OpenShift Container Platform 4.12

IBM Power へのインストール

IBM Power への OpenShift Container Platform のインストール

Last Updated: 2025-11-12

OpenShift Container Platform 4.12 IBM Power へのインストール

IBM Power への OpenShift Container Platform のインストール

Legal Notice

Copyright © 2025 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

このドキュメントでは、IBM Power に OpenShift Container Platform をインストールする方法を説明します。

Table of Contents

第1章 IBM POWER へのインストールの準備 **3**
1.1. 前提条件 3
1.2. IBM POWER に OPENSHIFT CONTAINER PLATFORM をインストールする方法の選択 3

第2章 クラスターの IBM POWER へのインストール **4**
2.1. 前提条件 4
2.2. OPENSHIFT CONTAINER PLATFORM のインターネットアクセス 4
2.3. USER-PROVISIONED INFRASTRUCTURE を使用したクラスターの要件 5
2.4. USER-PROVISIONED INFRASTRUCTURE の準備 19
2.5. USER-PROVISIONED INFRASTRUCTURE の DNS 解決の検証 21
2.6. クラスターノード SSH アクセス用の鍵ペアの生成 23
2.7. インストールプログラムの取得 25
2.8. バイナリーのダウンロードによる OPENSHIFT CLI のインストール 26
2.9. インストール設定ファイルの手動作成 28
2.10. CLUSTER NETWORK OPERATO の設定 44
2.11. KUBERNETES マニフェストおよび IGNITION 設定ファイルの作成 52
2.12. RHCOS のインストールおよび OPENSHIFT CONTAINER PLATFORM ブートストラッププロセスの開始 54
2.13. ブートストラッププロセスの完了まで待機する 67
2.14. CLI の使用によるクラスターへのログイン 68
2.15. マシンの証明書署名要求の承認 68
2.16. OPERATOR の初期設定 71
2.17. USER-PROVISIONED INFRASTRUCTURE でのインストールの完了 75
2.18. OPENSHIFT CONTAINER PLATFORM の TELEMETRY アクセス 77
2.19. 次のステップ 77

第3章 ネットワークが制限された環境での IBM POWER へのクラスターのインストール **78**
3.1. 前提条件 78
3.2. ネットワークが制限された環境でのインストールについて 78
3.3. OPENSHIFT CONTAINER PLATFORM のインターネットアクセス 79
3.4. USER-PROVISIONED INFRASTRUCTURE を使用したクラスターの要件 79
3.5. USER-PROVISIONED INFRASTRUCTURE の準備 93
3.6. USER-PROVISIONED INFRASTRUCTURE の DNS 解決の検証 95
3.7. クラスターノード SSH アクセス用の鍵ペアの生成 97
3.8. インストール設定ファイルの手動作成 99
3.9. CLUSTER NETWORK OPERATO の設定 116
3.10. KUBERNETES マニフェストおよび IGNITION 設定ファイルの作成 124
3.11. RHCOS のインストールおよび OPENSHIFT CONTAINER PLATFORM ブートストラッププロセスの開始 126
3.12. ブートストラッププロセスの完了まで待機する 139
3.13. CLI の使用によるクラスターへのログイン 140
3.14. マシンの証明書署名要求の承認 140
3.15. OPERATOR の初期設定 143
3.16. USER-PROVISIONED INFRASTRUCTURE でのインストールの完了 147
3.17. 次のステップ 150

第1章 IBM POWER へのインストールの準備

1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。

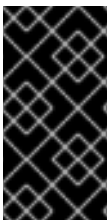
1.2. IBM POWER に OPENSHIFT CONTAINER PLATFORM をインストールする方法の選択

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

- **クラスターの IBM Power へのインストール**: OpenShift Container Platform を独自にプロビジョニングする IBM Power インフラストラクチャーにインストールできます。
- **ネットワークが制限された環境での IBM Power へのクラスターのインストール**: インストールリリースコンテンツの内部ミラーを使用して、独自にプロビジョニングする IBM Power インフラストラクチャーに OpenShift Container Platform をインストールできます。この方法を使用して、ソフトウェアコンポーネントを取得するためにアクティブなインターネット接続を必要としないクラスターをインストールできます。また、このインストール方法を使用して、クラスターが外部コンテンツに対する組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

第2章 クラスターの IBM POWER へのインストール

OpenShift Container Platform バージョン 4.12 では、独自にプロビジョニングする IBM Power インフラストラクチャーにクラスターをインストールできます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- インストールプロセスを開始する前に、既存のインストールファイルを取り除く必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- OpenShift Data Foundation またはクラスターでサポートされているその他のストレージプロトコルを使用して永続ストレージをプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**ReadWriteMany** のアクセスモードで永続ストレージを設定する必要があります。
- ファイアウォールを使用する場合は、クラスターがアクセスを必要とするサイトを許可するようにファイアウォールを設定する必要がある。

注記

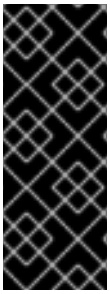
プロキシーを設定する場合は、このサイトリストも確認してください。

2.2. OPENSHIFT CONTAINER PLATFORM のインターネットアクセス

OpenShift Container Platform 4.12 では、クラスターをインストールするためにインターネットアクセスが必要になります。

インターネットへのアクセスは以下を実行するために必要です。

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



重要

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

2.3. USER-PROVISIONED INFRASTRUCTURE を使用したクラスターの要件

user-provisioned infrastructure を含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

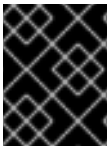
このセクションでは、user-provisioned infrastructure に OpenShift Container Platform をデプロイする要件を説明します。

2.3.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表2.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンに別の物理ホストを使用します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

2.3.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表2.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	2	16 GB	100 GB	300
コントロールプレーン	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU
2. OpenShift Container Platform と Kubernetes は、ディスクのパフォーマンスの影響を受けるため、特にコントロールプレーンノードの etcd には、より高速なストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

関連情報

- ストレージの最適化

2.3.3. IBM Power の最小要件

OpenShift Container Platform バージョン 4.12 は、以下の IBM ハードウェアにインストールできます。

- IBM Power9、または Power10 プロセッサーベースのシステム

注記

すべての IBM POWER8 モデル、IBM POWER9 AC922、IBM POWER9 IC922、および IBM POWER9 LC922 の RHCOS 機能のサポートは非推奨になりました。これらのハードウェアモデルは、OpenShift Container Platform 4.12 で引き続き完全にサポートされます。ただし、Red Hat は、新しいハードウェアモデルを使用することを推奨します。

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 つの IBM Power ベアメタルサーバーまたは 6 つの LPAR

オペレーティングシステム要件

- IBM Power9、または Power10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能
- 共有イーサネットアダプターを使用して仮想 I/O サーバーで利用可能
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 100GB / 16GB
- OpenShift Container Platform コンピュートマシン用に 100 GB / 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 100 GB / 16 GB

2.3.4. 推奨される IBM Power システム要件

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 つの IBM Power ベアメタルサーバーまたは 6 つの LPAR

オペレーティングシステム要件

- IBM Power9、または Power10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能

- 共有イーサネットアダプターを使用して仮想 I/O サーバーで利用可能
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 120 GB / 32 GB
- OpenShift Container Platform コンピュートマシン用に 120 GB / 32 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 120 GB / 16 GB

2.3.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求されるサービング証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

2.3.6. user-provisioned infrastructure のネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスが user-provisioned infrastructure で利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始** のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

2.3.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定され

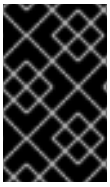
ます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

2.3.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

このセクションでは、必要なポートの詳細を説明します。



重要

インターネットに接続された OpenShift Container Platform 環境では、プラットフォームコンテナーのイメージをプルし、Red Hat にテレメトリーデータを提供するために、すべてのノードがインターネットにアクセスできる必要があります。

表2.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット

プロトコル	ポート	説明
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表2.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表2.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

user-provisioned infrastructure の NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

2.3.7. user-provisioned DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

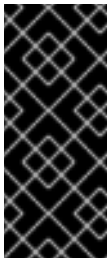
DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

注記

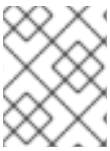
各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**user-provisioned infrastructure に関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、user-provisioned OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表2.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。



注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**user-provisioned infrastructure の DNS 解決の検証**のセクションを参照してください。

2.3.7.1. user-provisioned クラスターの DNS 設定の例

このセクションでは、user-provisioned infrastructure に OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

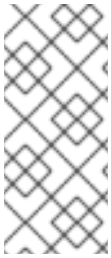
user-provisioned クラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、user-provisioned クラスターの名前解決の A レコードの例を示しています。

例2.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。



注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4 ブートストラップマシンの名前解決を提供します。

5 6 7 コントロールプレーンマシンの名前解決を提供します。

8 9 コンピュートマシンの名前解決を提供します。

user-provisioned クラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、user-provisioned クラスターの逆引き名前解決の PTR レコードの例を示しています。

例2.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

2.3.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

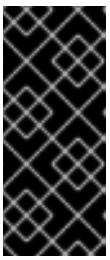
OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



重要

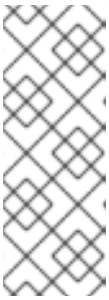
API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表2.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとのプロービングで、2 回連続成功すると正常、3 回連続失敗すると異常と判断する設定は、十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表2.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

2.3.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例2.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
  retries                 3
  timeout http-request    10s
```

```
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup (2)
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (4)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (5)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (6)
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

(1) ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

(2)(4) ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

(3) ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

(5) ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

(6) ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

2.4. USER-PROVISIONED INFRASTRUCTURE の準備

user-provisioned infrastructure に OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要を説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**user-provisioned infrastructure を使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x Tested Integrations ページを確認した。
- **user-provisioned infrastructure を使用したクラスターの要件** セクションで説明されているインフラストラクチャーの要件を確認した。

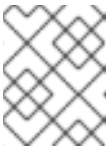
手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。
   a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。
   b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始** のセクションを参照してください。

c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項の詳細は、**DHCP を使用したクラスターノードのホスト名の設定** セクションを参照してください。

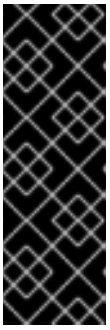


注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**user-provisioned infrastructure のネットワーク要件** のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**user-provisioned infrastructure のネットワーク要件** のセクションを参照してください。



重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

   a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

   b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
   OpenShift Container Platform DNS 要件の詳細は、**user-provisioned DNS 要件** のセクションを参照してください。

5. DNS 設定を検証します。

   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。

DNS 検証手順の詳細は、**user-provisioned infrastructure の DNS 解決の検証**のセクションを参照してください。

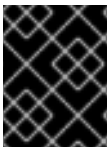
6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**user-provisioned infrastructure の負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

2.5. USER-PROVISIONED INFRASTRUCTURE の DNS 解決の検証

OpenShift Container Platform を user-provisioned infrastructure にインストールする前に、DNS 設定を検証できます。



重要

このセクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- user-provisioned infrastructure に必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
   ```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

   出力例

   ```
   api-int.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

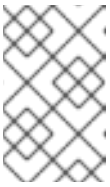
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

出力例

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
```

1 Kubernetes 内部 API のレコード名を指定します。

2 Kubernetes API のレコード名を指定します。



注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

出力例

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

2.6. クラスターノード SSH アクセス用の鍵ペアの生成

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

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

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

重要

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

注記

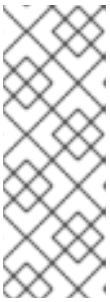
AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

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

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

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



注記

FIPS で検証済みまたは進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64**、**ppc64le**、および **s390x** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

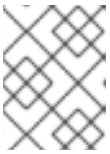
2. SSH 公開鍵を表示します。

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

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

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

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



注記

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

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

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

出力例

```
Agent pid 31874
```

注記

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

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

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

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

出力例

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

次のステップ

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

2.7. インストールプログラムの取得

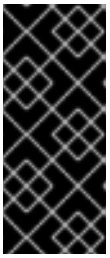
OpenShift Container Platform をインストールする前に、インストールに使用しているホストにインストールファイルをダウンロードします。

前提条件

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

手順

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



重要

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

重要

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

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

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

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

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

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.12 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.12 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

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

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

```
$ echo $PATH
```

検証

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.12 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
   PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

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

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

macOS への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.12 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。

注記

macOS arm64 の場合は、**OpenShift v4.12 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。

PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

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

