Array Size : 393152 (383.94 MiB 402.59 MB)
     Used Dev Size : 393152 (383.94 MiB 402.59 MB)
      Raid Devices : 2
     Total Devices : 2
       Persistence : Superblock is persistent

       Update Time : Wed Jul  7 11:18:24 2021
             State : clean 2
    Active Devices : 2 3
   Working Devices : 2 4
```

```
    Failed Devices : 0 (5)
     Spare Devices : 0

Consistency Policy : resync

          Name : any:md-boot (6)
          UUID : ccfa3801:c520e0b5:2bee2755:69043055
        Events : 19

    Number   Major   Minor   RaidDevice State
       0     252       3        0      active sync   /dev/sda3 (7)
       1     252       19       1      active sync   /dev/sdb3 (8)
```

(1) デバイスの RAID レベルを指定します。**raid1** は、RAID 1 ディスクミラーリングを示します。

(2) RAID デバイスの状態を指定します。

(3) (4) アクティブかつ機能している基礎となるディスクデバイスの数を示します。

(5) ステータスが failed のディスクデバイスの数を示します。

(6) ソフトウェア RAID デバイスの名前。

(7) (8) ソフトウェア RAID デバイスが使用する基礎となるディスクデバイスに関する情報を提供します。

c. ソフトウェア RAID デバイスにマウントされているファイルシステムの一覧を表示します。

```
# mount | grep /dev/md
```

出力例

```
/dev/md127 on / type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /etc type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /usr type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /sysroot type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/containers/storage/overlay type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/1 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/2 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/3 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
```

```
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-subpaths/etc/tuned/4 type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-subpaths/etc/tuned/5 type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md126 on /boot type ext4 (rw,relatime,seclabel)
```

この出力例では、**/boot** ファイルシステムが **/dev/md126** software RAID デバイスに、root ファイルシステムが **/dev/md127** にマウントされ t ています。

4. OpenShift Container Platform ノードタイプごとに検証手順を繰り返します。

関連情報

- TPM v2 および Tang 暗号化モードの詳細は、ポリシーベースの複号を使用して暗号化ボリュームの自動アンロックの設定 を参照してください。

17.1.4.4. RAID 対応のデータボリュームの設定

ソフトウェア RAID のパーティション設定を有効にして、外部データボリュームを提供できます。OpenShift Container Platform は、データ保護およびフォールトトレランスに対応するために RAID 0、RAID 1、RAID 4、RAID 5、RAID 6、および RAID 10 をサポートします。詳細は、ディスクのミラーリングについてを参照してください。

前提条件

- インストールノードで OpenShift Container Platform インストールプログラムをダウンロードしている。
- インストールノードに Butane をインストールしている。

注記

Butane は、OpenShift Container Platform が使用するコマンドラインユーティリティーで、マシン設定を作成するための便利で簡略化した構文を提供したり、マシン設定の追加検証を実行したりします。詳細は、**Butane を使用したマシン設定の作成** セクションを参照してください。

手順

1. ソフトウェア RAID を使用してデータボリュームを設定する Butane 設定を作成します。
   - ミラーリングされた起動ディスクに使用されるのと同じディスク上に RAID 1 を使用してデータボリュームを設定するには、**$HOME/clusterconfig/raid1-storage.bu** ファイルを作成します。以下に例を示します。

     ミラーリングされた起動ディスク上の RAID 1

```
variant: openshift
version: 4.8.0
metadata:
  name: raid1-storage
  labels:
    machineconfiguration.openshift.io/role: worker
```

```
boot_device:
  mirror:
    devices:
      - /dev/sda
      - /dev/sdb
storage:
  disks:
    - device: /dev/sda
      partitions:
        - label: root-1
          size_mib: 25000 1
        - label: var-1
    - device: /dev/sdb
      partitions:
        - label: root-2
          size_mib: 25000 2
        - label: var-2
  raid:
    - name: md-var
      level: raid1
      devices:
        - /dev/disk/by-partlabel/var-1
        - /dev/disk/by-partlabel/var-2
  filesystems:
    - device: /dev/md/md-var
      path: /var
      format: xfs
      wipe_filesystem: true
      with_mount_unit: true
```

1 2 データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小値が推奨されます。値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

- セカンダリーディスク上に RAID 1 を使用してデータボリュームを設定するには、**$HOME/clusterconfig/raid1-alt-storage.bu** ファイルを作成します。以下に例を示します。

  セカンダリーディスク上の RAID 1

```
variant: openshift
version: 4.8.0
metadata:
  name: raid1-alt-storage
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  disks:
    - device: /dev/sdc
      wipe_table: true
      partitions:
        - label: data-1
```

```
  - device: /dev/sdd
    wipe_table: true
    partitions:
      - label: data-2
raid:
  - name: md-var-lib-containers
    level: raid1
    devices:
      - /dev/disk/by-partlabel/data-1
      - /dev/disk/by-partlabel/data-2
filesystems:
  - device: /dev/md/md-var-lib-containers
    path: /var/lib/containers
    format: xfs
    wipe_filesystem: true
    with_mount_unit: true
```

2. 前のステップで作成した Butane 設定から RAID マニフェストを作成し、それを **<installation_directory>/openshift** ディレクトリーに保存します。たとえば、コンピュートノードのマニフェストを作成するには、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/<butane_config>.bu -o
<installation_directory>/openshift/<manifest_name>.yaml 1
```

