OpenShift Container Platform 4.18

任意のプラットフォームへのインストール

任意のプラットフォームへの OpenShift Container Platform のインストール

Last Updated: 2025-12-05

OpenShift Container Platform 4.18 任意のプラットフォームへのインストール

任意のプラットフォームへの OpenShift Container Platform のインストール

Legal Notice

Abstract

このドキュメントでは、OpenShift Container Platform を任意のプラットフォームにインストールする方法を説明します。

Table of Contents

第1章 クラスターの任意のプラットフォームへのインストール

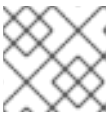
OpenShift Container Platform バージョン 4.18 では、仮想化およびクラウド環境を含め、独自にプロビジョニングする任意のインフラストラクチャーにクラスターをインストールできます。

重要

仮想化またはクラウド環境で OpenShift Container Platform クラスターのインストールを試行する前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

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。
- 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。

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

OpenShift Container Platform 4.18 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。

次のアクションを実行するには、インターネットにアクセスできる必要があります。

- OpenShift Cluster Manager 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。クラスターがインターネットにアクセスでき、Telemetry を無効にしていない場合、そのサービスによってクラスターのサブスクリプションが自動的に有効化されます。
- 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。

1.3. 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。

1.3.1. 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1.1 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。

ブートストラップおよびコントロールプレーンマシンでは、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。ただし、コンピュートマシンは、Red Hat Enterprise Linux CoreOS (RHCOS)、Red Hat Enterprise Linux (RHEL) 8.6 以降から選択できます。

RHCOS は Red Hat Enterprise Linux (RHEL) 9.2 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。

1.3.2. 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。

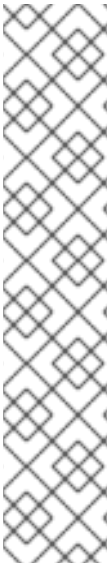
表1.2 最小リソース要件

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	4	16 GB	100 GB	300
コントロールプレーン	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS、RHEL 8.6 以降 [3]	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、(コアあたりのスレッド数 x コア数) x ソケット数 = 仮想 CPU という数式を使用して対応する比率を計算します。

2. OpenShift Container Platform および Kubernetes はディスクのパフォーマンスに敏感であり、特に 10 ms p99 fsync 期間を必要とするコントロールプレーンノード上の etcd には、高速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 が連動してスケーリングされるため、十分なパフォーマンスを得るには、ストレージボリュームを過剰に割り当てる必要がある場合がある点に注意してください。

3. すべての user-provisioned installation と同様に、クラスターで RHEL コンピュートマシンの使用を選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL 7 コンピュートマシンの使用は非推奨となり、OpenShift Container Platform 4.10 以降で削除されています。



注記

OpenShift Container Platform バージョン 4.18 の場合、RHCOS は RHEL バージョン 9.4 に基づいており、マイクロアーキテクチャー要件が更新されています。次のリストには、各アーキテクチャーに必要な最小限の命令セットアーキテクチャー (ISA) が含まれています。

- x86-64 アーキテクチャーには x86-64-v2 ISA が必要
- ARM64 アーキテクチャーには ARMv8.0-A ISA が必要
- IBM Power アーキテクチャーには Power 9 ISA が必要
- s390x アーキテクチャーには z14 ISA が必要

詳細は、アーキテクチャー (RHEL ドキュメント) を参照してください。

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。

1.3.3. 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。

1.3.4. 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。

注記

- 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。
- 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。

1.3.4.1. 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。

1.3.4.2. 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1.3 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
	22623	このポートは、マシン設定サーバーからのトラフィックを処理し、トラフィックをコントロールプレーンマシンに送信します。
UDP	**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

表1.4 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1.5 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

1.3.4.3. 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

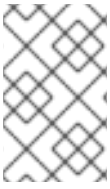
1.3.5. user-provisioned DNS 要件

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。

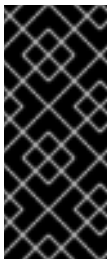


注記

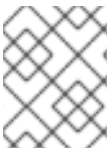
各クラスターノードにホスト名を提供するために DHCP サーバーを使用することが推奨されます。詳細は、**user-provisioned infrastructure に関する DHCP の推奨事項**のセクションを参照してください。

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1.6 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。

コンポーネント	レコード	説明
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。

1.3.5.1. 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1.1 DNS ゾーンデータベースのサンプル