```
$ oc <command>
```

2.9. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

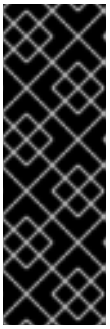
前提条件

- インストールプログラムで使用するための SSH 公開鍵がローカルマシン上に存在する。この鍵は、デバッグや障害復旧のために、クラスターノードへの SSH 認証に使用できます。
- OpenShift Container Platform インストールプログラムとクラスターのプルシークレットを取得している。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

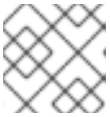
```
$ mkdir <installation_directory>
```



重要

このディレクトリーは必ず作成してください。ブートストラップ X.509 証明書などの一部のインストールアセットは、有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してください。

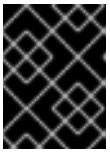
2. 提供されているサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、ファイルを **<installation_directory>** に保存します。



注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. 多くのクラスターのインストールに使用できるように、**install-config.yaml** ファイルをバックアップします。

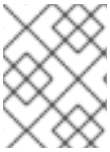


重要

インストールプロセスの次のステップで **install-config.yaml** ファイルを使用するため、今すぐこのファイルをバックアップしてください。

2.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。



注記

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

2.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表2.9 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**<metadata.name>.<baseDomain>** 形式を使用する **baseDomain** と **metadata.name** パラメーターの値を組み合わせたものです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト

パラメーター	説明	値
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

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

2.9.1.2. ネットワーク設定パラメーター

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

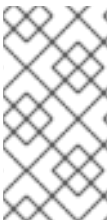
- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。
- Red Hat OpenShift Networking OpenShift SDN ネットワークプラグインを使用する場合、IPv4 アドレスファミリーのみがサポートされます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。

- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```



注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表2.10 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。**OpenShiftSDN** は、全 Linux ネットワーク用の CNI プラグインです。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。

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

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

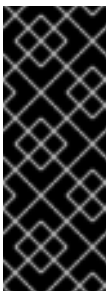
2.9.1.3. オプションの設定パラメーター

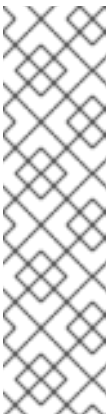
オプションのインストール設定パラメーターは、以下の表で説明されています。

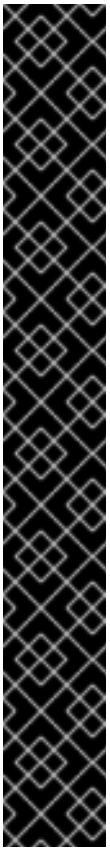
表2.11 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
capabilities	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳細は、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities.baselineCapabilitySet	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	文字列
capabilities.additionalEnabledCapabilities	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列

パラメーター	説明	値
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時マルチスレッドはマシンのコアのパフォーマンスを上げるために有効化されます。 **重要** 同時マルチスレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**alibabacloud**、**aws**、**azure**, **gcp**、**ibmcloud**、**nutanix**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
featureSet	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時マルチスレッドはマシンのコアのパフォーマンスを上げるために有効化されます。  **重要** 同時マルチスレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**alibabacloud**、**aws**、**azure**, **gcp**、**ibmcloud**、**nutanix**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。  **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。 **注記** AWS アカウントでサービスコントロールポリシー (SCP) が有効になっている場合は、**credentialsMode** パラメーターを **Mint**、**Passthrough** または **Manual** に設定する必要があります。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。FIPS 検証済み/Modules In Process 暗号ライブラリーの使用は、**x86_64**、**ppc64le**、および **s390x** アーキテクチャー上の OpenShift Container Platform デプロイメントでのみサポートされます。 **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。

パラメーター	説明	値
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

2.9.2. IBM Power のサンプル install-config.yaml ファイル

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

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
```

```
  replicas: 0 4
  architecture: ppc64le
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture: ppc64le
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、**compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

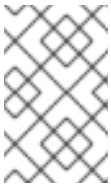
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform を user-provisioned infrastructure にインストールする場合は、この値を **0** に設定する必要があります。installer-provisioned installation では、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。user-provisioned installation では、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



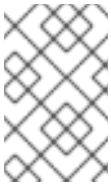
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされている値は **OVNKubernetes** と **OpenShiftSDN** です。デフォルトの値は **OVNkubernetes** です。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Power インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

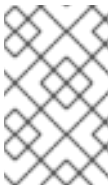
14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。FIPS 検証済み/Modules In Process 暗号ライブラリーの使用は、**x86_64**、**ppc64le**、および **s390x** アーキテクチャー上の OpenShift Container Platform デプロイメントでのみサポートされます。

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

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

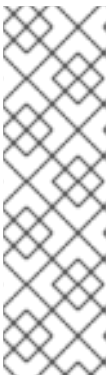
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

2.9.3. インストール時のクラスター全体のプロキシーの設定

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

前提条件

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



注記

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

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

手順

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

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション: **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-bundle** 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。

注記

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

注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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

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

注記

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

2.9.4. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで構成されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

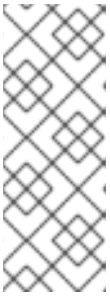
前提条件

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

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```



注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform を user-provisioned infrastructure にインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。installer-provisioned installation では、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、user-provisioned installation には適用されません。

3 ノードのクラスターのインストールで、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**user-provisioned infrastructure の負荷分散要件** のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、

アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

2.10. CLUSTER NETWORK OPERATO の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN や OVN-Kubernetes などのクラスターネットワークプラグイン。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

2.10.1. Cluster Network Operator 設定オブジェクト

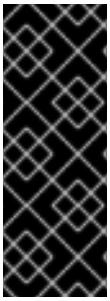
Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表2.12 Cluster Network Operator 設定オブジェクト

フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。

spec.clusterNetwork	**array**	Pod IP アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。

フィールド	型	説明
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。



重要

複数のネットワークにオブジェクトをデプロイする必要があるクラスターの場合は、**install-config.yaml** ファイルで定義されている各ネットワークタイプの **clusterNetwork.hostPrefix** パラメーターに、必ず同じ値を指定してください。**clusterNetwork.hostPrefix** パラメーターにそれぞれ異なる値を設定すると、OVN-Kubernetes ネットワークプラグインに影響が及び、異なるノード間のオブジェクトトラフィックをプラグインが効果的にルーティングできなくなる可能性があります。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表2.13 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。 **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。

フィールド	型	説明
openshiftSDNConfig	**object**	このオブジェクトは、OpenShift SDN ネットワークプラグインに対してのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OpenShift SDN ネットワークプラグインの設定
以下の表では、OpenShift SDN ネットワークプラグインの設定フィールドを説明します。

表2.14 openshiftSDNConfig オブジェクト

フィールド	型	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU をオーバーライドする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 クラスターインストール時またはインストール後のタスクとして値を設定できます。詳細は、OpenShift Container Platform Networking ドキュメントの "Changing the MTU for the cluster network" を参照してください。

フィールド	型	説明
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドを説明します。

表2.15 ovnKubernetesConfig オブジェクト

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU をオーバーライドする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 暗号化を有効にするために空のオブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。

フィールド	型	説明
gatewayConfig	**object**	オプション: Egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** Egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

フィールド	型	説明
v4InternalSubnet	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 このフィールドは、インストール後に変更できません。	デフォルト値は **100.64.0.0/16** です。

フィールド	型	説明
v6InternalSubnet	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 このフィールドは、インストール後に変更できません。	デフォルト値は **fd98::/48** です。

表2.16 policyAuditConfig オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。

フィールド	型	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表2.17 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの Egress トラフィックを送信するには、このフィールドを **true** に設定します。 **注記** OpenShift Container Platform 4.12 では、egress IP はプライマリーインターフェイスのみに割り当てられます。そのため、**routingViaHost** を **true** に設定すると、OpenShift Container Platform 4.12 で egress IP は機能しません。 インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、Egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、Egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを **true** に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
```

```
    type: OVNKubernetes
    ovnKubernetesConfig:
      mtu: 1400
      genevePort: 6081
      ipsecConfig: {}
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表2.18 **kubeProxyConfig** オブジェクト

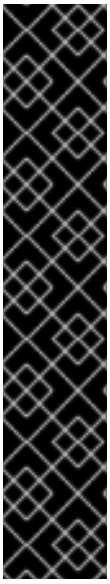
フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値:
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

2.11. KUBERNETES マニフェストおよび IGNITION 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。



重要

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

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。Linux バージョンのインストールプログラム (アーキテクチャーポストフィックスなし) は、ppc64le でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

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

1 **<installation_directory>** には、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。

重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

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

1 **<installation_directory>** には、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

2.12. RHCOS のインストールおよび OPENSHIFT CONTAINER PLATFORM ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Power インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

ISO イメージまたはネットワーク PXE ブートを使用する手順を実行して RHCOS をマシンにインストールできます。

2.12.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定** のセクションを確認している。

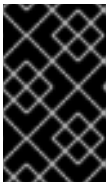
手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

   コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

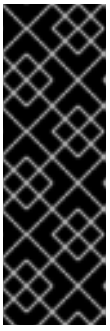
4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンド

の出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

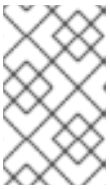
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **core** ユーザーにはインストールを実行するために必要な root 特権がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。

重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

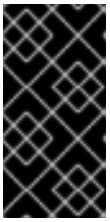
9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

11. 継続してクラスターの他のマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

2.12.1.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションを説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

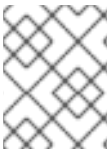
2.12.1.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法を説明しています。



注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

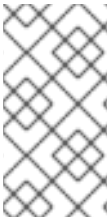
次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 **IP** アドレスの設定

IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない **IP** アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの **DHCP** の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、使用されていない **enp2s0** では DHCP が無効になっています。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 **IP** 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの **VLAN** の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の **DNS** サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切りリストを表します。**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **bond=** を使用してボンディングされたインターフェイスを作成する場合は、ボンディングされたインターフェイスの IP アドレスの割り当て方法やその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

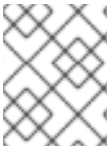
```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team=name[:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1, em2**) のコンマ区切りリストを表します。



注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、こちらの Red Hat ナレッジベース記事 を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

2.12.2. PXE ブートを使用した RHCOS のインストール

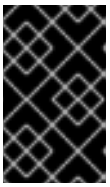
PXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定** のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

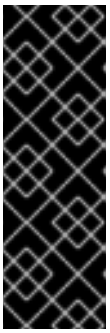
3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法

は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



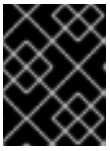
重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、**initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE インストールを設定し、インストールを開始します。
以下の例で示されるご使用の環境のメニューエントリーを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。

重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

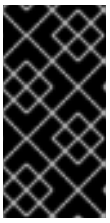
8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

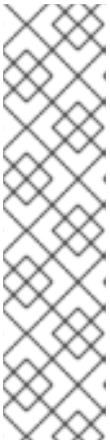
10. クラスターのマシンの作成を続行します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。



注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

2.12.3. RHCOS のカーネル引数でのマルチパスの有効化

OpenShift Container Platform 4.9 以降では、インストール時に、プロビジョニングしたノードのマルチパスを有効にできます。RHCOS は、プライマリーディスクでのマルチパスをサポートします。マルチパス化により、ハードウェア障害に強力な耐障害性に利点が追加され、ホストの可用性が向上されます。

初回のクラスターの作成時に、カーネル引数をすべてのマスターまたはワーカーノードに追加しないといけない場合があります。カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入できます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

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

2. カーネル引数をワーカーまたコントロールプレーンノードに追加するかどうかを決定します。

- マシン設定ファイルを作成します。たとえば、**master** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-master-kargs-mpath.yaml** を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "master"
  name: 99-master-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

3. ワーカーノードでマルチパスを有効にするには、以下を実行します。

- マシン設定ファイルを作成します。たとえば、**worker** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-worker-kargs-mpath.yaml** を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "worker"
  name: 99-worker-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

クラスターの作成を継続できます。

重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

MPIO が失敗する場合は、bootlist コマンドを使用して、別の論理デバイス名でブートデバイスリストを更新します。このコマンドは、ブートリストを表示し、システムが通常モードで起動したときのブートデバイスを指定します。

a. ブートリストを表示し、システムが通常モードで起動した場合に使用可能なブートデバイスを指定するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o
sda
```

b. 通常モードのブートリストを更新し、別のデバイス名を追加するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
sdc
sdd
sde
```

元のブートディスクパスがダウンすると、ノードは通常のブートデバイスリストに登録された別のデバイスから再起動します。

2.13. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。
- お使いのマシンでインターネットに直接アクセスできるか、HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

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

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

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.25.0 up
```

```
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

2.14. CLI の使用によるクラスターへのログイン

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

前提条件

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

手順

1. **kubeadmin** 認証情報をエクスポートします。

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

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

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

```
$ oc whoami
```

出力例

```
system:admin
```

2.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンに対して 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

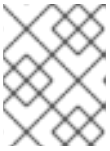
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.25.0
master-1  Ready    master  63m  v1.25.0
master-2  Ready    master  64m  v1.25.0
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

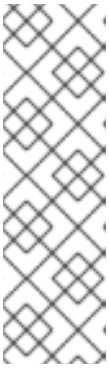
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードの ID を確認します。