1 **<butane_config>** および **<manifest_name>** を直前の手順のファイル名に置き換えます。たとえば、セカンダリーディスクの場合は、**raid1-alt-storage.bu** および **raid1-alt-storage.yaml** になります。

3. 今後マニフェストを更新する必要がある場合には、Butane 設定を保存します。

4. 残りの OpenShift Container Platform インストールを継続します。

17.1.5. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定できます。

手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。

注記

Butane の詳細は、Butane を使用したマシン設定の作成を参照してください。

```
variant: openshift
version: 4.8.0
metadata:
  name: 99-worker-chrony 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
```

```
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 3
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst 4
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

1 2 コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

3 マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc <mc-name> -o yaml** で YAML ファイルを確認できます。

4 DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。または、NTP サーバーの **1.rhel.pool.ntp.org**、**2.rhel.pool.ntp.org**、または **3.rhel.pool.ntp.org** のいずれかを指定できます。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

- クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
- クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-chrony.yaml
```

17.1.6. 関連情報

- Butane の詳細は、Butane を使用したマシン設定の作成 を参照してください。
- FIPS サポートの詳細は、FIPS 暗号のサポート を参照してください。

17.2. ファイアウォールの設定

ファイアウォールを使用する場合、OpenShift Container Platform が機能するために必要なサイトにアクセスできるように設定する必要があります。一部のサイトにはアクセスを常に付与し、クラスターをホストするために Red Hat Insights、Telemetry サービス、クラウドを使用したり、特定のビルドストラテジーをホストする場合に追加のアクセスを付与する必要があります。

17.2.1. OpenShift Container Platform のファイアウォールの設定

OpenShift Container Platform をインストールする前に、ファイアウォールを、OpenShift Container Platform が必要とするサイトへのアクセスを付与するように設定する必要があります。

コントローラーノードのみで実行されるサービスとワーカーノードで実行されるサービスの設定に関する特別な考慮事項はありません。

手順

1. 以下のレジストリー URL を許可リストに指定します。

URL	ポート	機能
registry.redhat.io	443、80	コアコンテナーイメージを指定します。
access.redhat.com	443、80	コアコンテナーイメージを指定します。
quay.io	443、80	コアコンテナーイメージを指定します。
cdn.quay.io	443、80	コアコンテナーイメージを指定します。
cdn01.quay.io	443、80	コアコンテナーイメージを指定します。
cdn02.quay.io	443、80	コアコンテナーイメージを指定します。
cdn03.quay.io	443、80	コアコンテナーイメージを指定します。
sso.redhat.com	443、80	**https://console.redhat.com/openshift** サイトは、**sso.redhat.com** からの認証を使用します。

   ホワイトリストでは、**cdn0[1-3].quay.io** の代わりにワイルドカード ***.quay.io** を使用できます。**quay.io** などのサイトを許可リストに追加するには、***.quay.io** などのワイルドカードエントリーを拒否リストに加えないでください。ほとんどの場合、イメージレジストリーはコンテンツ配信ネットワーク (CDN) を使用してイメージを提供します。ファイアウォールがアクセスをブロックすると、最初のダウンロード要求が **cdn01.quay.io** などのホスト名にリダイレクトされるときに、イメージのダウンロードが拒否されます。

2. ビルドに必要な言語またはフレームワークのリソースを提供するサイトを許可リストに指定します。

3. Telemetry を無効にしていない場合は、以下の URL へのアクセスを許可して Red Hat Insights にアクセスできるようにする必要があります。

URL	ポート	機能
cert-api.access.redhat.com	443、80	Telemetry で必須
api.access.redhat.com	443、80	Telemetry で必須
infogw.api.openshift.com	443、80	Telemetry で必須

URL	ポート	機能
console.redhat.com/api/ingress, **cloud.redhat.com/api/ingress**	443、80	Telemetry および **insights-operator** で必須

4. Amazon Web Services (AWS) 、Microsoft Azure、または Google Cloud Platform (GCP) を使用してクラスターをホストする場合、クラウドプロバイダー API およびそのクラウドの DNS を提供する URL へのアクセスを付与する必要があります。

クラウド	URL	ポート	機能
AWS	***.amazonaws.com**	443、80	AWS サービスおよびリソースへのアクセスに必要です。AWS ドキュメントの AWS Service Endpoints を参照し、使用するリージョンを許可するエンドポイントを判別します。
GCP	***.googleapis.com**	443、80	GCP サービスおよびリソースへのアクセスに必要です。GCP ドキュメントの Cloud Endpoints を参照し、API を許可するエンドポイントを判別します。
	accounts.google.com	443、80	GCP アカウントへのアクセスに必要です。
Azure	**management.azure.com**	443、80	Azure サービスおよびリソースへのアクセスに必要です。Azure ドキュメントで Azure REST API Reference を参照し、API を許可するエンドポイントを判別します。
	***.blob.core.windows.net**	443、80	Ignition ファイルのダウンロードに必要です。
	login.microsoftonline.com	443、80	Azure サービスおよびリソースへのアクセスに必要です。Azure ドキュメントで Azure REST API Reference を参照し、API を許可するエンドポイントを判別します。