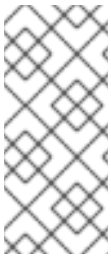
```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
```

```
;
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1.2 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。

1.3.6. user-provisioned infrastructure 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。

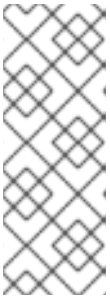
重要

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1.7 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1.8 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。

1.3.6.1. user-provisioned 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、user-provisioned 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1.3 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 compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
```

```
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.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。

1.4. 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。

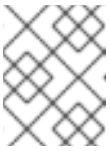
手順

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。

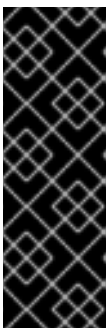
   c. DHCP サーバー設定でクラスターノードのホスト名を定義します。ホスト名に関する考慮事項の詳細は、**DHCP を使用したクラスターノードのホスト名の設定** セクションを参照してください。



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。

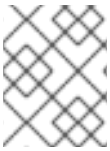
    b. Kubernetes API、ブートストラップマシン、コントロールプレーンマシン、およびコンピュートマシンの逆引き DNS 解決を設定します。
    OpenShift Container Platform DNS 要件の詳細は、**user-provisioned DNS 要件** のセクションを参照してください。

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。

1.5. 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。

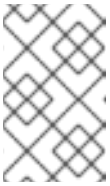
```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

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。

1.6. 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

プラットフォーム固有の方法で設定した鍵ではなく、ローカルの鍵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

x86_64、**ppc64le**、および **s390x** アーキテクチャーのみで FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用する OpenShift Container Platform クラスターをインストールする予定がある場合は、**ed25519** アルゴリズムを使用するキーを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。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、キーをインストールプログラムに指定する必要があります。

1.7. 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。

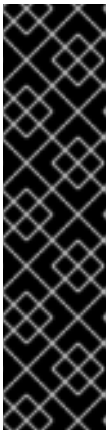
手順

1. Red Hat Hybrid Cloud Console の Cluster Type ページに移動します。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。

   ヒント

   特定の OpenShift Container Platform リリースのバイナリーをダウンロード することもできます。

2. ページの **Run it yourself** セクションからインフラストラクチャープロバイダーを選択します。
3. **OpenShift Installer** のドロップダウンメニューからホストオペレーティングシステムとアーキテクチャーを選択し、**Download Installer** をクリックします。
4. ダウンロードしたファイルを、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。クラスターを削除するには、両方のファイルが必要です。
- 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。

5. 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
```

6. 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。

ヒント

Red Hat カスタマーポータル からインストールプログラムを取得することもできます。このページでは、ダウンロードするインストールプログラムのバージョンを指定できます。ただし、このページにアクセスするには、有効なサブスクリプションが必要です。

1.8. OPENSHIFT CLI のインストール

OpenShift CLI (**oc**) をインストールすると、コマンドラインインターフェイスから OpenShift Container Platform を操作できます。**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 4.18 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。

1.8.1. 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 v4.18 Linux Clients** 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>
```

1.8.2. 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 v4.18 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展開します。

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>
```

1.8.3. 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 v4.18 macOS Clients** 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 v4.18 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

- **oc** コマンドを使用してインストールを確認します。

```
$ oc <command>
```

1.9. 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

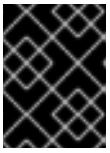
このディレクトリーは必ず作成してください。ブートストラップ X.509 証明書などの一部のインストールアセットは、有効期限が短いため、インストールディレクトリーを再利用しないでください。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してください。

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。

1.9.1. 他のプラットフォーム用のサンプル 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。

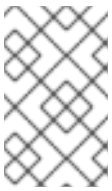
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
```

```
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。



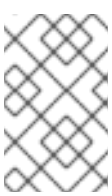
注記

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。サポートされる値はデフォルト値の **OVNKubernetes** のみです。

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。プラットフォーム用に追加のプラットフォーム設定変数を指定することはできません。

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トマシンの管理など、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 FIPS モードを設定する方法の詳細は、RHEL から FIPS モードへの切り替え を参照してください。

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

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。

1.9.2. 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)、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。

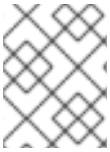
注記

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。

1.9.3. 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。

1.10. 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。

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
```

1.11. 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ベアメタル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。

RHCOS をマシンにインストールするには、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のいずれかを実行します。