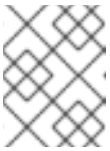
- それらを個別に承認するには、それぞれの有効な CSR に以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

–

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR に以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.25.0
   master-1  Ready     master  73m  v1.25.0
   master-2  Ready     master  74m  v1.25.0
   worker-0  Ready     worker  11m  v1.25.0
   worker-1  Ready     worker  11m  v1.25.0
   ```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

2.16. OPERATOR の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

   ```
   $ watch -n5 oc get clusteroperators
   ```

■

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False      19m
baremetal                                  4.12.0    True        False         False      37m
cloud-credential                           4.12.0    True        False         False      40m
cluster-autoscaler                         4.12.0    True        False         False      37m
config-operator                            4.12.0    True        False         False      38m
console                                    4.12.0    True        False         False      26m
csi-snapshot-controller                    4.12.0    True        False         False      37m
dns                                        4.12.0    True        False         False      37m
etcd                                       4.12.0    True        False         False      36m
image-registry                             4.12.0    True        False         False      31m
ingress                                    4.12.0    True        False         False      30m
insights                                   4.12.0    True        False         False      31m
kube-apiserver                             4.12.0    True        False         False      26m
kube-controller-manager                    4.12.0    True        False         False      36m
kube-scheduler                             4.12.0    True        False         False      36m
kube-storage-version-migrator              4.12.0    True        False         False      37m
machine-api                                4.12.0    True        False         False      29m
machine-approver                           4.12.0    True        False         False      37m
machine-config                             4.12.0    True        False         False      36m
marketplace                                4.12.0    True        False         False      37m
monitoring                                 4.12.0    True        False         False      29m
network                                    4.12.0    True        False         False      38m
node-tuning                                4.12.0    True        False         False      37m
openshift-apiserver                        4.12.0    True        False         False      32m
openshift-controller-manager               4.12.0    True        False         False      30m
openshift-samples                          4.12.0    True        False         False      32m
operator-lifecycle-manager                 4.12.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False      32m
service-ca                                 4.12.0    True        False         False      38m
storage                                    4.12.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

2.16.1. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定に関する手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

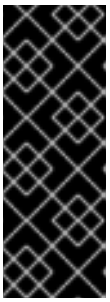
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

2.16.1.1. IBM Power の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Power にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
-
```

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME             VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.12          True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

2.16.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2.17. USER-PROVISIONED INFRASTRUCTURE でのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False      19m
baremetal                                  4.12.0    True        False         False      37m
cloud-credential                           4.12.0    True        False         False      40m
cluster-autoscaler                         4.12.0    True        False         False      37m
config-operator                            4.12.0    True        False         False      38m
console                                    4.12.0    True        False         False      26m
csi-snapshot-controller                    4.12.0    True        False         False      37m
dns                                        4.12.0    True        False         False      37m
etcd                                       4.12.0    True        False         False      36m
image-registry                             4.12.0    True        False         False      31m
ingress                                    4.12.0    True        False         False      30m
insights                                   4.12.0    True        False         False      31m
kube-apiserver                             4.12.0    True        False         False      26m
kube-controller-manager                    4.12.0    True        False         False      36m
kube-scheduler                             4.12.0    True        False         False      36m
kube-storage-version-migrator              4.12.0    True        False         False      37m
machine-api                                4.12.0    True        False         False      29m
machine-approver                           4.12.0    True        False         False      37m
machine-config                             4.12.0    True        False         False      36m
marketplace                                4.12.0    True        False         False      37m
monitoring                                 4.12.0    True        False         False      29m
network                                    4.12.0    True        False         False      38m
node-tuning                                4.12.0    True        False         False      37m
openshift-apiserver                        4.12.0    True        False         False      32m
openshift-controller-manager               4.12.0    True        False         False      30m
```

```
openshift-samples                          4.12.0    True        False        False      32m
operator-lifecycle-manager                 4.12.0    True        False        False      37m
operator-lifecycle-manager-catalog         4.12.0    True        False        False      37m
operator-lifecycle-manager-packageserver   4.12.0    True        False        False      32m
service-ca                                 4.12.0    True        False        False      38m
storage                                    4.12.0    True        False        False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

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

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                         READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1        9m
   openshift-apiserver             apiserver-67b9g                              1/1     Running    0
   3m
   ```

```
openshift-apiserver                apiserver-ljcmx                         1/1     Running     0
1m
openshift-apiserver                apiserver-z25h4                         1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. マルチパスを有効にするための追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

2.18. OPENSHIFT CONTAINER PLATFORM の TELEMETRY アクセス

OpenShift Container Platform 4.12 では、クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは OpenShift Cluster Manager Hybrid Cloud Console に登録されます。

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

関連情報

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

2.19. 次のステップ

- RHCOS のカーネル引数でのマルチパスの有効化
- クラスターのカスタマイズ
- 必要に応じて、リモートヘルスレポート を作成できます。

第3章 ネットワークが制限された環境での IBM POWER へのクラスターのインストール

OpenShift Container Platform バージョン 4.12 では、ネットワークが制限された環境で独自にプロビジョニングする IBM Power インフラストラクチャーに、クラスターをインストールできます。

重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

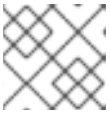
3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認した。
- クラスターインストール方法の選択およびそのユーザー向けの準備 を確認した。
- ネットワークが制限された環境でのインストール用にミラーレジストリーを作成 し、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得した。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

  重要

  インストールメディアにアクセスできるマシンでインストール手順が実行されるようにします。

- OpenShift Data Foundation またはクラスターでサポートされているその他のストレージプロトコルを使用して永続ストレージをプロビジョニングした。プライベートイメージレジストリーをデプロイするには、**ReadWriteMany** のアクセスモードで永続ストレージを設定する必要があります。
- クラスターがアクセスする必要がある サイトを許可するようにファイアウォールを設定 した (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

  注記

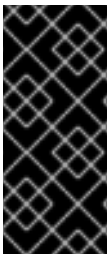
  プロキシーを設定する場合は、このサイトリストも確認してください。

3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.12 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

ネットワークが制限されたインストールを完了するには、OpenShift イメージレジストリーのコンテン

ツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

user-provisioned installation の設定は複雑であるため、user-provisioned infrastructure を使用してネットワークが制限されたインストールを試行する前に、標準的な user-provisioned infrastructure を実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

3.3. OPENSHIFT CONTAINER PLATFORM のインターネットアクセス

OpenShift Container Platform 4.12 では、クラスターのインストールに必要なイメージを取得するために、インターネットにアクセスする必要があります。

インターネットへのアクセスは以下を実行するために必要です。

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

重要

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

3.4. USER-PROVISIONED INFRASTRUCTURE を使用したクラスターの要件

user-provisioned infrastructure を含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

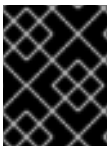
このセクションでは、user-provisioned infrastructure に OpenShift Container Platform をデプロイする要件を説明します。

3.4.1. クラスターのインストールに必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

表3.1 最低限必要なホスト

ホスト	説明
1 つの一時的なブートストラップマシン	クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。
3 つのコントロールプレーンマシン	コントロールプレーンマシンは、コントロールプレーンを設定する Kubernetes および OpenShift Container Platform サービスを実行します。
少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。	OpenShift Container Platform ユーザーが要求するワークロードは、コンピュートマシンで実行されます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンに別の物理ホストを使用します。

ブートストラップ、コントロールプレーンおよびコンピュートマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

3.4.2. クラスターインストールの最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

表3.2 最小リソース要件

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ	1 秒あたりの入出力 (IOPS) [2]
ブートストラップ	RHCOS	2	16 GB	100 GB	300
コントロールプレーン	RHCOS	2	16 GB	100 GB	300
Compute	RHCOS	2	8 GB	100 GB	300

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

2. OpenShift Container Platform と Kubernetes は、ディスクのパフォーマンスの影響を受けるため、特にコントロールプレーンノードの etcd には、より高速なストレージが推奨されます。多くのクラウドプラットフォームでは、ストレージサイズと IOPS スケールが一緒にあるため、十分なパフォーマンスを得るためにストレージボリュームの割り当てが必要になる場合があります。

プラットフォームのインスタンスタイプがクラスターマシンの最小要件を満たす場合、これは OpenShift Container Platform で使用することがサポートされます。

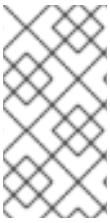
関連情報

- ストレージの最適化

3.4.3. IBM Power の最小要件

OpenShift Container Platform バージョン 4.12 は、以下の IBM ハードウェアにインストールできます。

- IBM Power9、または Power10 プロセッサーベースのシステム



注記

すべての IBM POWER8 モデル、IBM POWER9 AC922、IBM POWER9 IC922、および IBM POWER9 LC922 の RHCOS 機能のサポートは非推奨になりました。これらのハードウェアモデルは、OpenShift Container Platform 4.12 で引き続き完全にサポートされます。ただし、Red Hat は、新しいハードウェアモデルを使用することを推奨します。

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 つの IBM Power ベアメタルサーバーまたは 6 つの LPAR

オペレーティングシステム要件

- IBM Power9、または Power10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能

- 共有イーサネットアダプターを使用して仮想 I/O サーバーで利用可能
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 100GB / 16GB
- OpenShift Container Platform コンピュートマシン用に 100 GB / 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 100 GB / 16 GB

3.4.4. 推奨される IBM Power システム要件

ハードウェア要件

- 複数の PowerVM サーバーにまたがる 6 つの IBM Power ベアメタルサーバーまたは 6 つの LPAR

オペレーティングシステム要件

- IBM Power9、または Power10 プロセッサーベースのシステムの 1 つのインスタンス

IBM Power インスタンスで、以下を設定します。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

IBM Power ゲスト仮想マシンのディスクストレージ

- ローカルストレージ、または vSCSI、NPIV (N-Port ID Virtualization) または SSP(共有ストレージプール) を使用して仮想 I/O サーバーによってプロビジョニングされるストレージ

PowerVM ゲスト仮想マシンのネットワーク

- 専用の物理アダプター、または SR-IOV 仮想機能
- 共有イーサネットアダプターを使用して仮想 I/O サーバーで利用可能
- IBM vNIC を使用して仮想 I/O サーバーによって仮想化される

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 120 GB / 32 GB
- OpenShift Container Platform コンピュートマシン用に 120 GB / 32 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 120 GB / 16 GB

3.4.5. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認しま

す。**machine-approver** は、kubelet 認証情報を使用して要求されるサービング証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

3.4.6. user-provisioned infrastructure のネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** でネットワークを設定し、Ignition 設定ファイルをフェッチする必要があります。

初回の起動時に、マシンには DHCP サーバーを使用して設定される IP アドレス設定、または必要な起動オプションを指定して静的に設定される IP アドレス設定が必要です。ネットワーク設定の確立後に、マシンは HTTP または HTTPS サーバーから Ignition 設定ファイルをダウンロードします。その後、Ignition 設定ファイルは各マシンの正確な状態を設定するために使用されます。Machine Config Operator はインストール後に、新しい証明書やキーの適用など、マシンへの追加の変更を完了します。

クラスターマシンの長期管理に DHCP サーバーを使用することが推奨されます。DHCP サーバーが永続 IP アドレス、DNS サーバー情報、およびホスト名をクラスターマシンに提供するように設定されていることを確認します。

注記

DHCP サービスが user-provisioned infrastructure で利用できない場合は、IP ネットワーク設定および DNS サーバーのアドレスを RHCOS のインストール時にノードに提供することができます。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始** のセクションを参照してください。

Kubernetes API サーバーはクラスターマシンのノード名を解決できる必要があります。API サーバーおよびワーカーノードが異なるゾーンに置かれている場合、デフォルトの DNS 検索ゾーンを、API サーバーでノード名を解決できるように設定することができます。もう 1 つの実行可能な方法として、ノードオブジェクトとすべての DNS 要求の両方において、ホストを完全修飾ドメイン名で常に参照します。

3.4.6.1. DHCP を使用したクラスターノードのホスト名の設定

Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、ホスト名は NetworkManager 経由で設定されます。デフォルトでは、マシンは DHCP 経由でホスト名を取得します。ホスト名が DHCP によって提供されない場合、カーネル引数を介して静的に設定される場合、または別の方法でホスト名が取得される場合は、逆引き DNS ルックアップによって取得されます。逆引き DNS ルックアップは、ネットワークがノードで初期化された後に発生し、解決に時間がかかる場合があります。その他のシステムサービスは、これより前に起動し、ホスト名を **localhost** または同様のものとして検出できます。これを回避するには、DHCP を使用して各クラスターノードのホスト名を指定できます。

また、DHCP を介してホスト名を設定すると、DNS スプリットホライズンが実装されている環境での手動の DNS レコード名設定エラーを回避できます。

3.4.6.2. ネットワーク接続の要件

OpenShift Container Platform クラスターのコンポーネントが通信できるように、マシン間のネットワーク接続を設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