5. 以下の URL を許可リストに指定します。

URL	ポート	機能

URL	ポート	機能
mirror.openshift.com	443、80	ミラーリングされたインストールのコンテンツおよびイメージへのアクセスに必要。Cluster Version Operator には単一の機能ソースのみが必要ですが、このサイトはリリースイメージ署名のソースでもあります。
storage.googleapis.com/openshift-release	443、80	リリースイメージ署名のソース (ただし、Cluster Version Operator には単一の機能ソースのみが必要)。
***.apps.<cluster_name>.<base_domain>**	443、80	Ingress ワイルドカードをインストール時に設定しない限り、デフォルトのクラスタールートへのアクセスに必要。
quayio-production-s3.s3.amazonaws.com	443、80	AWS で Quay イメージコンテンツにアクセスするために必要。
api.openshift.com	443、80	クラスタートークンの両方が必要であり、クラスターに更新が利用可能かどうかを確認するために必要です。
rhcos-redirector.apps.art.xq1c.p1.openshiftapps.com, **rhcos.mirror.openshift.com**	443、80	Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードするために必要。
console.redhat.com/openshift	443、80	クラスタートークンに必須
registry.access.redhat.com	443、80	**odo** CLI に必須
sso.redhat.com	443、80	**https://console.redhat.com/openshift** サイトは、**sso.redhat.com** からの認証を使用します。

Operator にはヘルスチェックを実行するためのルートアクセスが必要です。具体的には、認証および Web コンソール Operator は 2 つのルートに接続し、ルートが機能することを確認しま

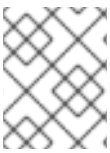
す。クラスター管理者として操作を実行しており、***.apps.<cluster_name>.<base_domain>** を許可しない場合は、これらのルートを許可します。

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**、またはフィールドが空でない場合に **consoles.operator/cluster** オブジェクトの **spec.route.hostname** フィールドに指定されるホスト名。

6. オプションのサードパーティーコンテンツに対する次の URL を許可リストに追加します。

URL	ポート	機能
registry.connect.redhat.com	443、80	すべてのサードパーティーのイメージと認定 Operator に必要です。
rhc4tp-prod-z8cxf-image-registry-us-east-1-evenkyleffocxqvofrk.s3.dualstack.us-east-1.amazonaws.com	443、80	**registry.connect.redhat.com** でホストされているコンテナーイメージにアクセスできます
oso-rhc4tp-docker-registry.s3-us-west-2.amazonaws.com	443、80	Sonatype Nexus、F5 Big IP Operator に必要です。

7. デフォルトの Red Hat Network Time Protocol (NTP) サーバーを使用する場合は、以下の URL を許可します。
   - **1.rhel.pool.ntp.org**
   - **2.rhel.pool.ntp.org**
   - **3.rhel.pool.ntp.org**



注記

デフォルトの Red Hat NTP サーバーを使用しない場合は、プラットフォームの NTP サーバーを確認し、ファイアウォールでこれを許可します。

第18章 インストールの検証

インストール後に、本書の手順を実行して OpenShift Container Platform クラスターのステータスを確認できます。

18.1. インストールログの確認

OpenShift Container Platform インストールログでインストールの概要を確認できます。インストールに成功すると、クラスターへのアクセスに必要な情報はログに追加されます。

前提条件

- インストールホストにアクセスできる。

手順

- インストールホストのインストールディレクトリーにある **.openshift_install.log** ログファイルを確認します。

```
$ cat <install_dir>/.openshift_install.log
```

出力例

以下の例で説明されているように、インストールに成功すると、クラスター認証情報はログの末尾に追加されます。

```
...
time="2020-12-03T09:50:47Z" level=info msg="Install complete!"
time="2020-12-03T09:50:47Z" level=info msg="To access the cluster as the system:admin
user when using 'oc', run 'export KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'"
time="2020-12-03T09:50:47Z" level=info msg="Access the OpenShift web-console here:
https://console-openshift-console.apps.mycluster.example.com"
time="2020-12-03T09:50:47Z" level=info msg="Login to the console with user: \"kubeadmin\",
and password: \"6zYIx-ckbW3-4d2Ne-IWvDF\""
time="2020-12-03T09:50:47Z" level=debug msg="Time elapsed per stage:"
time="2020-12-03T09:50:47Z" level=debug msg="    Infrastructure: 6m45s"
time="2020-12-03T09:50:47Z" level=debug msg="Bootstrap Complete: 11m30s"
time="2020-12-03T09:50:47Z" level=debug msg=" Bootstrap Destroy: 1m5s"
time="2020-12-03T09:50:47Z" level=debug msg=" Cluster Operators: 17m31s"
time="2020-12-03T09:50:47Z" level=info msg="Time elapsed: 37m26s"
```