注記

このインストールガイドに含まれるコンピュートノードのデプロイメント手順は、RHCOS 固有のものです。代わりに RHEL ベースのコンピュートノードのデプロイを選択する場合は、システム更新の実行、パッチの適用、その他すべての必要なタスクの完了など、オペレーティングシステムのライフサイクルの管理と保守をすべて担当します。RHEL8 コンピュートマシンのみがサポートされています。

以下の方法を使用して、ISO および PXE のインストール時に RHCOS を設定できます。

- カーネル引数: カーネル引数を使用してインストール固有の情報を提供できます。たとえば、HTTP サーバーにアップロードした RHCOS インストールファイルの場所と、インストールするノードタイプの Ignition 設定ファイルの場所を指定できます。PXE インストールの場合、**APPEND** パラメーターを使用して、ライブインストーラーのカーネルに引数を渡すことができます。ISO インストールの場合は、ライブ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。いずれのインストールの場合でも、特殊な **coreos.inst.*** 引数を使用してライブインストーラーに指示を与えたり、標準のカーネルサービスをオンまたはオフにするために標準のインストールブート引数を使用したりできます。

- Ignition 設定: OpenShift Container Platform Ignition 設定ファイル (***.ign**) は、インストールするノードのタイプに固有のものです。RHCOS のインストール時にブートストラップ、コントロールプレーン、またはコンピュートノードの Ignition 設定ファイルの場所を渡して、初回起動時に有効にされるようにします。特別なケースでは、ライブシステムに渡すために個別の制限付き Ignition 設定を作成できます。この Ignition 設定は、インストールが正常に完了したことをプロビジョニングシステムに報告するなどの一連のタスクを実行する可能性があります。この特別な Ignition 設定は、インストール済みシステムの初回ブート時に適用される **coreos-installer** によって使用されます。標準のコントロールプレーンおよびコンピュートノードの Ignition 設定をライブ ISO に直接指定しないでください。

- **coreos-installer**: ライブ ISO インストーラーをシェルプロンプトで起動できます。これにより、初回のブート前にさまざまな方法で永続的なシステムの準備を行うことができます。特に、**coreos-installer** コマンドを実行すると、追加するさまざまなアーティファクトを特定し、ディスクパーティションを使用し、ネットワークを設定できます。場合によっては、ライブシステムで機能を設定し、それらをインストールされたシステムにコピーできます。

注記

バージョン **0.17.0-3** 以降、**coreos-installer** プログラムを実行するには RHEL 9 以降が必要です。古いバージョンの **coreos-installer** を使用して、新しい OpenShift Container Platform リリースの RHCOS アーティファクトをカスタマイズし、メタルイメージをディスクにインストールすることもできます。**coreos-installer** バイナリーの古いバージョンは、**coreos-installer** image mirror ページからダウンロードできます。

ISO または PXE インストールを使用するかどうかは、状況によって異なります。PXE インストールには、利用可能な DHCP サービスとさらなる準備が必要ですが、インストールプロセスはさらに自動化することが可能です。ISO インストールは主に手動によるプロセスで、複数のマシンを設定する場合には使用しにくい可能性があります。

1.11.1. 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。

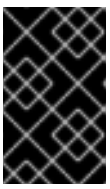
手順

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-4.18-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.18-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.18-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.18/<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。

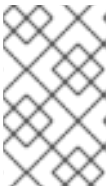
ISO ファイルの名前は以下の例のようになります。

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

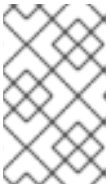
RHCOS インストール起動プロセスを中断して、カーネル引数を追加できます。ただし、この ISO 手順では、カーネル引数を追加する代わりに、以下の手順で説明しているように **coreos-installer** コマンドを使用する必要があります。

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、該当するノードタイプ用の Ignition 設定ファイルを指す 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>
\ 1
--ignition-hash=sha512-<digest> 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

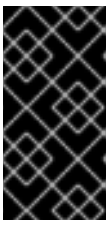
TLS を使用する HTTPS サーバーを使用して Ignition 設定ファイルを提供する場合は、**coreos-installer** を実行する前に、内部認証局 (CA) をシステムのトラストストアに追加できます。

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-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf0116e80c59
d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