このセクションでは、必要なポートの詳細を説明します。

表3.3 すべてのマシンからすべてのマシンへの通信に使用されるポート

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
	500	IPsec IKE パケット
	4500	IPsec NAT-T パケット
	123	UDP ポート **123** のネットワークタイムプロトコル (NTP) 外部 NTP タイムサーバーが設定されている場合は、UDP ポート **123** を開く必要があります。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート
ESP	該当なし	IPsec Encapsulating Security Payload (ESP)

表3.4 すべてのマシンからコントロールプレーンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表3.5 コントロールプレーンマシンからコントロールプレーンマシンへの通信に使用されるポート

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

user-provisioned infrastructure の NTP 設定
OpenShift Container Platform クラスターは、デフォルトでパブリック Network Time Protocol (NTP) サーバーを使用するように設定されます。ローカルのエンタープライズ NTP サーバーを使用する必要があるか、クラスターが切断されたネットワークにデプロイされている場合は、特定のタイムサーバーを使用するようにクラスターを設定できます。詳細は、**chrony タイムサービスの設定** のドキュメントを参照してください。

DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

関連情報

- chrony タイムサービスの設定

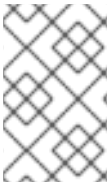
3.4.7. user-provisioned DNS 要件

OpenShift Container Platform のデプロイメントでは、以下のコンポーネントに DNS 名前解決が必要です。

- Kubernetes API
- OpenShift Container Platform のアプリケーションワイルドカード
- ブートストラップ、コントロールプレーンおよびコンピュートマシン

また、Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンに逆引き DNS 解決も必要です。

DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。ホスト名が DHCP によって提供されていない場合は、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するため、逆引きレコードは重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

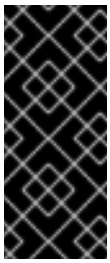


注記

各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**user-provisioned infrastructure に関する DHCP の推奨事項**のセクションを参照してください。

以下の DNS レコードは、user-provisioned OpenShift Container Platform クラスターに必要で、これはインストール前に設定されている必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表3.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	API ロードバランサーを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	API ロードバランサーを内部的に識別するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	アプリケーション Ingress ロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコード。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。 たとえば、**console-openshift-console.apps.<cluster_name>.<base_domain>** は、OpenShift Container Platform コンソールへのワイルドカードルートとして使用されます。
ブートストラップマシン	**bootstrap.<cluster_name>.<base_domain>.**	ブートストラップマシンを識別するための DNS A / AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コントロールプレーンマシン	**<control_plane><n>.<cluster_name>.<base_domain>.**	コントロールプレーンノードの各マシンを特定するための DNS A/AAAA または CNAME レコードおよび DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。
コンピュートマシン	**<compute><n>.<cluster_name>.<base_domain>.**	ワーカーノードの各マシンを特定するための DNS A/AAAA または CNAME レコード、および DNS PTR レコード。これらのレコードは、クラスター内のノードで解決できる必要があります。

注記

OpenShift Container Platform 4.4 以降では、DNS 設定で etcd ホストおよび SRV レコードを指定する必要はありません。

ヒント

dig コマンドを使用して、名前および逆引き名前解決を確認することができます。検証手順の詳細は、**user-provisioned infrastructure の DNS 解決の検証**のセクションを参照してください。

3.4.7.1. user-provisioned クラスターの DNS 設定の例

このセクションでは、user-provisioned infrastructure に OpenShift Container Platform をデプロイするための DNS 要件を満たす A および PTR レコード設定サンプルを提供します。サンプルは、特定の DNS ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、クラスター名は **ocp4** で、ベースドメインは **example.com** です。

user-provisioned クラスターの DNS A レコードの設定例

BIND ゾーンファイルの以下の例は、user-provisioned クラスターの名前解決の A レコードの例を示しています。

例3.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
bootstrap.ocp4.example.com. IN A 192.168.1.96 4
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 5
control-plane1.ocp4.example.com. IN A 192.168.1.98 6
control-plane2.ocp4.example.com. IN A 192.168.1.99 7
;
compute0.ocp4.example.com. IN A 192.168.1.11 8
```

```
compute1.ocp4.example.com. IN A 192.168.1.7 9
;
;EOF
```

1. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照します。
2. Kubernetes API の名前解決を提供します。レコードは API ロードバランサーの IP アドレスを参照し、内部クラスター通信に使用されます。
3. ワイルドカードルートの名前解決を提供します。レコードは、アプリケーション Ingress ロードバランサーの IP アドレスを参照します。アプリケーション Ingress ロードバランサーは、Ingress コントローラー Pod を実行するマシンをターゲットにします。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

4. ブートストラップマシンの名前解決を提供します。
5. 6. 7. コントロールプレーンマシンの名前解決を提供します。
8. 9. コンピュートマシンの名前解決を提供します。

user-provisioned クラスターの DNS PTR レコードの設定例

以下の BIND ゾーンファイルの例では、user-provisioned クラスターの逆引き名前解決の PTR レコードの例を示しています。

例3.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. 3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. 5
```

```
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;EOF
```

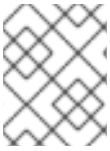
1 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照します。

2 Kubernetes API の逆引き DNS 解決を提供します。PTR レコードは、API ロードバランサーのレコード名を参照し、内部クラスター通信に使用されます。

3 ブートストラップマシンの逆引き DNS 解決を提供します。

4 5 6 コントロールプレーンマシンの逆引き DNS 解決を提供します。

7 8 コンピュートマシンの逆引き DNS 解決を提供します。

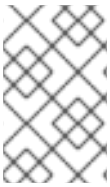


注記

PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。

3.4.8. ユーザーによってプロビジョニングされるインフラストラクチャーの負荷分散要件

OpenShift Container Platform をインストールする前に、API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングする必要があります。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

Red Hat Enterprise Linux (RHEL) インスタンスを使用して API およびアプリケーションイングレスロードバランサーをデプロイする場合は、RHEL サブスクリプションを別途購入する必要があります。

負荷分散インフラストラクチャーは以下の要件を満たす必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

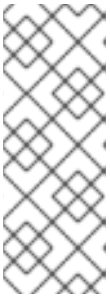
重要

API ロードバランサーのセッションの永続性は設定しないでください。Kubernetes API サーバーのセッション永続性を設定すると、OpenShift Container Platform クラスターとクラスター内で実行される Kubernetes API の過剰なアプリケーショントラフィックによりパフォーマンスの問題が発生する可能性があります。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表3.7 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとのプロービングで、2 回連続成功すると正常、3 回連続失敗すると異常と判断する設定は、十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。
以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP または SSL パススルーモードと呼ばれます。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ヒント

クライアントの実際の IP アドレスがアプリケーション Ingress ロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表3.8 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress コントローラー Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

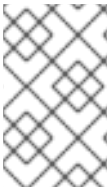
注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

3.4.8.1. ユーザーによってプロビジョニングされるクラスターのロードバランサーの設定例

このセクションでは、ユーザーによってプロビジョニングされるクラスターの負荷分散要件を満たす API およびアプリケーション Ingress ロードバランサーの設定例を説明します。この例は、HAProxy ロードバランサーの **/etc/haproxy/haproxy.cfg** 設定です。この例では、特定の負荷分散ソリューションを選択するためのアドバイスを提供することを目的としていません。

この例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。



注記

HAProxy をロードバランサーとして使用し、SELinux が **enforcing** に設定されている場合は、**setsebool -P haproxy_connect_any=1** を実行して、HAProxy サービスが設定済みの TCP ポートにバインドできることを確認する必要があります。

例3.3 API およびアプリケーション Ingress ロードバランサーの設定例

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
```

```
    maxconn     4000
    daemon
defaults
    mode                    http
    log                     global
    option                  dontlognull
    option http-server-close
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
listen api-server-6443 1
    bind *:6443
    mode tcp
    option  httpchk GET /readyz HTTP/1.0
    option  log-health-checks
    balance roundrobin
    server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup 2
    server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
    server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
    server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 3
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 4
    server master0 master0.ocp4.example.com:22623 check inter 1s
    server master1 master1.ocp4.example.com:22623 check inter 1s
    server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 5
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:443 check inter 1s
    server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 6
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp4.example.com:80 check inter 1s
    server worker1 worker1.ocp4.example.com:80 check inter 1s
```

1 ポート **6443** は Kubernetes API トラフィックを処理し、コントロールプレーンマシンを参照します。

2 4

ブートストラップエントリーは、OpenShift Container Platform クラスターのインストール前に有効にし、ブートストラッププロセスの完了後にそれらを削除する必要があります。

3 ポート **22623** はマシン設定サーバートラフィックを処理し、コントロールプレーンマシンを参照します。

5 ポート **443** は HTTPS トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

6 ポート **80** は HTTP トラフィックを処理し、Ingress コントローラー Pod を実行するマシンを参照します。Ingress コントローラー Pod はデフォルトでコンピュートマシンで実行されます。

注記

ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。

ヒント

HAProxy をロードバランサーとして使用する場合は、HAProxy ノードで **netstat -nltupe** を実行して、ポート **6443**、**22623**、**443**、および **80** で **haproxy** プロセスがリッスンしていることを確認することができます。

3.5. USER-PROVISIONED INFRASTRUCTURE の準備

user-provisioned infrastructure に OpenShift Container Platform をインストールする前に、基礎となるインフラストラクチャーを準備する必要があります。

このセクションでは、OpenShift Container Platform インストールの準備としてクラスターインフラストラクチャーを設定するために必要な手順の概要を説明します。これには、クラスターノード用の IP ネットワークおよびネットワーク接続を設定し、ファイアウォール経由で必要なポートを有効にし、必要な DNS および負荷分散インフラストラクチャーの設定が含まれます。

準備後、クラスターインフラストラクチャーは、**user-provisioned infrastructure を使用したクラスターの要件** セクションで説明されている要件を満たす必要があります。

前提条件

- OpenShift Container Platform 4.x Tested Integrations ページを確認した。
- **user-provisioned infrastructure を使用したクラスターの要件** セクションで説明されているインフラストラクチャーの要件を確認した。

手順

1. DHCP を使用して IP ネットワーク設定をクラスターノードに提供する場合は、DHCP サービスを設定します。

a. ノードの永続 IP アドレスを DHCP サーバー設定に追加します。設定で、関連するネットワークインターフェイスの MAC アドレスを、各ノードの目的の IP アドレスと一致させます。

b. DHCP を使用してクラスターマシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP サーバー設定を介してクラスターノードが使用する永続 DNS サーバーアドレスを定義します。

注記

DHCP サービスを使用しない場合、IP ネットワーク設定と DNS サーバーのアドレスを RHCOS インストール時にノードに指定する必要があります。ISO イメージからインストールしている場合は、ブート引数として渡すことができます。静的 IP プロビジョニングと高度なネットワークオプションの詳細は、**RHCOS のインストールと OpenShift Container Platform ブートストラッププロセスの開始** のセクションを参照してください。

c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項の詳細は、**DHCP を使用したクラスターノードのホスト名の設定** セクションを参照してください。

注記

DHCP サービスを使用しない場合、クラスターノードは逆引き DNS ルックアップを介してホスト名を取得します。

2. ネットワークインフラストラクチャーがクラスターコンポーネント間の必要なネットワーク接続を提供することを確認します。要件に関する詳細は、**user-provisioned infrastructure のネットワーク要件** のセクションを参照してください。

3. OpenShift Container Platform クラスターコンポーネントで通信するために必要なポートを有効にするようにファイアウォールを設定します。必要なポートの詳細は、**user-provisioned infrastructure のネットワーク要件** のセクションを参照してください。

重要

デフォルトで、ポート **1936** は OpenShift Container Platform クラスターにアクセスできます。これは、各コントロールプレーンノードがこのポートへのアクセスを必要とするためです。

Ingress ロードバランサーを使用してこのポートを公開しないでください。これを実行すると、Ingress コントローラーに関連する統計やメトリクスなどの機密情報が公開される可能性があるためです。

4. クラスターに必要な DNS インフラストラクチャーを設定します。

a. Kubernetes API、アプリケーションワイルドカード、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの DNS 名前解決を設定します。

b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
OpenShift Container Platform DNS 要件の詳細は、**user-provisioned DNS 要件** のセクションを参照してください。

5. DNS 設定を検証します。

   a. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答の IP アドレスが正しいコンポーネントに対応することを確認します。

   b. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答のレコード名が正しいコンポーネントに対応することを確認します。
   DNS 検証手順の詳細は、**user-provisioned infrastructure の DNS 解決の検証**のセクションを参照してください。

6. 必要な API およびアプリケーションの Ingress 負荷分散インフラストラクチャーをプロビジョニングします。要件に関する詳細は、**user-provisioned infrastructure の負荷分散要件**のセクションを参照してください。

注記

一部の負荷分散ソリューションでは、負荷分散を初期化する前に、クラスターノードの DNS 名前解決を有効化する必要があります。

3.6. USER-PROVISIONED INFRASTRUCTURE の DNS 解決の検証

OpenShift Container Platform を user-provisioned infrastructure にインストールする前に、DNS 設定を検証できます。

重要

このセクションの検証手順は、クラスターのインストール前に正常に実行される必要があります。

前提条件

- user-provisioned infrastructure に必要な DNS レコードを設定している。

手順

1. インストールノードから、Kubernetes API、ワイルドカードルート、およびクラスターノードのレコード名に対して DNS ルックアップを実行します。応答に含まれる IP アドレスが正しいコンポーネントに対応することを確認します。

   a. Kubernetes API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> 1
```