18.2. イメージのプルソースの表示

ネットワークに制限のないクラスターの場合には、**crictl images** など、ノードでコマンドを使用して、プルしたイメージのソースを表示できます。

ただし、非接続インストールでは、プルされたイメージのソースを表示するには、以下の手順のように CRI-O ログを確認して、**Trying to access** のログエントリーを特定する必要があります。**crictl images** コマンドなど、イメージプルソースを表示する他の方法では、イメージがミラーリングされた場所からプルされている場合でも、ミラーリングされていないイメージ名を表示します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- マスターまたはワーカーノードの CRI-O ログを確認します。

```
$  oc adm node-logs <node_name> -u crio
```

出力例

Trying to access ログエントリーは、イメージがプルされる場所を示します。

```
...
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1366]: time="2021-08-05
10:33:21.594930907Z" level=info msg="Pulling image: quay.io/openshift-release-dev/ocp-
release:4.8.4-ppc64le" id=abcd713b-d0e1-4844-ac1c-474c5b60c07c
name=/runtime.v1alpha2.ImageService/PullImage
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1484]: time="2021-03-17
02:52:50.194341109Z" level=info msg="Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\""
Mar 17 02:52:50 ip-10-0-138-140.ec2.internal crio[1484]: time="2021-03-17
02:52:50.226788351Z" level=info msg="Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\""
...
```

ログは、前述の例のように、イメージのプルソースを 2 回表示する場合があります。

ImageContentSourcePolicy オブジェクトに複数のミラーを一覧表示する場合には、OpenShift Container Platform はイメージを設定に一覧表示されている順序でプルしようとします。以下に例を示します。

```
Trying to access \"li0317gcp1.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\"
Trying to access \"li0317gcp2.mirror-
registry.qe.gcp.devcluster.openshift.com:5000/ocp/release@sha256:1926eae7cacb9c00f142ec
98b00628970e974284b6ddaf9a6a086cb9af7a6c31\"
```

18.3. クラスターのバージョン、ステータス、および更新の詳細の取得

oc get clusterversion コマンドを実行して、クラスターのバージョンおよびステータスを表示できます。ステータスがインストールが進行中であることを示す場合、Operator のステータスで詳細を確認できます。

現在の更新チャネルを一覧表示し、利用可能なクラスターの更新を確認することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

手順

1. クラスターのバージョンと全体のステータスを取得します。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.6.4     True        False         6m25s   Cluster version is 4.6.4
```

この出力例は、クラスターが正常にインストールされていることを示しています。

2. クラスターのステータスがインストールが進行中であることを示す場合、Operator のステータスを確認してより詳細な進捗情報を取得できます。

```
$ oc get clusteroperators.config.openshift.io
```

3. クラスター仕様、更新の可用性、および更新履歴の詳細な要約を取得します。

```
$ oc describe clusterversion
```

4. 現在の更新チャネルを一覧表示します。

```
$ oc get clusterversion -o jsonpath='{.items[0].spec}{"\n"}'
```

出力例

```
{"channel":"stable-4.6","clusterID":"245539c1-72a3-41aa-9cec-72ed8cf25c5c"}
```

5. 利用可能なクラスターの更新を確認します。

```
$ oc adm upgrade
```

出力例

```
Cluster version is 4.6.4

Updates:

VERSION IMAGE
4.6.6   quay.io/openshift-release-dev/ocp-release@sha256:c7e8f18e8116356701bd23ae3a23fb9892dd5ea66c8300662ef30563d7104f39
```

関連情報

- インストールが進行中の場合に Operator のステータスをクエリーする方法についての詳細は、インストール後の Operator ステータスのクエリー について参照してください。
- Operator に関連する問題の調査についての詳細は、Operator の問題のトラブルシューティング について参照してください。

- クラスターの更新についての詳細は、クラスターの更新 を参照してください。
- アップグレードリリースチャネルの概要については、OpenShift Container Platform アップグレードチャネルおよびリリース について参照してください。

18.4. CLI を使用したクラスターノードのステータスのクエリー

インストール後にクラスターノードのステータスを確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. クラスターノードのステータスを一覧表示します。出力に、予想されるすべてのコントロールプレーンおよびコンピュートノードの一覧が表示され、各ノードのステータスが **Ready** であることを確認します。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS   ROLES    AGE   VERSION
compute-1.example.com        Ready    worker   33m   v1.19.0+9f84db3
control-plane-1.example.com  Ready    master   41m   v1.19.0+9f84db3
control-plane-2.example.com  Ready    master   45m   v1.19.0+9f84db3
compute-2.example.com        Ready    worker   38m   v1.19.0+9f84db3
compute-3.example.com        Ready    worker   33m   v1.19.0+9f84db3
control-plane-3.example.com  Ready    master   41m   v1.19.0+9f84db3
```