1. 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。

2. 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。

3. 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
```

4. 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。

1.11.2. PXE または iPXE ブートを使用した RHCOS のインストール

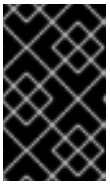
PXE または iPXE ブートを使用してマシンに RHCOS をインストールできます。

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 または iPXE 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。

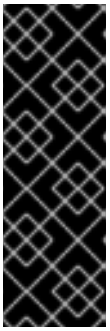
3. RHCOS イメージミラー ページからオペレーティングシステムインスタンスをインストールするための推奨される方法に必要な RHCOS **kernel**、**initramfs**、および **rootfs** ファイルを取得することは可能ですが、RHCOS ファイルの正しいバージョンを取得するための推奨される方法は、**openshift-install** コマンドの出力から取得することです。

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

出力例

```
"<url>/art/storage/releases/rhcos-4.18-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.18-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.18-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.18-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.18-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.18-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.18-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.18-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.18-s390x/<release>/s390x/rhcos-<release>-live-
```

```
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.18/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.18/<release>/x86_64/rhcos-<release>-live-
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.18/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```



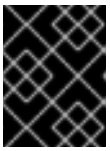
重要

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 または iPXE インストールを設定し、インストールを開始します。ご使用の環境に関する以下の例で示されるメニューエントリーのいずれかを変更し、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。

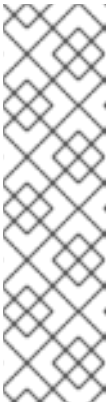
- PXE(**x86_64**) の場合:

```
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

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。

- iPXE (**x86_64** + **aarch64**) の場合:

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
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 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img 3
boot
```

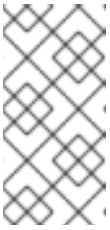
1 HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd=main** 引数は UEFI システムでの起動に必要であり、**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

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 **initramfs** ファイルの場所を指定します。

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、**kernel** 行に **console=** 引数を 1 つ以上追加します。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。



注記

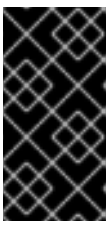
aarch64 アーキテクチャーで CoreOS **kernel** をネットワークブートするには、**IMAGE_GZIP** オプションが有効になっているバージョンの iPXE ビルドを使用する必要があります。iPXE の **IMAGE_GZIP** オプション を参照してください。

- **aarch64** 上の PXE (第 2 段階として UEFI と Grub を使用) の場合:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
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 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1. HTTP/TFTP 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。**kernel** パラメーター値は、TFTP サーバー上の **kernel** ファイルの場所になります。**coreos.live.rootfs_url** パラメーター値は **rootfs** ファイルの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバー上のブートストラップ Ignition 設定ファイルの場所になります。
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. TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

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。

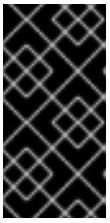
8. RHCOS のインストール後に、システムは再起動します。再起動中、システムは指定した Ignition 設定ファイルを適用します。