   1 **<nameserver_ip>** をネームサーバーの IP アドレスに、**<cluster_name>** をクラスター名に、**<base_domain>** をベースドメイン名に置き換えます。

   出力例

```
api.ocp4.example.com.  604800 IN A 192.168.1.5
```

b. Kubernetes 内部 API レコード名に対してルックアップを実行します。結果が API ロードバランサーの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
```

出力例

```
api-int.ocp4.example.com.  604800 IN A 192.168.1.5
```

c. ***.apps.<cluster_name>.<base_domain>** DNS ワイルドカードルックアップの例をテストします。すべてのアプリケーションのワイルドカードルックアップは、アプリケーション Ingress ロードバランサーの IP アドレスに解決する必要があります。

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

出力例

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```

注記

出力例では、同じロードバランサーが Kubernetes API およびアプリケーションの Ingress トラフィックに使用されます。実稼働のシナリオでは、API およびアプリケーション Ingress ロードバランサーを個別にデプロイし、それぞれのロードバランサーインフラストラクチャーを分離してスケーリングすることができます。

random は、別のワイルドカード値に置き換えることができます。たとえば、OpenShift Container Platform コンソールへのルートをクエリーできます。

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

出力例

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. ブートストラップ DNS レコード名に対してルックアップを実行します。結果がブートストラップノードの IP アドレスを参照することを確認します。

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

出力例

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. この方法を使用して、コントロールプレーンおよびコンピュートノードの DNS レコード名に対してルックアップを実行します。結果が各ノードの IP アドレスに対応していることを確認します。

2. インストールノードから、ロードバランサーとクラスターノードの IP アドレスに対して逆引き DNS ルックアップを実行します。応答に含まれるレコード名が正しいコンポーネントに対応することを確認します。

   a. API ロードバランサーの IP アドレスに対して逆引き参照を実行します。応答に、Kubernetes API および Kubernetes 内部 API のレコード名が含まれていることを確認します。

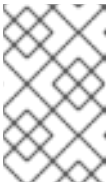
   ```
   $ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
   ```

   出力例

   ```
   5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com. 1
   5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com. 2
   ```

   1 Kubernetes 内部 API のレコード名を指定します。

   2 Kubernetes API のレコード名を指定します。

   注記

   PTR レコードは、OpenShift Container Platform アプリケーションのワイルドカードには必要ありません。アプリケーション Ingress ロードバランサーの IP アドレスに対する逆引き DNS 解決の検証手順は必要ありません。

   b. ブートストラップノードの IP アドレスに対して逆引き参照を実行します。結果がブートストラップノードの DNS レコード名を参照していることを確認します。

   ```
   $ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
   ```

   出力例

   ```
   96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
   ```

   c. この方法を使用して、コントロールプレーンおよびコンピュートノードの IP アドレスに対して逆引きルックアップを実行します。結果が各ノードの DNS レコード名に対応していることを確認します。

3.7. クラスターノード SSH アクセス用の鍵ペアの生成

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

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

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

重要

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

注記

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

手順

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

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

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

注記

FIPS で検証済みまたは進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64**、**ppc64le**、および **s390x** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. SSH 公開鍵を表示します。

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

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

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

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

注記

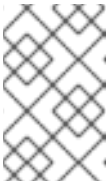
一部のディストリビューションでは、~/**.ssh/id_rsa** および ~/**.ssh/id_dsa** などのデフォルトの SSH 秘密鍵のアイデンティティーは自動的に管理されます。

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

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

出力例