2. 各クラスターノードの CPU およびメモリーリソースの可用性を確認します。

```
$ oc adm top nodes
```

出力例

```
NAME                         CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
compute-1.example.com        128m         8%     1132Mi          16%
control-plane-1.example.com  801m         22%    3471Mi          23%
control-plane-2.example.com  1718m        49%    6085Mi          40%
compute-2.example.com        935m         62%    5178Mi          75%
compute-3.example.com        111m         7%     1131Mi          16%
control-plane-3.example.com  942m         26%    4100Mi          27%
```

関連情報

- ノードの正常性の確認およびノードの問題の調査方法についての詳細は、ノードの正常性の確認 を参照してください。

18.5. OPENSHIFT CONTAINER PLATFORM WEB コンソールでのクラスターステータスの確認

以下の情報は、OpenShift Container Platform Web コンソールの **Overview** ページで確認できます。

- クラスターの一般的なステータス
- コントロールプレーン、クラスター Operator、およびストレージのステータス
- CPU、メモリー、ファイルシステム、ネットワーク転送、および Pod の可用性
- クラスターの API アドレス、クラスター ID、およびプロバイダーの名前
- クラスターのバージョン情報
- 現在の更新チャネルの詳細や利用可能な更新を含むクラスター更新のステータス
- ノード、Pod、ストレージクラスの詳細を示すクラスターインベントリー、および永続ボリューム要求 (PVC) 情報
- 継続中のクラスターのアクティビティーおよび最近のイベントの一覧

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **Administrator** パースペクティブで、**Home → Overview** に移動します。

18.6. RED HAT OPENSHIFT CLUSTER MANAGER のクラスターステータスの確認

OpenShift Cluster Manager で、クラスターのステータスに関する詳細情報を確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **Administrator** パースペクティブで、**Home → Overview → Details → OpenShift Cluster Manager** に移動し、OpenShift Cluster Manager のクラスターの **Overview** ページを開きます。

注記

または、OpenShift Cluster Manager に直接移動して、使用可能なクラスターのリストからクラスター ID を選択することもできます。

2. **Overview** ページで、クラスターについての以下の情報を確認します。
   - vCPU およびメモリー可用性およびリソースの使用状況

- クラスター ID、ステータス、タイプ、場所、およびプロバイダー名
- ノード数 (ノードタイプ別)
- クラスターバージョンの詳細、クラスターの作成日、およびクラスター所有者の名前
- クラスターのライフサイクルサポートのステータス
- サービスレベルアグリーメント (SLA) のステータス、サブスクリプションユニットタイプ、クラスターの実稼働ステータス、サブスクリプションの義務、サービスレベルなどのサブスクリプション情報
- クラスターの履歴

3. **Monitoring** ページに移動し、以下の情報を確認します。
   - 検出されたすべての問題の一覧
   - 実行されるアラートの一覧
   - クラスター Operator のステータスおよびバージョン
   - クラスターリソースの使用状況
4. **Insights** ページに移動し、Red Hat Insights で提供される以下の情報を確認します。
   - リスクのレベルで分類された、クラスターがさらされる可能性のある問題
   - カテゴリー別のヘルスチェックのステータス