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
```

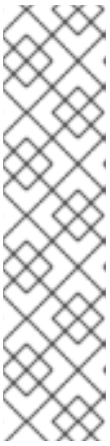
10. クラスターのマシンの作成を続行します。



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。

1.11.3. 高度な RHCOS インストール設定

OpenShift Container Platform 用の Red Hat Enterprise Linux CoreOS (RHCOS) ノードを手動でプロビジョニングする主な利点として、デフォルトの OpenShift Container Platform インストール方法では利用できない設定を実行できることがあります。このセクションでは、以下のような手法で実行できるいくつかの設定を説明します。

- カーネル引数をライブインストーラーに渡す
- ライブシステムからの **coreos-installer** の手動による実行
- ライブ ISO または PXE ブートイメージのカスタマイズ

このセクションで説明されている手動の Red Hat Enterprise Linux CoreOS (RHCOS) インストールの高度な設定トピックは、ディスクパーティション設定、ネットワーク、および複数の異なる方法での Ignition 設定の使用に関連しています。

1.11.3.1. PXE および ISO インストールの高度なネットワークオプションの使用

OpenShift Container Platform ノードのネットワークはデフォルトで DHCP を使用して、必要な設定をすべて収集します。静的 IP アドレスを設定したり、ボンディングなどの特別な設定を行う場合は、以下のいずれかの方法で実行できます。

- ライブインストーラーの起動時に、特別なカーネルパラメーターを渡します。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。
- ライブインストーラーのシェルプロンプトからネットワークを設定し、それらの設定をインストール済みシステムにコピーして、インストール済みシステムの初回起動時に有効になるようにします。

PXE または iPXE インストールを設定するには、以下のオプションのいずれかを使用します。

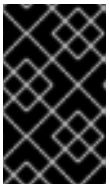
- 「詳細な RHCOS インストールリファレンスの表」を参照してください。
- マシン設定を使用してネットワークファイルをインストール済みシステムにコピーします。

ISO インストールを設定するには、以下の手順に従います。

手順

1. ISO インストーラーを起動します。
2. ライブシステムシェルプロンプトから、**nmcli** または **nmtui** などの利用可能な RHEL ツールを使用して、ライブシステムのネットワークを設定します。
3. **coreos-installer** コマンドを実行してシステムをインストールし、**--copy-network** オプションを追加してネットワーク設定をコピーします。以下に例を示します。

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```



重要

--copy-network オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。

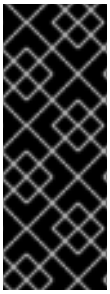
4. インストール済みのシステムで再起動します。

関連情報

- **nmcli** ツールおよび **nmtui** ツールの詳細は、RHEL 8 ドキュメントの Getting started with nmcli および Getting started with nmtui を参照してください。

1.11.3.2. ディスクパーティション設定

ディスクパーティションは、Red Hat Enterprise Linux CoreOS (RHCOS) のインストール時に OpenShift Container Platform クラスターノードに作成されます。デフォルトのパーティション設定をオーバーライドしない限り、特定のアーキテクチャーの各 RHCOS ノードで同じパーティションレイアウトが使用されます。RHCOS のインストール時に、ルートファイルシステムのサイズが拡大し、ターゲットデバイスの残りの使用可能なスペースが使用されます。



重要

ノードでカスタムパーティションスキームを使用すると、OpenShift Container Platform が一部のノードパーティションでモニタリングやアラートを行わなくなる可能性があります。デフォルトのパーティション設定をオーバーライドする場合は、OpenShift Container Platform がホストファイルシステムを監視する方法の詳細について Understanding OpenShift File System Monitoring (eviction conditions) を参照してください。

OpenShift Container Platform は、次の 2 つのファイルシステム識別子を監視します。

- **nodefs**: **/var/lib/kubelet** を含むファイルシステム
- **imagefs**: **/var/lib/containers** を含むファイルシステム

デフォルトのパーティションスキームの場合、**nodefs** と **imagefs** は同じルートファイルシステム (/) を監視します。

RHCOS を OpenShift Container Platform クラスターノードにインストールするときにデフォルトのパーティション設定をオーバーライドするには、別のパーティションを作成する必要があります。コンテナーとコンテナーイメージ用に別のストレージパーティションを追加する状況を考えてみましょう。たとえば、**/var/lib/containers** を別のパーティションにマウントすると、kubelet が **/var/lib/containers** を **imagefs** ディレクトリーとして、ルートファイルシステムを **nodefs** ディレクトリーとして個別に監視します。

重要

より大きなファイルシステムをホストするためにディスクサイズを変更した場合は、別の **/var/lib/containers** パーティションを作成することを検討してください。多数の割り当てグループによって発生する CPU 時間の問題を軽減するには、**xfs** 形式のディスクのサイズを変更することを検討してください。

1.11.3.2.1. 個別の /var パーティションの作成

通常は、RHCOS のインストール時に作成されるデフォルトのディスクパーティションを使用する必要があります。ただし、拡張するディレクトリーの個別のパーティションの作成が必要となる場合もあります。

OpenShift Container Platform は、ストレージを **/var** ディレクトリーまたは **/var** のサブディレクトリーのいずれかに割り当てる単一のパーティションの追加をサポートします。以下に例を示します。

- **/var/lib/containers**: イメージやコンテナーがシステムにさらに追加されると拡張するコンテナー関連のコンテンツを保持します。
- **/var/lib/etcd**: etcd ストレージのパフォーマンスの最適化などの目的で分離する必要のあるデータを保持します。
- **/var**: 監査などの目的に合わせて分離させる必要のあるデータを保持します。

重要

ディスクサイズが 100 GB を超える場合、特に 1 TB を超える場合は、別の **/var** パーティションを作成します。

/var ディレクトリーのコンテンツを個別に保存すると、必要に応じてこれらの領域のストレージの拡大を容易にし、後で OpenShift Container Platform を再インストールして、そのデータをそのまま保持することができます。この方法では、すべてのコンテナーを再度プルする必要はありません。また、システムの更新時に大きなログファイルをコピーする必要もありません。

/var ディレクトリーまたは **/var** のサブディレクトリーの個別のパーティションを使用すると、パーティション設定されたディレクトリーでのデータの増加によりルートファイルシステムが一杯になることを避けることもできます。

以下の手順では、インストールの準備フェーズでノードタイプの Ignition 設定ファイルにラップされるマシン設定マニフェストを追加して、別の **/var** パーティションを設定します。

手順

1. インストールホストで、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>
```