```
Agent pid 31874
```



注記

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

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

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

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

出力例

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

次のステップ

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

3.8. インストール設定ファイルの手動作成

クラスターをインストールするには、インストール設定ファイルを手動で作成する必要があります。

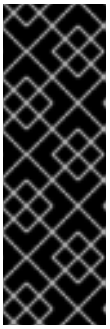
前提条件

- インストールプログラムで使用するための SSH 公開鍵がローカルマシン上に存在する。この鍵は、デバッグや障害復旧のために、クラスターノードへの SSH 認証に使用できます。
- OpenShift Container Platform インストールプログラムとクラスターのプルシークレットを取得している。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

このディレクトリーは必ず作成してください。ブートストラップ X.509 証明書などの一部のインストールアセットは、有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してください。

2. 提供されているサンプルの **install-config.yaml** ファイルテンプレートをカスタマイズし、ファイルを **<installation_directory>** に保存します。

注記

この設定ファイルの名前を **install-config.yaml** と付ける必要があります。

3. 多くのクラスターのインストールに使用できるように、**install-config.yaml** ファイルをバックアップします。

重要

インストールプロセスの次のステップで **install-config.yaml** ファイルを使用するため、今すぐこのファイルをバックアップしてください。

3.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、環境の詳細を記述するカスタマイズされた **install-config.yaml** インストール設定ファイルを指定します。

注記

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

3.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表3.9 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストールプログラムは、古い API バージョンもサポートしている場合があります。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**<metadata.name>.<baseDomain>** 形式を使用する **baseDomain** と **metadata.name** パラメーターの値を組み合わせたものです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールを実行する特定のプラットフォームの設定: **alibabacloud**、**aws**、**baremetal**、**azure**、**gcp**、**ibmcloud**、**nutanix**、**openstack**、**ovirt**、**vsphere**、または **{}**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームを参照してください。	オブジェクト

パラメーター	説明	値

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

3.8.1.2. ネットワーク設定パラメーター

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

- Red Hat OpenShift Networking OVN-Kubernetes ネットワークプラグインを使用する場合、IPv4 と IPv6 の両方のアドレスファミリーがサポートされます。
- Red Hat OpenShift Networking OpenShift SDN ネットワークプラグインを使用する場合、IPv4 アドレスファミリーのみがサポートされます。

両方の IP アドレスファミリーを使用するようにクラスターを設定する場合は、次の要件を確認してください。

- どちらの IP ファミリーも、デフォルトゲートウェイに同じネットワークインターフェイスを使用する必要があります。
- 両方の IP ファミリーにデフォルトゲートウェイが必要です。
- すべてのネットワーク設定パラメーターに対して、IPv4 アドレスと IPv6 アドレスを同じ順序で指定する必要があります。たとえば、以下の設定では、IPv4 アドレスは IPv6 アドレスの前に記載されます。

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```

注記

Globalnet は、Red Hat OpenShift Data Foundation ディザスターリカバリーソリューションではサポートされていません。局地的なディザスターリカバリーのシナリオでは、各クラスター内のクラスターとサービスネットワークに重複しない範囲のプライベート IP アドレスを使用するようにしてください。

表3.10 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールする Red Hat OpenShift Networking ネットワークプラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。**OpenShiftSDN** は、全 Linux ネットワーク用の CNI プラグインです。**OVNKubernetes** は、Linux ネットワークと、Linux サーバーと Windows サーバーの両方を含む Linux ネットワークおよびハイブリッドネットワーク用の CNI プラグインです。デフォルトの値は **OVNkubernetes** です。

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

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。

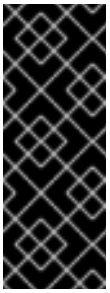
パラメーター	説明	値
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。 複数の IP カーネル引数を指定する場合、**machineNetwork.cidr** の値はプライマリーネットワークの CIDR である必要があります。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

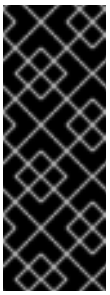
3.8.1.3. オプションの設定パラメーター

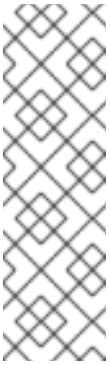
オプションのインストール設定パラメーターは、以下の表で説明されています。

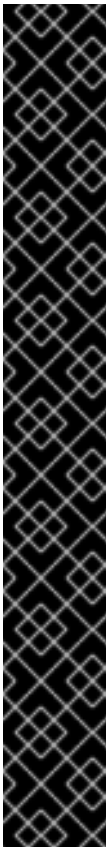
表3.11 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
capabilities	オプションのコアクラスターコンポーネントのインストールを制御します。オプションのコンポーネントを無効にすることで、OpenShift Container Platform クラスターのフットプリントを削減できます。詳細は、**インストール** の「クラスター機能ページ」を参照してください。	文字列配列
capabilities.baselineCapabilitySet	有効にするオプション機能の初期セットを選択します。有効な値は **None**、**v4.11**、**v4.12**、**vCurrent** です。デフォルト値は **vCurrent** です。	文字列
capabilities.additionalEnabledCapabilities	オプションの機能のセットを、**baselineCapabilitySet** で指定したものを超えて拡張します。このパラメーターで複数の機能を指定できます。	文字列配列
compute	コンピュートノードを設定するマシンの設定。	**MachinePool** オブジェクトの配列。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時マルチスレッドはマシンのコアのパフォーマンスを上げるために有効化されます。  **重要** 同時マルチスレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**alibabacloud**、**aws**、**azure**, **gcp**、**ibmcloud**、**nutanix**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
featureSet	機能セットのクラスターを有効にします。機能セットは、デフォルトで有効にされない OpenShift Container Platform 機能のコレクションです。インストール中に機能セットを有効にする方法の詳細は、「機能ゲートの使用による各種機能の有効化」を参照してください。	文字列。**TechPreviewNoUpgrade** など、有効にする機能セットの名前。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **ppc64le** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時マルチスレッドはマシンのコアのパフォーマンスを上げるために有効化されます。  **重要** 同時マルチスレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**alibabacloud**、**aws**、**azure**, **gcp**、**ibmcloud**、**nutanix**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
credentialsMode	Cloud Credential Operator (CCO) モード。モードを指定しないと、CCO は指定された認証情報の機能を動的に判別しようとします。この場合、複数のモードがサポートされるプラットフォームで Mint モードが優先されます。 **注記** すべてのクラウドプロバイダーですべての CCO モードがサポートされているわけではありません。CCO モードの詳細は、**Cluster Operators リファレンス** の **Cloud Credential Operator** を参照してください。  **注記** AWS アカウントでサービスコントロールポリシー (SCP) が有効になっている場合は、**credentialsMode** パラメーターを **Mint**、**Passthrough** または **Manual** に設定する必要があります。	**Mint**、**Passthrough**、**Manual**、または空の文字列 (**""**)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **重要** クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。FIPS 検証済み/Modules In Process 暗号ライブラリーの使用は、**x86_64**、**ppc64le**、および **s390x** アーキテクチャー上の OpenShift Container Platform デプロイメントでのみサポートされます。 **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。

パラメーター	説明	値
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。デフォルト値は **External** です。 このパラメーターを **Internal** に設定することは、クラウド以外のプラットフォームではサポートされません。 **重要** フィールドの値が **Internal** に設定されている場合、クラスターは機能しなくなります。詳細は、BZ#1953035 を参照してください。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

3.8.2. IBM Power のサンプル install-config.yaml ファイル

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

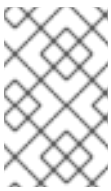
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
```

```
  replicas: 0 4
  architecture : ppc64le
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture: ppc64le
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OVNKubernetes 11
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
additionalTrustBundle: | 17
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 18
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、**compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) またはハイパースレッディングを有効/無効にするかどうかを指定します。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



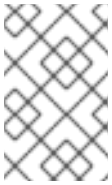
注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** ファイルであるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 OpenShift Container Platform を user-provisioned infrastructure にインストールする場合は、この値を **0** に設定する必要があります。installer-provisioned installation では、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。user-provisioned installation では、クラスターのインストールの終了前にコンピュートマシンを手動でデプロイする必要があります。



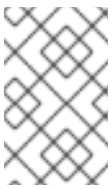
注記

3 ノードクラスターをインストールする場合は、Red Hat Enterprise Linux CoreOS (RHCOS) マシンをインストールする際にコンピュートマシンをデプロイしないでください。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこれらの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。



注記

クラス E の CIDR 範囲は、将来の使用のために予約されています。クラス E CIDR 範囲を使用するには、ネットワーク環境がクラス E CIDR 範囲内の IP アドレスを受け入れるようにする必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定されている場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。これにより、510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 インストールするクラスターネットワークプラグイン。サポートされている値は **OVNKubernetes** と **OpenShiftSDN** です。デフォルトの値は **OVNkubernetes** です。

12 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。このブロックは既存の物理ネットワークと重複できません。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

13 プラットフォームを **none** に設定する必要があります。IBM Power インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

重要

プラットフォームタイプ **none** でインストールされたクラスターは、Machine API を使用したコンピューティングマシンの管理など、一部の機能を使用できません。この制限は、クラスターに接続されている計算マシンが、通常はこの機能をサポートするプラットフォームにインストールされている場合でも適用されます。このパラメーターは、インストール後に変更することはできません。

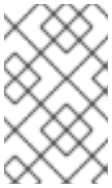
14 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。FIPS 検証済み/Modules In Process 暗号ライブラリーの使用は、**x86_64**、**ppc64le**、および **s390x** アーキテクチャー上の OpenShift Container Platform デプロイメントでのみサポートされます。

15 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** に、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーの SSH 公開鍵。



注記

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

17 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

18 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

3.8.3. インストール時のクラスター全体のプロキシーの設定

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

前提条件

- 既存の **install-config.yaml** ファイルがある。
- クラスターがアクセスする必要のあるサイトを確認済みで、それらのいずれかがプロキシーを

バイパスする必要があるかどうかを判別している。デフォルトで、すべてのクラスター Egress トラフィック (クラスターをホストするクラウドに関するクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。プロキシーを必要に応じてバイパスするために、サイトを **Proxy** オブジェクトの **spec.noProxy** フィールドに追加している。

注記

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

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

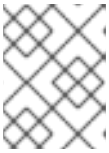
手順

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

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

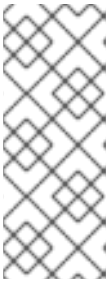
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。
3. プロキシーから除外するための宛先ドメイン名、IP アドレス、または他のネットワーク CIDR のコンマ区切りのリスト。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、**y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。
5. オプション: **trustedCA** フィールドの **user-ca-bundle** 設定マップを参照する **Proxy** オブジェクトの設定を決定するポリシー。許可される値は **Proxyonly** および **Always** です。**Proxyonly** を使用して、**http/https** プロキシーが設定されている場合にのみ **user-ca-**

bundle 設定マップを参照します。**Always** を使用して、常に **user-ca-bundle** 設定マップを参照します。デフォルト値は **Proxyonly** です。



注記

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



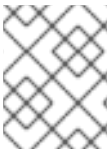
注記

インストーラーがタイムアウトした場合は、インストーラーの **wait-for** コマンドを使用してデプロイメントを再起動してからデプロイメントを完了します。以下に例を示します。

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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

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



注記

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

3.8.4. 3 ノードクラスターの設定

オプションで、3 台のコントロールプレーンマシンのみで構成されるベアメタルクラスターに、ゼロコンピュートマシンをデプロイできます。これにより、テスト、開発、および実稼働に使用するための小規模なリソース効率の高いクラスターが、クラスター管理者および開発者に提供されます。

3 ノードの OpenShift Container Platform 環境では、3 つのコントロールプレーンマシンがスケジュール対象となります。つまり、アプリケーションのワークロードがそれらで実行されるようにスケジュールされます。

前提条件

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

手順

- 以下の **compute** スタンザに示されるように、コンピュートレプリカの数が **install-config.yaml** ファイルで **0** に設定されることを確認します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

注記

デプロイするコンピュートマシンの数にかかわらず、OpenShift Container Platform を user-provisioned infrastructure にインストールする際に、コンピュートマシンの **replicas** パラメーターの値を **0** に設定する必要があります。installer-provisioned installation では、パラメーターはクラスターが作成し、管理するコンピュートマシンの数を制御します。これは、コンピュートマシンが手動でデプロイされる、user-provisioned installation には適用されません。

3 ノードのクラスターのインストールで、以下の手順を実行します。

- ゼロ (0) コンピュートノードで 3 ノードクラスターをデプロイする場合、Ingress コントローラー Pod はコントロールプレーンノードで実行されます。3 ノードクラスターデプロイメントでは、HTTP および HTTPS トラフィックをコントロールプレーンノードにルーティングするようにアプリケーション Ingress ロードバランサーを設定する必要があります。詳細は、**user-provisioned infrastructure の負荷分散要件** のセクションを参照してください。

- 以下の手順で Kubernetes マニフェストファイルを作成する際に、**<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルの **mastersSchedulable** パラメーターが **true** に設定されていることを確認します。これにより、アプリケーションのワークロードがコントロールプレーンノードで実行できます。

- Red Hat Enterprise Linux CoreOS (RHCOS) マシンを作成する際にはコンピュートノードをデプロイしないでください。

3.9. CLUSTER NETWORK OPERATO の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前のカスタムリソース (CR) オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のフィールドを指定します。

CNO 設定は、**Network.config.openshift.io** API グループの **Network** API からクラスターのインストール時に以下のフィールドを継承し、これらのフィールドは変更できません。

clusterNetwork

Pod IP アドレスの割り当てに使用する IP アドレスプール。

serviceNetwork

サービスの IP アドレスプール。

defaultNetwork.type

OpenShift SDN や OVN-Kubernetes などのクラスターネットワークプラグイン。

defaultNetwork オブジェクトのフィールドを **cluster** という名前の CNO オブジェクトに設定することにより、クラスターのクラスターネットワークプラグイン設定を指定できます。

3.9.1. Cluster Network Operator 設定オブジェクト

Cluster Network Operator (CNO) のフィールドは以下の表で説明されています。

表3.12 Cluster Network Operator 設定オブジェクト

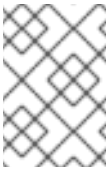
フィールド	型	説明
metadata.name	**string**	CNO オブジェクトの名前。この名前は常に **cluster** です。
spec.clusterNetwork	**array**	Pod IP アドレスの割り当て、サブネット接頭辞の長さのクラスター内の個別ノードへの割り当てに使用される IP アドレスのブロックを指定するリストです。以下に例を示します。 spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.serviceNetwork	**array**	サービスの IP アドレスのブロック。OpenShift SDN および OVN-Kubernetes ネットワークプラグインは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。以下に例を示します。 spec: serviceNetwork: - 172.30.0.0/14 マニフェストを作成する前に、このフィールドを **install-config.yaml** ファイルでのみカスタマイズすることができます。この値は、マニフェストファイルでは読み取り専用です。
spec.defaultNetwork	**object**	クラスターネットワークのネットワークプラグインを設定します。
spec.kubeProxyConfig	**object**	このオブジェクトのフィールドは、kube-proxy 設定を指定します。OVN-Kubernetes クラスターネットワークプラグインを使用している場合、kube-proxy 設定は機能しません。

重要

複数のネットワークにオブジェクトをデプロイする必要があるクラスターの場合は、**install-config.yaml** ファイルで定義されている各ネットワークタイプの **clusterNetwork.hostPrefix** パラメーターに、必ず同じ値を指定してください。**clusterNetwork.hostPrefix** パラメーターにそれぞれ異なる値を設定すると、OVN-Kubernetes ネットワークプラグインに影響が及び、異なるノード間のオブジェクトトラフィックをプラグインが効果的にルーティングできなくなる可能性があります。

defaultNetwork オブジェクト設定
defaultNetwork オブジェクトの値は、以下の表で定義されます。

表3.13 defaultNetwork オブジェクト

フィールド	型	説明
type	**string**	**OpenShiftSDN** または **OVNKubernetes** のいずれか。Red Hat OpenShift Networking ネットワークプラグインは、インストール中に選択されます。この値は、クラスターのインストール後は変更できません。  **注記** OpenShift Container Platform は、デフォルトで OVN-Kubernetes ネットワークプラグインを使用します。
openshiftSDNConfig	**object**	このオブジェクトは、OpenShift SDN ネットワークプラグインに対してのみ有効です。
ovnKubernetesConfig	**object**	このオブジェクトは、OVN-Kubernetes ネットワークプラグインに対してのみ有効です。

OpenShift SDN ネットワークプラグインの設定
以下の表では、OpenShift SDN ネットワークプラグインの設定フィールドを説明します。

表3.14 openshiftSDNConfig オブジェクト

フィールド	型	説明
mode	**string**	OpenShift SDN のネットワーク分離モードを設定します。デフォルト値は **NetworkPolicy** です。 **Multitenant** および **Subnet** の値は、OpenShift Container Platform 3.x との後方互換性を維持するために利用できますが、その使用は推奨されていません。この値は、クラスターのインストール後は変更できません。

フィールド	型	説明
mtu	**integer**	VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU をオーバーライドする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。 クラスターインストール時またはインストール後のタスクとして値を設定できます。詳細は、OpenShift Container Platform Networking ドキュメントの "Changing the MTU for the cluster network" を参照してください。
vxlanPort	**integer**	すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。この値は、クラスターのインストール後は変更できません。 別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。 Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

OVN-Kubernetes ネットワークプラグインの設定
次の表では、OVN-Kubernetes ネットワークプラグインの設定フィールドを説明します。

表3.15 ovnKubernetesConfig オブジェクト

フィールド	型	説明

フィールド	型	説明
mtu	**integer**	Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU をオーバーライドする必要はありません。 自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。 クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
genevePort	**integer**	すべての Geneve パケットに使用するポート。デフォルト値は **6081** です。この値は、クラスターのインストール後は変更できません。
ipsecConfig	**object**	IPsec 暗号化を有効にするために空のオブジェクトを指定します。
policyAuditConfig	**object**	ネットワークポリシー監査ロギングをカスタマイズする設定オブジェクトを指定します。指定されていない場合は、デフォルトの監査ログ設定が使用されます。
gatewayConfig	**object**	オプション: Egress トラフィックのノードゲートウェイへの送信方法をカスタマイズするための設定オブジェクトを指定します。 **注記** Egress トラフィックの移行中は、Cluster Network Operator (CNO) が変更を正常にロールアウトするまで、ワークロードとサービストラフィックに多少の中断が発生することが予想されます。

フィールド	型	説明
v4InternalSubnet	既存のネットワークインフラストラクチャーが **100.64.0.0/16** IPv4 サブネットと重複している場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。たとえば、**clusterNetwork.cidr** 値が **10.128.0.0/14** で、**clusterNetwork.hostPrefix** 値が **/23** の場合、ノードの最大数は **2^(23-14)=512** です。 このフィールドは、インストール後に変更できません。	デフォルト値は **100.64.0.0/16** です。

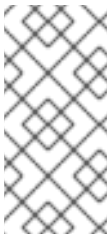
フィールド	型	説明
v6InternalSubnet	既存のネットワークインフラストラクチャーが **fd98::/48** IPv6 サブネットと重複する場合は、OVN-Kubernetes による内部使用のために別の IP アドレス範囲を指定できます。IP アドレス範囲が、OpenShift Container Platform インストールで使用される他のサブネットと重複しないようにする必要があります。IP アドレス範囲は、クラスターに追加できるノードの最大数より大きくする必要があります。 このフィールドは、インストール後に変更できません。	デフォルト値は **fd98::/48** です。

表3.16 policyAuditConfig オブジェクト

フィールド	型	説明
rateLimit	integer	ノードごとに毎秒生成されるメッセージの最大数。デフォルト値は、1秒あたり **20** メッセージです。
maxFileSize	integer	監査ログの最大サイズ (バイト単位)。デフォルト値は **50000000** (50MB) です。

フィールド	型	説明
destination	string	以下の追加の監査ログターゲットのいずれかになります。 **libc** ホスト上の journald プロセスの libc **syslog()** 関数。 **udp:<host>:<port>** syslog サーバー。**<host>:<port>** を syslog サーバーのホストおよびポートに置き換えます。 **unix:<file>** **<file>** で指定された Unix ドメインソケットファイル。 **null** 監査ログを追加のターゲットに送信しないでください。
syslogFacility	string	RFC5424 で定義される **kern** などの syslog ファシリティー。デフォルト値は **local0** です。

表3.17 gatewayConfig オブジェクト

フィールド	型	説明
routingViaHost	**boolean**	Pod からホストネットワークスタックへの Egress トラフィックを送信するには、このフィールドを **true** に設定します。  **注記** OpenShift Container Platform 4.12 では、egress IP はプライマリーインターフェイスのみに割り当てられます。そのため、**routingViaHost** を **true** に設定すると、OpenShift Container Platform 4.12 で egress IP は機能しません。 インストールおよびアプリケーションがカーネルルーティングテーブルに手動設定されたルートに依存するなど非常に特化されている場合には、Egress トラフィックをホストネットワークスタックにルーティングすることを推奨します。デフォルトでは、Egress トラフィックは OVN で処理され、クラスターを終了するために処理され、トラフィックはカーネルルーティングテーブルの特殊なルートによる影響を受けません。デフォルト値は **false** です。 このフィールドで、Open vSwitch ハードウェアオフロード機能との対話が可能になりました。このフィールドを **true** に設定すると、egress トラフィックがホストネットワークスタックで処理されるため、パフォーマンス的に、オフロードによる利点は得られません。

IPsec が有効な OVN-Kubernetes 設定の例