関連情報

- クラスターの潜在的な問題を特定する方法についての詳細は、Insights の使用によるクラスター関連の問題の特定 について参照してください。

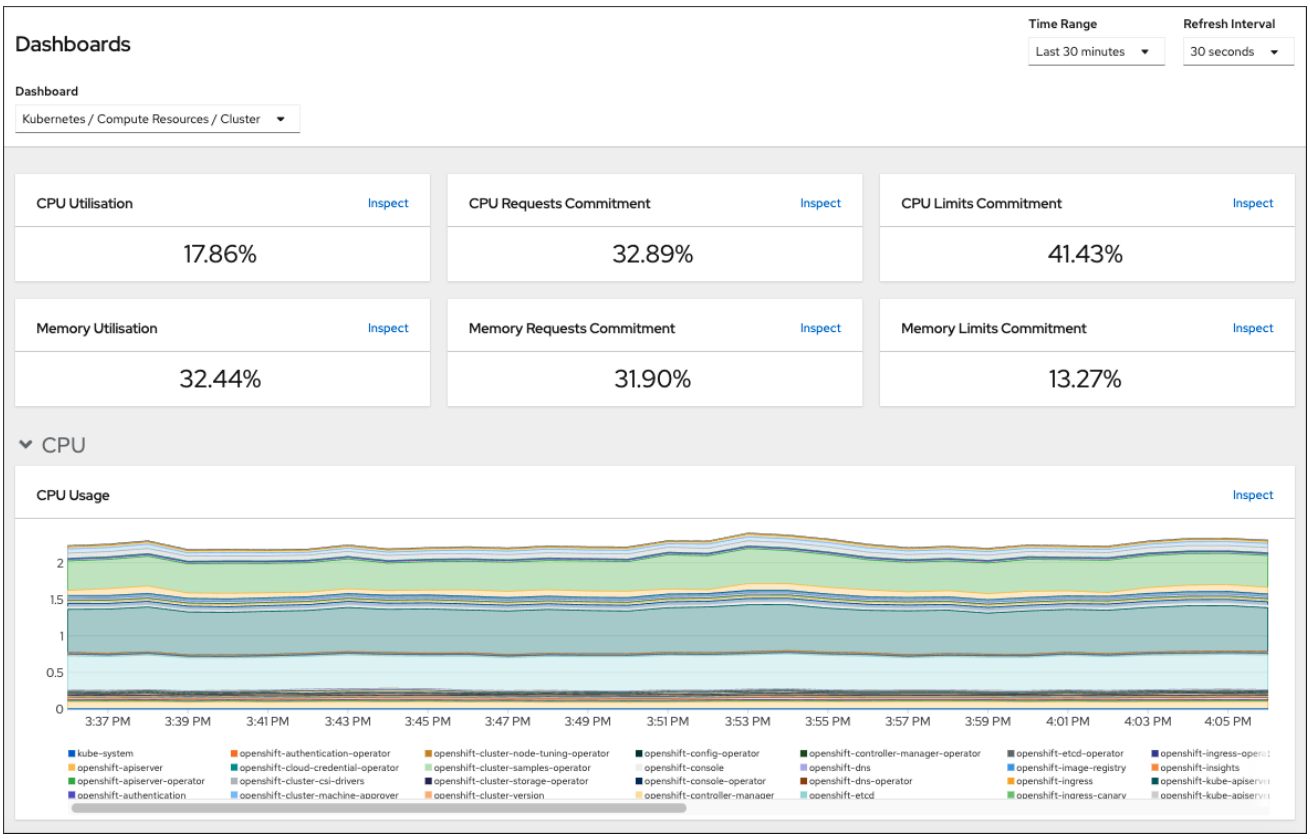
18.7. クラスターリソースの可用性および使用状況の確認

OpenShift Container Platform は、クラスターコンポーネントの状態を理解するのに役立つ包括的なモニタリングダッシュボードのセットを提供します。

Administrator パースペクティブでは、以下を含む OpenShift Container Platform のコアコンポーネントのダッシュボードにアクセスできます。

- etcd
- Kubernetes コンピュートリソース
- Kubernetes ネットワークリソース
- Prometheus
- クラスターおよびノードのパフォーマンスに関連するダッシュボード

図18.1 コンピュートリソースダッシュボードの例



前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**Monitoring → Dashboards** に移動します。
2. **Dashboard** 一覧でダッシュボードを選択します。**etcd** ダッシュボードなどの一部のダッシュボードは、選択時に追加のサブメニューを生成します。
3. 必要に応じて、**Time Range** 一覧でグラフの時間範囲を選択します。
   - 事前定義済みの期間を選択します。
   - **時間範囲** 一覧で **カスタムの時間範囲** を選択して、カスタムの時間範囲を設定します。
     a. **From** および **To** の日付と時間を入力または選択します。
     b. **Save** をクリックして、カスタムの時間範囲を保存します。
4. オプション: **Refresh Interval** を選択します。
5. 特定の項目についての詳細情報を表示するには、ダッシュボードの各グラフにカーソルを合わせます。

関連情報

- OpenShift Container Platform モニターリングスタックの詳細は、Monitoring Overview を参照してください。

18.8. 実行されるアラートの一覧表示

アラートは、定義された条件のセットが OpenShift Container Platform クラスターで true の場合に通知を提供します。OpenShift Container Platform Web コンソールでアラート UI を使用して、クラスターで実行されているアラートを確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **Administrator** パースペクティブで、**Monitoring → Alerting → Alerts** ページに移動します。
2. **Severity**、 **State**、および **Source** が含まれる、実行されているアラートを確認します。
3. **Alert Details** ページで詳細情報を表示するためにアラートを選択します。

関連情報

- OpenShift Container Platform のアラートについての詳細は、アラートの管理 を参照してください。

18.9. 次のステップ

- クラスターのインストールに関する問題がある場合には、インストールのトラブルシューティング を参照してください。
- OpenShift Container Platform のインストール後に、クラスターをさらに拡張し、カスタマイズ できます。

第19章 インストールの問題のトラブルシューティング

OpenShift Container Platform のインストールした場合のトラブルシューティングのために、ブートストラップおよびコントロールプレーン (またはマスター) マシンからログを収集できます。インストールプログラムからデバッグ情報を取得することもできます。

19.1. 前提条件

- OpenShift Container Platform クラスターのインストールを試みたが、インストールに失敗している。

19.2. 失敗したインストールのログの収集

SSH キーをインストールプログラムに指定している場合、失敗したインストールについてのデータを収集することができます。

注記

実行中のクラスターからログを収集する場合とは異なるコマンドを使用して失敗したインストールについてのログを収集します。実行中のクラスターからログを収集する必要がある場合は、**oc adm must-gather** コマンドを使用します。

前提条件

- OpenShift Container Platform のインストールがブートストラッププロセスの終了前に失敗している。ブートストラップノードは実行中であり、SSH でアクセスできる。
- **ssh-agent** プロセスはコンピューター上でアクティブであり、**ssh-agent** プロセスとインストールプログラムの両方に同じ SSH キーを提供している。
- 独自にプロビジョニングしたインフラストラクチャーにクラスターのインストールを試行した場合には、ブートストラップおよびコントロールプレーンノード (別名マスターノード) の完全修飾ドメイン名がある。