2. 追加のパーティションを設定する Butane 設定を作成します。たとえば、**$HOME/clusterconfig/98-var-partition.bu** ファイルに名前を付け、ディスクのデバイス名を **worker** システムのストレージデバイスの名前に変更し、必要に応じてストレージサイズを設定します。以下の例では、**/var** ディレクトリーを別のパーティションにマウントします。

```
variant: openshift
version: 4.18.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1. パーティションを設定する必要のあるディスクのストレージデバイス名。
2. データパーティションをブートディスクに追加する場合は、25000 のメビバイトの最小のオフセット値が推奨されます。ルートファイルシステムは、指定したオフセットまでの利用可能な領域をすべて埋めるためにサイズを自動的に変更します。オフセット値の指定がない場合や、指定した値が推奨される最小値よりも小さい場合、生成されるルートファイルシステムのサイズは小さ過ぎるため、RHCOS の再インストールでデータパーティションの最初の部分が上書きされる可能性があります。

3 データパーティションのサイズ (メビバイト単位)。

4 コンテナーストレージに使用されるファイルシステムでは、**prjquota** マウントオプションを有効にする必要があります。

注記

個別の **/var** パーティションを作成する場合、異なるインスタンスタイプに同じデバイス名がない場合は、コンピュートノードに異なるインスタンスタイプを使用することはできません。

3. Butane config からマニフェストを作成し、**clusterconfig/openshift** ディレクトリーに保存します。たとえば、以下のコマンドを実行します。

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

4. Ignition 設定ファイルを作成します。

```
$ 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。

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

<installation_directory>/manifest ディレクトリーおよび **<installation_directory>/openshift** ディレクトリーのファイルは、**98-var-partition** カスタム **MachineConfig** オブジェクトが含まれるファイルを含む Ignition 設定ファイルにラップされます。

次のステップ

- RHCOS のインストール時に Ignition 設定ファイルを参照して、カスタムディスクのパーティション設定を適用することができます。

1.11.3.2.2. 既存パーティションの保持

ISO インストールの場合は、インストーラーに 1 つ以上の既存パーティションを維持させる **coreos-installer** コマンドにオプションを追加することができます。PXE インストールの場合、**coreos.inst.*** オプションを **APPEND** パラメーターに追加して、パーティションを保持できます。

保存したパーティションは、既存の OpenShift Container Platform システムからのデータパーティションである可能性があります。パーティションラベルまたは番号のいずれかで保持する必要のあるディスクパーティションを特定できます。

注記

既存のパーティションを保存し、それらのパーティションが RHCOS の十分な領域を残さない場合、インストールは失敗します。この場合、保存したパーティションが破損することはありません。

ISO インストール時の既存パーティションの保持

この例では、パーティションラベルが **data** (**data***) で始まるパーティションを保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
--save-partlabel 'data*' \
/dev/disk/by-id/scsi-<serial_number>
```

以下の例では、ディスク上の 6 番目のパーティションを保持する方法で **coreos-installer** を実行する方法を説明しています。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
--save-partindex 6 /dev/disk/by-id/scsi-<serial_number>
```

この例では、パーティション 5 以上を保持します。

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
--save-partindex 5- /dev/disk/by-id/scsi-<serial_number>
```

パーティションの保存が使用された以前の例では、**coreos-installer** はパーティションをすぐに再作成します。

PXE インストール時の既存パーティションの保持

この **APPEND** オプションは、パーティションラベルが 'data' ('data*') で始まるパーティションを保持します。

```
coreos.inst.save_partlabel=data*
```

この **APPEND** オプションは、パーティション 5 以上を保持します。