```
defaultNetwork:
```

```
    type: OVNKubernetes
    ovnKubernetesConfig:
      mtu: 1400
      genevePort: 6081
      ipsecConfig: {}
```

kubeProxyConfig オブジェクト設定
kubeProxyConfig オブジェクトの値は以下の表で定義されます。

表3.18 kubeProxyConfig オブジェクト

フィールド	型	説明
iptablesSyncPeriod	**string**	**iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ ドキュメントで説明されています。 **注記** OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

proxyArguments.iptables-min-sync-period	**array**	**iptables** ルールを更新する前の最小期間。このフィールドにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go **time** パッケージ で説明されています。デフォルト値:
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

3.10. KUBERNETES マニフェストおよび IGNITION 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを設定するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

インストール設定ファイルは Kubernetes マニフェストに変換されます。マニフェストは Ignition 設定ファイルにラップされます。これはクラスターマシンを設定するために後で使用されます。

重要

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

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

注記

マニフェストおよび Ignition ファイルを生成するインストールプログラムはアーキテクチャー固有であり、クライアントイメージミラー から取得できます。Linux バージョンのインストールプログラム (アーキテクチャーポストフィックスなし) は、ppc64le でのみ実行されます。このインストーラープログラムは、Mac OS バージョンとしても利用できます。

前提条件

- OpenShift Container Platform インストールプログラムを取得していること。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成していること。

手順

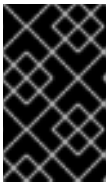
1. OpenShift Container Platform のインストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

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

1 **<installation_directory>** には、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

警告

3 ノードクラスターをインストールしている場合は、以下の手順を省略してコントロールプレーンノードをスケジュール対象にします。



重要

コントロールプレーンノードをデフォルトのスケジュール不可からスケジュール可に設定するには、追加のサブスクリプションが必要です。これは、コントロールプレーンノードがコンピュートノードになるためです。

2. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルの **mastersSchedulable** パラメーターが **false** に設定されていることを確認します。この設定により、Pod がコントロールプレーンマシンにスケジュールされなくなります。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、これが **false** に設定されていることを確認します。

   c. ファイルを保存し、終了します。

3. Ignition 設定ファイルを作成するには、インストールプログラムが含まれるディレクトリーから以下のコマンドを実行します。

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

1 **<installation_directory>** には、同じインストールディレクトリーを指定します。

Ignition 設定ファイルは、インストールディレクトリー内のブートストラップ、コントロールプレーン、およびコンピュートノード用に作成されます。**kubeadmin-password** および **kubeconfig** ファイルが **./<installation_directory>/auth** ディレクトリーに作成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

3.11. RHCOS のインストールおよび OPENSHIFT CONTAINER PLATFORM ブートストラッププロセスの開始

OpenShift Container Platform を独自にプロビジョニングする IBM Power インフラストラクチャーにインストールするには、Red Hat Enterprise Linux CoreOS (RHCOS) をマシンにインストールする必要があります。RHCOS のインストール時に、インストールするマシンのタイプについて OpenShift Container Platform インストールプログラムによって生成された Ignition 設定ファイルを指定する必要があります。適切なネットワーク、DNS、および負荷分散インフラストラクチャーが設定されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS マシンの再起動後に自動的に開始されます。

ISO イメージまたはネットワーク PXE ブートを使用する手順を実行して RHCOS をマシンにインストールできます。

3.11.1. ISO イメージを使用した RHCOS のインストール

ISO イメージを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定** のセクションを確認している。

手順

1. それぞれの Ignition 設定ファイルの SHA512 ダイジェストを取得します。たとえば、Linux を実行しているシステムで以下を使用して、**bootstrap.ign** Ignition 設定ファイルの SHA512 ダイジェストを取得できます。

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   ダイジェストは、クラスターノードの Ignition 設定ファイルの信頼性を検証するために、後の手順で **coreos-installer** に提供されます。

2. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

   重要

   HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

3. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

   出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

   コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

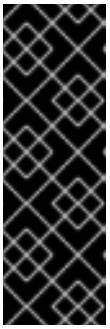
4. RHCOS イメージのミラー ページから、オペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS イメージを取得することは可能ですが、RHCOS イメージの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンド

の出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

出力例

```
"location": "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、このインストールではサポートされません。

ISO ファイルの名前は以下の例のようになります。

rhcos-<version>-live.<architecture>.iso

5. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - Lights Out Management (LOM) インターフェイスを使用して ISO リダイレクトを使用します。
6. オプションを指定したり、ライブ起動シーケンスを中断したりせずに、RHCOS ISO イメージを起動します。インストーラーが RHCOS ライブ環境でシェルプロンプトを起動するのを待ちます。

注記

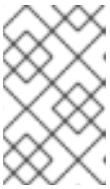
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

7. **coreos-installer** コマンドを実行し、インストール要件を満たすオプションを指定します。少なくとも、ノードタイプの Ignition 設定ファイルを参照する URL と、インストール先のデバイスを指定する必要があります。

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
--ignition-hash=sha512-<digest> 1 2
```

1 1 **core** ユーザーにはインストールを実行するために必要な root 特権がないため、 **sudo** を使用して **coreos-installer** コマンドを実行する必要があります。

2 **--ignition-hash** オプションは、Ignition 設定ファイルを HTTP URL を使用して取得し、クラスターノードの Ignition 設定ファイルの信頼性を検証するために必要です。**<digest>** は、先の手順で取得した Ignition 設定ファイル SHA512 ダイジェストです。



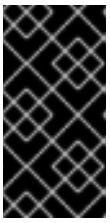
注記

TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

以下の例では、**/dev/sda** デバイスへのブートストラップノードのインストールを初期化します。ブートストラップノードの Ignition 設定ファイルは、IP アドレス 192.168.1.2 で HTTP Web サーバーから取得されます。

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

9. RHCOS のインストール後、システムを再起動する必要があります。システムの再起動後、指定した Ignition 設定ファイルを適用します。

10. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

11. 継続してクラスターの他のマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、OpenShift Container Platform のインストール前に少なくとも 2 つのコンピュートマシンも作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

3.11.1.1. 詳細の RHCOS インストールリファレンス

このセクションでは、Red Hat Enterprise Linux CoreOS (RHCOS) の手動インストールプロセスを変更できるようにするネットワーク設定および他の高度なオプションを説明します。以下の表では、RHCOS ライブインストーラーおよび **coreos-installer** コマンドで使用できるカーネル引数およびコマンドラインのオプションを説明します。

3.11.1.1.1. ISO インストールのネットワークおよびボンディングのオプション

ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。

重要

ネットワーク引数を手動で追加する場合は、**rd.neednet=1** カーネル引数を追加して、ネットワークを initramfs で有効にする必要があります。

以下の情報は、ISO インストール用に RHCOS ノードでネットワークおよびボンディングを設定する例を示しています。この例では、**ip=**、**nameserver=**、および **bond=** カーネル引数の使用方法を説明しています。

注記

順序は、カーネル引数の **ip=**、**nameserver=**、および **bond=** を追加する場合に重要です。

ネットワークオプションは、システムの起動時に **dracut** ツールに渡されます。**dracut** でサポートされるネットワークオプションの詳細は、**dracut.cmdline** man ページ を参照してください。

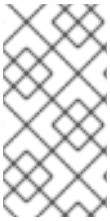
次の例は、ISO インストールのネットワークオプションです。

DHCP または静的 **IP** アドレスの設定

IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。静的 IP を設定する場合、各ノードで DNS サーバー IP アドレス (**nameserver=<dns_ip>**) を特定する必要があります。次の例では、以下を設定します。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- ホスト名: **core0.example.com**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```



注記

DHCP を使用して RHCOS マシンの IP アドレスを設定する場合、マシンは DHCP を介して DNS サーバー情報も取得します。DHCP ベースのデプロイメントの場合、DHCP サーバー設定を使用して RHCOS ノードが使用する DNS サーバーアドレスを定義できます。

静的ホスト名を使用しない **IP** アドレスの設定
静的ホスト名を割り当てずに IP アドレスを設定できます。静的ホスト名がユーザーによって設定されていない場合は、逆引き DNS ルックアップによって取得され、自動的に設定されます。静的ホスト名なしで IP アドレスを設定するには、次の例を参照してください。

- ノードの IP アドレス: **10.10.10.2**
- ゲートウェイアドレス: **10.10.10.254**
- ネットワーク: **255.255.255.0**
- DNS サーバーアドレス: **4.4.4.41**
- auto-configuration の値を **none** に設定します。IP ネットワークが静的に設定されている場合には、自動設定は必要ありません。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

複数のネットワークインターフェイスの指定
複数の **ip=** エントリーを設定することで、複数のネットワークインターフェイスを指定できます。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

デフォルトゲートウェイとルートの設定
オプション: **rd.route=** value を設定して、追加のネットワークへのルートを設定できます。

注記

1 つまたは複数のネットワークを設定する場合、1 つのデフォルトゲートウェイが必要です。追加のネットワークゲートウェイがプライマリーネットワークゲートウェイと異なる場合、デフォルトゲートウェイはプライマリーネットワークゲートウェイである必要があります。

- 次のコマンドを実行して、デフォルトゲートウェイを設定します。

```
ip=::10.10.10.254::::
```

- 次のコマンドを入力して、追加ネットワークのルートを設定します。

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

単一インターフェイスでの **DHCP** の無効化
2 つ以上のネットワークインターフェイスがあり、1 つのインターフェイスのみが使用される場合などに、1 つのインターフェイスで DHCP を無効にします。この例では、**enp1s0** インターフェイスには静的ネットワーク設定があり、使用されていない **enp2s0** では DHCP が無効になっています。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

DHCP と静的 **IP** 設定の組み合わせ
以下のように、複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

個々のインターフェイスでの **VLAN** の設定
オプション: **vlan=** パラメーターを使用して、個別のインターフェイスに VLAN を設定できます。

- ネットワークインターフェイスで VLAN を設定し、静的 IP アドレスを使用するには、次のコマンドを実行します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- ネットワークインターフェイスで VLAN を設定し、DHCP を使用するには、次のコマンドを実行します。

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

複数の **DNS** サーバーの指定
以下のように、各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング

オプション: **bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。次の例を参照してください。

- ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
  name は、ボンディングデバイス名 (**bond0**) で、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切りリストを表します。**options** はボンディングオプションのコンマ区切りのリストです。**modinfo bonding** を入力して、利用可能なオプションを表示します。

- **bond=** を使用してボンディングされたインターフェイスを作成する場合は、ボンディングされたインターフェイスの IP アドレスの割り当て方法やその他の情報を指定する必要があります。

- DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

複数のネットワークインターフェイスの単一インターフェイスへのボンディング
任意: 以下のように、**vlan=** パラメーターを指定して、DHCP を使用して、ボンディングされたインターフェイスで VLAN を設定できます。

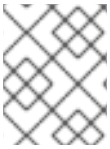
```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

次の例を使用して、VLAN でボンディングされたインターフェイスを設定し、静的 IP アドレスを使用します。

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

ネットワークチーミングの使用
任意: **team=** パラメーターを指定して、ボンディングの代わりにネットワークチーミングを使用できます。

- チームインターフェイス設定の構文は **team=name[:network_interfaces]** です。
  name はチームデバイス名 (**team0**)、**network_interfaces** は物理 (イーサネット) インターフェイス (**em1, em2**) のコンマ区切りリストを表します。



注記

RHCOS が次のバージョンの RHEL に切り替わると、チーミングは非推奨になる予定です。詳細は、こちらの Red Hat ナレッジベース記事 を参照してください。

次の例を使用して、ネットワークチームを設定します。

```
team=team0:em1,em2
ip=team0:dhcp
```

3.11.2. PXE ブートを使用した RHCOS のインストール

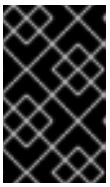
PXE ブートを使用してマシンに RHCOS をインストールできます。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- 適切な PXE インフラストラクチャーを設定していること。
- お使いのコンピューターからアクセスでき、作成するマシンからもアクセスできる HTTP サーバーがある。
- ネットワークやディスクパーティションなどのさまざまな機能の設定方法について、**高度な RHCOS インストール設定** のセクションを確認している。

手順

1. インストールプログラムが作成したブートストラップ、コントロールプレーン、およびコンピュートノード Ignition 設定ファイルを HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

HTTP サーバーに保存する前に、Ignition 設定で設定内容を追加したり、変更したりできます。インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. インストールホストから、Ignition 設定ファイルが URL で利用可能であることを確認します。以下の例では、ブートストラップノードの Ignition 設定ファイルを取得します。

```
$ curl -k http://<HTTP_server>/bootstrap.ign 1
```