手順

1. ブートストラップおよびコントロールプレーンマシンからインストールログを収集するために必要なコマンドを生成します。
   - インストーラーでプロビジョニングされたインフラストラクチャーを使用する場合は、インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

     ```
     $ ./openshift-install gather bootstrap --dir <installation_directory> 1
     ```

     1 **installation_directory** は、**./openshift-install create cluster** を実行した際に指定したディレクトリーです。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

     インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムは、ホスト名または IP アドレスを指定しなくてもよいようにクラスターについての情報を保存します。

- 各自でプロビジョニングしたインフラストラクチャーを使用した場合は、インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install gather bootstrap --dir <installation_directory> \ 1
  --bootstrap <bootstrap_address> \ 2
  --master <master_1_address> \ 3
  --master <master_2_address> \ 4
  --master <master_3_address>" 5
```

1. **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

2. **<bootstrap_address>** は、クラスターのブートストラップマシンの完全修飾ドメイン名または IP アドレスです。

3 4 5 クラスター内のそれぞれのコントロールプレーン (またはマスター) マシンについては、**<master_*_address>** をその完全修飾ドメイン名または IP アドレスに置き換えます。

注記

デフォルトクラスターには 3 つのコントロールプレーンマシンが含まれます。クラスターが使用する数にかかわらず、表示されるようにすべてのコントロールプレーンマシンを一覧表示します。

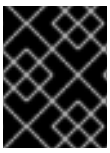
出力例

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
<timestamp>.tar.gz"
```

インストールの失敗についての Red Hat サポートケースを作成する場合は、圧縮したログをケースに含めるようにしてください。

19.3. ホストへの SSH アクセスによるログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。



重要

デフォルトでは、OpenShift Container Platform ノードへの SSH アクセスは、Red Hat Open Stack Platform (RHOSP) ベースのインストールでは無効になっています。

前提条件

- ホストへの SSH アクセスがあること。

手順

1. 以下を実行し、**journalctl** コマンドを使用してブートストラップホストから **bootkube.service** サービスログを収集します。

```
$ journalctl -b -f -u bootkube.service
```

2. podman ログを使用して、ブートストラップホストのコンテナーログを収集します。これは、ホストからすべてのコンテナーログを取得するためにループで表示されます。

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. または、以下を実行し、**tail** コマンドを使用してホストのコンテナーログを収集します。

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. 以下を実行し、**journalctl** コマンドを使用して **kubelet.service** および **crio.service** サービスログをマスターホストおよびワーカーホストから収集します。

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. 以下を実行し、**tail** コマンドを使用してマスターホストおよびワーカーホストのコンテナーログを収集します。

```
$ sudo tail -f /var/log/containers/*
```

19.4. ホストへの SSH アクセスを使用しないログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

ノードへの SSH アクセスがない場合は、システムジャーナルにアクセスし、ホストで生じていることを調査できます。

前提条件

- OpenShift Container Platform のインストールが完了している。
- API サービスが機能している。
- システム管理者権限がある。

手順

1. 以下を実行し、**/var/log** の下にある **journald** ユニットログにアクセスします。

```
$ oc adm node-logs --role=master -u kubelet
```

2. 以下を実行し、**/var/log** の下にあるホストファイルのパスにアクセスします。

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

19.5. インストールプログラムからのデバッグ情報の取得

以下のアクションのいずれかを使用して、インストールプログラムからデバッグ情報を取得できます。

- 非表示の **.openshift_install.log** ファイルで過去のインストールからのデバッグ情報を確認します。たとえば、 以下を入力します。

```
$ cat ~/<installation_directory>/.openshift_install.log 1
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。

- インストールプログラムが含まれるディレクトリーに切り替え、**--log-level=debug** でこれを再実行します。

```
$ ./openshift-install create cluster --dir <installation_directory> --log-level debug 1
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。

19.6. OPENSHIFT CONTAINER PLATFORM クラスターの再インストール

OpenShift Container Platform のインストールに失敗して問題をデバッグおよび解決できない場合は、新しい OpenShift Container Platform クラスターのインストールを検討してください。完全に消去してから、インストールプロセスを開始しなおしてください。ユーザープロビジョニングインフラストラクチャー (UPI) をインストールする場合は、クラスターを手動で破棄し、関連するすべてのリソースを削除する必要があります。次の手順は、インストーラーでプロビジョニングされたインフラストラクチャー (IPI) のインストール用です。

手順

1. クラスターを破棄し、インストールディレクトリー内の非表示のインストーラー状態ファイルなども含めて、クラスターに関連付けられているすべてのリソースを削除します。

```
$ ./openshift-install destroy cluster --dir <installation_directory> 1
```

1 **installation_directory** は、**./openshift-install create cluster** を実行した際に指定したディレクトリーです。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