```
coreos.inst.save_partindex=5-
```

この **APPEND** オプションは、パーティション 6 を保持します。

```
coreos.inst.save_partindex=6
```

1.11.3.3. Ignition 設定の特定

RHCOS の手動インストールを実行する場合、提供できる Ignition 設定には 2 つのタイプがあり、それぞれを提供する理由もそれぞれ異なります。

- **Permanent install Ignition config**: すべての手動の RHCOS インストールは、**bootstrap.ign**、**master.ign**、および **worker.ign** などの **openshift-installer** が生成した Ignition 設定ファイルのいずれかを渡し、インストールを実行する必要があります。

重要

これらの Ignition 設定ファイルを直接変更することは推奨されません。前述のセクションの例で説明されているように、Ignition 設定ファイルにラップされるマニフェストファイルを更新できます。

PXE インストールの場合、**coreos.inst.ignition_url=** オプションを使用して、**APPEND** 行に Ignition 設定を渡します。ISO インストールの場合、シェルプロンプトで ISO を起動した後に、**--ignition-url=** オプションを指定した **coreos-installer** コマンドラインで Ignition 設定を特定します。いずれの場合も、HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。

- **Live install Ignition config**: このタイプは、**coreos-installer customize** サブコマンドとそのさまざまなオプションを使用して作成できます。この方法では、Ignition 設定はライブインストールメディアに渡され、起動直後に実行され、RHCOS システムがディスクにインストールされる前または後にセットアップタスクを実行します。この方法は、マシン設定を使用して実行できない高度なパーティション設定など、一度の適用後に再度適用する必要のないタスクの実行にのみ使用する必要があります。

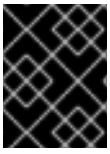
PXE または ISO ブートの場合、Ignition 設定を作成し、**ignition.config.url=** オプションに対して **APPEND** を実行し、Ignition 設定の場所を特定できます。また、**ignition.firstboot ignition.platform.id=metal** も追加する必要があります。追加しない場合は、**ignition.config.url** が無視されます。

1.11.3.4. 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。

1.11.3.4.1. 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

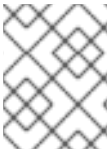
ISO イメージから RHCOS をインストールする場合、そのイメージを起動してノードのネットワークを設定する際に手動でカーネル引数を追加できます。ネットワークの引数が指定されていない場合、RHCOS が Ignition 設定ファイルを取得するためにネットワークが必要であることを検知する際に、DHCP が initramfs でアクティベートされます。



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。

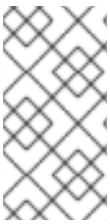
次の例は、ISO インストールのネットワークオプションです。

1.11.3.4.1.1. 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。

1.11.3.4.1.2. 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
```

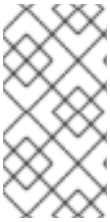
1.11.3.4.1.3. 複数のネットワークインターフェイスの指定

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
```

1.11.3.4.1.4. 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
  ```

1.11.3.4.1.5. 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
```

1.11.3.4.1.6. 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
```

1.11.3.4.1.7. 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
```

1.11.3.4.1.8. 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
```

1.11.3.4.1.9. 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ための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は物理 (イーサネット) インターフェイスのコンマ区切りのリスト (**em1,em2**) を表し、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
```

1.11.3.4.1.10. 複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合する

オプション: **bond=** オプションを使用して、複数の SR-IOV ネットワークインターフェイスをデュアルポート NIC インターフェイスに結合できます。

各ノードで、次のタスクを実行する必要があります。

1. SR-IOV デバイスの管理 のガイダンスに従って、SR-IOV 仮想機能 (VF) を作成します。「仮想マシンへの SR-IOV ネットワークデバイスの接続」セクションの手順に従います。
2. ボンドを作成し、目的の VF をボンドに接続し、ネットワークボンディングの設定 のガイダンスに従って、ボンドリンクの状態を設定します。説明されている手順のいずれかに従って、結合を作成します。

次の例は、使用する必要がある構文を示しています。

- 結合インターフェイスを設定するための構文は、**bond=<name>[:<network_interfaces>][:options]** です。
  <name> はボンディングデバイス名 (**bond0**)、**<network_interfaces>** は仮想機能 (VF) をカーネル内の既知の名前で表し、**ip link** コマンド (**eno1f0**、**eno2f0**) の出力に表示されます。**options** は結合オプションのコンマ区切りリストです。**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:eno1f0,eno2f0:mode=active-backup
    ip=bond0:dhcp
    ```

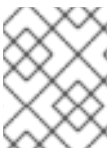
  - 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。

    ```
    bond=bond0:eno1f0,eno2f0:mode=active-backup
    ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
    ```

1.11.3.4.1.11. 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
```

-

1.11.3.4.2. ISO および PXE インストール用のcoreos-installer オプション

RHCOS は、ISO イメージから RHCOS ライブ環境に起動した後に、コマンドプロンプトで **coreos-installer install <options> <device>** を実行してインストールできます。