出力例

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

コマンドで **bootstrap.ign** を **master.ign** または **worker.ign** に置き換え、コントロールプレーンおよびコンピュートノードの Ignition 設定ファイルも利用可能であることを検証します。

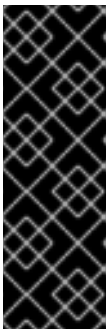
3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法

は、**openshift-install** コマンドの出力から取得することです。

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

出力例

```
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



重要

RHCOS アーティファクトは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。この手順で説明されている適切な **kernel**、**initramfs**、および **rootfs** アーティファクトのみを使用します。RHCOS QCOW2 イメージは、このインストールタイプではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン番号が含まれます。以下の例のようになります。

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. **rootfs**、**kernel**、および **initramfs** ファイルを HTTP サーバーにアップロードします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージの PXE インストールを設定し、インストールを開始します。
以下の例で示されるご使用の環境のメニューエントリーを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
(2) (3)
```

(1) (1) HTTP サーバーにアップロードしたライブ **kernel** ファイルの場所を指定します。URL は HTTP、TFTP、または FTP である必要があります。HTTPS および NFS はサポートされません。

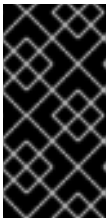
(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は **initramfs** ファイルの場所であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所、また **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。**APPEND** 行にカーネル引数を追加して、ネットワークやその他の起動オプションを設定することもできます。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? と、「高度な RHCOS インストール設定」セクションの「PXE および ISO インストール用シリアルコンソールの有効化」を参照してください。

7. マシンのコンソールで RHCOS インストールの進捗を監視します。



重要

OpenShift Container Platform のインストールを開始する前に、各ノードでインストールが成功していることを確認します。インストールプロセスを監視すると、発生する可能性のある RHCOS インストールの問題の原因を特定する上でも役立ちます。

8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

9. コンソール出力をチェックして、Ignition が実行されたことを確認します。

コマンドの例

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

10. クラスターのマシンの作成を続行します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

必要なネットワーク、DNS、およびロードバランサーインフラストラクチャーが配置されている場合、OpenShift Container Platform ブートストラッププロセスは RHCOS ノードの再起動後に自動的に起動します。

注記

RHCOS ノードには、**core** ユーザーのデフォルトのパスワードは含まれません。ノードには、**ssh core@<node>.<cluster_name>.<base_domain>** を、**install_config.yaml** ファイルで指定したパブリックキーとペアになる SSH プライベートキーへのアクセスのあるユーザーとして実行してアクセスできます。RHCOS を実行する OpenShift Container Platform 4 クラスターノードは変更できず、Operator を使用してクラスターの変更を適用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、インストールの問題を調査する際に、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、デバッグまたは障害復旧に SSH アクセスが必要になることがあります。

3.11.3. RHCOS のカーネル引数でのマルチパスの有効化

OpenShift Container Platform 4.9 以降では、インストール時に、プロビジョニングしたノードのマルチパスを有効にできます。RHCOS は、プライマリーディスクでのマルチパスをサポートします。マルチパス化により、ハードウェア障害に強力な耐障害性に利点が追加され、ホストの可用性が向上されます。

初回のクラスターの作成時に、カーネル引数をすべてのマスターまたはワーカーノードに追加しないといけない場合があります。カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入できます。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、クラスターの Kubernetes マニフェストを生成します。

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

2. カーネル引数をワーカーまたコントロールプレーンノードに追加するかどうかを決定します。

- マシン設定ファイルを作成します。たとえば、**master** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-master-kargs-mpath.yaml** を作成します。

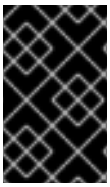
```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "master"
  name: 99-master-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

3. ワーカーノードでマルチパスを有効にするには、以下を実行します。

- マシン設定ファイルを作成します。たとえば、**worker** ラベルを追加し、マルチパスカーネル引数を指定するようクラスターに指示する **99-worker-kargs-mpath.yaml** を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "worker"
  name: 99-worker-kargs-mpath
spec:
  kernelArguments:
    - 'rd.multipath=default'
    - 'root=/dev/disk/by-label/dm-mpath-root'
```

クラスターの作成を継続できます。



重要

マルチパスを完全に有効にするには、インストール後の追加の手順が必要です。詳細は、**インストール後のマシン設定タスク** の「RHCOS でのカーネル引数を使用したマルチパスの有効化」を参照してください。

MPIO が失敗する場合は、bootlist コマンドを使用して、別の論理デバイス名でブートデバイスリストを更新します。このコマンドは、ブートリストを表示し、システムが通常モードで起動したときのブートデバイスを指定します。

a. ブートリストを表示し、システムが通常モードで起動した場合に使用可能なブートデバイスを指定するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o
sda
```

b. 通常モードのブートリストを更新し、別のデバイス名を追加するには、以下のコマンドを実行します。

```
$ bootlist -m normal -o /dev/sdc /dev/sdd /dev/sde
sdc
sdd
sde
```

元のブートディスクパスがダウンすると、ノードは通常のブートデバイスリストに登録された別のデバイスから再起動します。

3.12. ブートストラッププロセスの完了まで待機する

OpenShift Container Platform ブートストラッププロセスは、初回のクラスターノードのディスクにインストールされている永続的な RHCOS 環境での起動後に開始します。Ignition 設定ファイルで指定される設定情報は、ブートストラッププロセスを初期化し、マシンに OpenShift Container Platform をインストールするために使用されます。ブートストラッププロセスが完了するまで待機する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを作成している。
- 適切なネットワーク、DNS および負荷分散インフラストラクチャーを設定している。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- RHCOS をクラスターマシンにインストールし、OpenShift Container Platform インストールプログラムで生成される Ignition 設定ファイルを指定している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

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

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

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.25.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、ブートストラップマシン自体を削除し、再フォーマットすることができます。

3.13. CLI の使用によるクラスターへのログイン

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

前提条件

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

手順

1. **kubeadmin** 認証情報をエクスポートします。

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

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

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

```
$ oc whoami
```

出力例

```
system:admin
```

3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンに対して 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

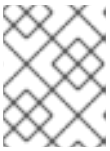
1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.25.0
master-1  Ready     master  63m  v1.25.0
master-2  Ready     master  64m  v1.25.0
```

出力には作成したすべてのマシンがリスト表示されます。



注記

上記の出力には、一部の CSR が承認されるまで、ワーカーノード (ワーカーノードとも呼ばれる) が含まれない場合があります。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

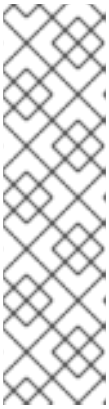
```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認された後に、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要になります。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。



注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードの ID を確認します。

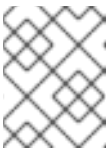
- それらを個別に承認するには、それぞれの有効な CSR に以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



注記

一部の Operator は、一部の CSR が承認されるまで利用できない可能性があります。

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR に以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

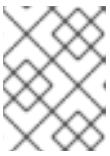
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.25.0
master-1  Ready     master  73m  v1.25.0
master-2  Ready     master  74m  v1.25.0
worker-0  Ready     worker  11m  v1.25.0
worker-1  Ready     worker  11m  v1.25.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

3.15. OPERATOR の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False     19m
baremetal                                  4.12.0    True        False         False     37m
cloud-credential                           4.12.0    True        False         False     40m
cluster-autoscaler                         4.12.0    True        False         False     37m
config-operator                            4.12.0    True        False         False     38m
```

```
console                                    4.12.0    True        False        False      26m
csi-snapshot-controller                    4.12.0    True        False        False      37m
dns                                        4.12.0    True        False        False      37m
etcd                                       4.12.0    True        False        False      36m
image-registry                             4.12.0    True        False        False      31m
ingress                                    4.12.0    True        False        False      30m
insights                                   4.12.0    True        False        False      31m
kube-apiserver                             4.12.0    True        False        False      26m
kube-controller-manager                    4.12.0    True        False        False      36m
kube-scheduler                             4.12.0    True        False        False      36m
kube-storage-version-migrator              4.12.0    True        False        False      37m
machine-api                                4.12.0    True        False        False      29m
machine-approver                           4.12.0    True        False        False      37m
machine-config                             4.12.0    True        False        False      36m
marketplace                                4.12.0    True        False        False      37m
monitoring                                 4.12.0    True        False        False      29m
network                                    4.12.0    True        False        False      38m
node-tuning                                4.12.0    True        False        False      37m
openshift-apiserver                        4.12.0    True        False        False      32m
openshift-controller-manager               4.12.0    True        False        False      30m
openshift-samples                          4.12.0    True        False        False      32m
operator-lifecycle-manager                 4.12.0    True        False        False      37m
operator-lifecycle-manager-catalog         4.12.0    True        False        False      37m
operator-lifecycle-manager-packageserver   4.12.0    True        False        False      32m
service-ca                                 4.12.0    True        False        False      38m
storage                                    4.12.0    True        False        False      37m
```

2. 利用不可の Operator を設定します。

3.15.1. デフォルトの OperatorHub カタログソースの無効化

Red Hat によって提供されるコンテンツを調達する Operator カタログおよびコミュニティープロジェクトは、OpenShift Container Platform のインストール時にデフォルトで OperatorHub に設定されます。ネットワークが制限された環境では、クラスター管理者としてデフォルトのカタログを無効にする必要があります。

手順

- **disableAllDefaultSources: true** を **OperatorHub** オブジェクトに追加して、デフォルトカタログのソースを無効にします。

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

ヒント

または、Web コンソールを使用してカタログソースを管理できます。**Administration → Cluster Settings → Configuration → OperatorHub** ページから、**Sources** タブをクリックして、個別のソースを作成、更新、削除、無効化、有効化できます。

3.15.2. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定に関する手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

3.15.2.1. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、Image Registry Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **managementState** Image Registry Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
```

3.15.2.2. IBM Power の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- IBM Power にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

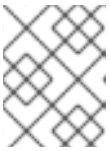
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```



注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME              VERSION       AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry    4.12          True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

3.15.2.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

3.16. USER-PROVISIONED INFRASTRUCTURE でのインストールの完了

Operator の設定が完了したら、独自に提供するインフラストラクチャーへのクラスターのインストールを完了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False      19m
baremetal                                  4.12.0    True        False         False      37m
cloud-credential                           4.12.0    True        False         False      40m
cluster-autoscaler                         4.12.0    True        False         False      37m
config-operator                            4.12.0    True        False         False      38m
console                                    4.12.0    True        False         False      26m
csi-snapshot-controller                    4.12.0    True        False         False      37m
dns                                        4.12.0    True        False         False      37m
etcd                                       4.12.0    True        False         False      36m
image-registry                             4.12.0    True        False         False      31m
ingress                                    4.12.0    True        False         False      30m
insights                                   4.12.0    True        False         False      31m
kube-apiserver                             4.12.0    True        False         False      26m
kube-controller-manager                    4.12.0    True        False         False      36m
kube-scheduler                             4.12.0    True        False         False      36m
kube-storage-version-migrator              4.12.0    True        False         False      37m
machine-api                                4.12.0    True        False         False      29m
machine-approver                           4.12.0    True        False         False      37m
machine-config                             4.12.0    True        False         False      36m
marketplace                                4.12.0    True        False         False      37m
monitoring                                 4.12.0    True        False         False      29m
network                                    4.12.0    True        False         False      38m
node-tuning                                4.12.0    True        False         False      37m
openshift-apiserver                        4.12.0    True        False         False      32m
openshift-controller-manager               4.12.0    True        False         False      30m
openshift-samples                          4.12.0    True        False         False      32m
operator-lifecycle-manager                 4.12.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False      32m
service-ca                                 4.12.0    True        False         False      38m
storage                                    4.12.0    True        False         False      37m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

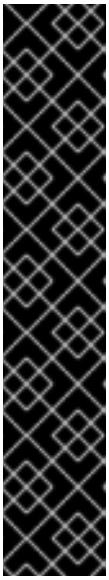
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

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

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

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

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

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod のリストを表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                               1/1     Running     0
   3m
   openshift-apiserver             apiserver-ljcmx                               1/1     Running     0
   1m
   openshift-apiserver             apiserver-z25h4                               1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力にリスト表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. マルチパスを有効にするための追加の手順が必要です。インストール時にマルチパスを有効にしないでください。
   詳細は、**インストール後のマシン設定タスク** ドキュメントの RHCOS でのカーネル引数を使用したマルチパスの有効化を参照してください。

4. Cluster registration ページでクラスターを登録します。

3.17. 次のステップ

- RHCOS のカーネル引数でのマルチパスの有効化
- クラスターのカスタマイズ
- クラスターのインストールに使用したミラーレジストリーに信頼された CA がある場合は、追加のトラストストアを設定 してその CA をクラスターに追加します。
- 必要に応じて、リモートヘルスレポート を作成できます。
- 必要に応じて、非接続クラスターの登録 を参照してください。