2. クラスターを再インストールする前に、インストールディレクトリーを削除してください。

```
$ rm -rf <installation_directory>
```

3. OpenShift Container Platform クラスターの新規インストール手順に従います。

関連情報

- OpenShift Container Platform クラスターのアンインストール

第20章 FIPS 暗号のサポート

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールすることができます。

クラスター内の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの場合、この変更は、ユーザーがクラスターのデプロイメント時に変更できるクラスターオプションを制御する **install-config.yaml** ファイルのオプションのステータスに基づいてマシンがデプロイされる際に適用されます。Red Hat Enterprise Linux (RHEL) マシンでは、ワーカーマシンとして使用する予定のマシンにオペレーティングシステムをインストールする場合に FIPS モードを有効にする必要があります。これらの設定方法により、クラスターが FIPS コンプライアンス監査の要件を満たすことを確認できます。初期システムの起動前は、FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号パッケージのみが有効になります。

FIPS はクラスターが使用するオペレーティングシステムの初回の起動前に有効にされている必要があり、クラスターをデプロイしてから FIPS を有効にすることはできません。

20.1. OPENSHIFT CONTAINER PLATFORM での FIPS 検証

OpenShift Container Platform は、それが使用するオペレーティングシステムのコンポーネント用に RHEL および RHCOS 内の特定の FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用します。RHEL7 core crypto components を参照してください。たとえば、ユーザーが OpenShift Container Platform クラスターおよびコンテナーに対して SSH を実行する場合、それらの接続は適切に暗号化されます。

OpenShift Container Platform コンポーネントは Go で作成され、Red Hat の golang コンパイラーを使用してビルドされます。クラスターの FIPS モードを有効にすると、暗号署名を必要とするすべての OpenShift Container Platform コンポーネントは RHEL および RHCOS 暗号ライブラリーを呼び出します。

表20.1 OpenShift Container Platform 4.8 における FIPS モード属性および制限

属性	制限
RHEL 7 オペレーティングシステムの FIPS サポート。	FIPS 実装は、ハッシュ関数を計算し、そのハッシュに基づくキーを検証する単一の機能を提供しません。この制限については、今後の OpenShift Container Platform リリースで継続的に評価され、改善されます。
CRI-O ランタイムの FIPS サポート。	
OpenShift Container Platform サービスの FIPS サポート。	
RHEL 7 および RHCOS バイナリーおよびイメージから取得される FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号化モジュールおよびアルゴリズム。	
FIPS と互換性のある golang コンパイラーの使用。	TLS FIPS サポートは完全に実装されていませんが、今後の OpenShift Container Platform リリースで予定されています。

属性	制限
複数のアーキテクチャー間の FIPS サポート。	現時点で、FIPS は **x86_64** アーキテクチャーを使用する OpenShift Container Platform デプロイメントでのみサポートされています。

20.2. クラスターが使用するコンポーネントでの FIPS サポート

OpenShift Container Platform クラスター自体は FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用しますが、OpenShift Container Platform クラスターをサポートするシステムが暗号化の FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用していることを確認してください。

20.2.1. etcd

etcd に保存されるシークレットが FIPS 検証済み/進行中のモジュール (Modules in Process) の暗号を使用できるようにするには、ノードを FIPS モードで起動します。クラスターを FIPS モードでインストールした後に、FIPS 承認の **aes cbc** 暗号アルゴリズムを使用して etcd データを暗号化 できます。

20.2.2. ストレージ

ローカルストレージの場合は、RHEL が提供するディスク暗号化または RHEL が提供するディスク暗号化を使用する Container Native Storage を使用します。RHEL が提供するディスク暗号を使用するボリュームにすべてのデータを保存し、クラスター用に FIPS モードを有効にすることで、移動しないデータと移動するデータまたはネットワークデータは FIPS の検証済み/進行中のモジュール (Modules in Process) の暗号化によって保護されます。ノードのカスタマイズ で説明されているように、各ノードのルートファイルシステムを暗号化するようにクラスターを設定できます。

20.2.3. ランタイム

コンテナーに対して FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号モジュールを使用しているホストで実行されていることを認識させるには、CRI-O を使用してランタイムを管理します。CRI-O は FIPS モードをサポートします。このモードでは、コンテナーが FIPS モードで実行されていることを認識できるように設定されます。

20.3. FIPS モードでのクラスターのインストール

FIPS モードでクラスターをインストールするには、必要なインフラストラクチャーにカスタマイズされたクラスターをインストールする方法についての説明に従ってください。クラスターをデプロイする前に、**fips: true** を **install-config.yaml** ファイルに設定していることを確認します。

- Amazon Web Services
- Microsoft Azure
- ベアメタル
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

注記

Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。

AES CBC 暗号化を etcd データストアに適用するには、クラスターのインストール後に etcd データの暗号化 プロセスに従ってください。

RHEL ノードをクラスターに追加する場合は、初回の起動前に FIPS モードをマシン上で有効にしていることを確認してください。RHEL 7 ドキュメントの RHEL コンピュートマシンの OpenShift Container Platform クラスターへの追加 および FIPS モードの有効化 を参照してください。