以下の表は、**coreos-installer** コマンドに渡すことのできるサブコマンド、オプションおよび引数を示しています。

表1.9 coreos-installer サブコマンド、コマンドラインオプション、および引数

coreos-installer install サブコマンド	
サブコマンド	**説明**
$ coreos-installer install <options> <device>	Ignition 設定を ISO イメージに埋め込みます。
coreos-installer install サブコマンドオプション	
オプション	**説明**
-u、--image-url <url>	イメージの URL を手動で指定します。
-f、--image-file <path>	ローカルイメージファイルを手動で指定します。デバッグに使用されます。
-i、--ignition-file <path>	ファイルから Ignition 設定を埋め込みます。
-I、--ignition-url <URL>	URL から Ignition 設定を埋め込みます。
--ignition-hash <digest>	Ignition 設定の **type-value** をダイジェスト値を取得します。
-p、--platform <name>	インストール済みシステムの Ignition プラットフォーム ID を上書きします。
--console <spec>	インストールされたシステムのカーネルとブートローダーコンソールを設定します。**<spec>** の形式の詳細は、Linux カーネルシリアルコンソール のドキュメントを参照してください。
--append-karg <arg>...	インストール済みシステムにデフォルトのカーネル引数を追加します。
--delete-karg <arg>...	インストール済みシステムからデフォルトのカーネル引数を削除します。

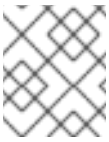
-n、--copy-network	インストール環境からネットワーク設定をコピーします。 **重要** **--copy-network** オプションは、**/etc/NetworkManager/system-connections** にあるネットワーク設定のみをコピーします。特に、システムのホスト名はコピーされません。
--network-dir <path>	**-n** を指定して使用する場合。デフォルトは **/etc/NetworkManager/system-connections/** です。
--save-partlabel <lx>..	このラベル glob でパーティションを保存します。
--save-partindex <id>...	この数または範囲でパーティションを保存します。
--insecure	RHCOS イメージ署名の検証を省略します。
--insecure-ignition	HTTPS またはハッシュなしで Ignition URL を許可します。
--architecture <name>	ターゲット CPU アーキテクチャー。有効な値は **x86_64** および **aarch64** です。
--preserve-on-error	エラー時のパーティションテーブルは消去しないでください。
-h、--help	ヘルプ情報を表示します。
coreos-installer インストールサブコマンド引数	
引数	**説明**
<device>	宛先デバイス。
coreos-installer ISO サブコマンド	
サブコマンド	**説明**

$ coreos-installer iso customize <options> <ISO_image>	RHCOS ライブ ISO イメージをカスタマイズします。
coreos-installer iso reset <options> <ISO_image>	RHCOS ライブ ISO イメージをデフォルト設定に復元します。
coreos-installer iso ignition remove <options> <ISO_image>	ISO イメージから埋め込まれた Ignition 設定を削除します。

coreos-installer ISO カスタマイズサブコマンドオプション

オプション	説明
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--dest-karg-append <arg>	宛先システムの各起動にカーネル引数を追加します。
--dest-karg-delete <arg>	宛先システムの各起動からカーネル引数を削除します。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。
--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
--post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
--live-karg-append <arg>	ライブ環境の各ブートにカーネル引数を追加します。

--live-karg-delete <arg>	ライブ環境の各ブートからカーネル引数を削除します。
--live-karg-replace <k=o=n>	ライブ環境の各起動で、**key=old=new** の形式でカーネル引数を置き換えます。
-f, --force	既存の Ignition 設定を上書きします。
-o、--output <path>	新しい出力ファイルに ISO を書き込みます。
-h、--help	ヘルプ情報を表示します。
coreos-installer PXE サブコマンド	
サブコマンド	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
coreos-installer pxe customize <options> <path>	RHCOS ライブ PXE ブート設定をカスタマイズします。
coreos-installer pxe ignition wrap <options>	イメージに Ignition 設定をラップします。
coreos-installer pxe ignition unwrap <options> <image_name>	イメージでラップされた Ignition 設定を表示します。
coreos-installer PXE はサブコマンドオプションをカスタマイズします	
オプション	**説明**
これらのオプションすべてがすべてのサブコマンドで使用できる訳ではないことに注意してください。	
--dest-ignition <path>	指定された Ignition 設定ファイルを宛先システムの新しい設定フラグメントにマージします。
--dest-console <spec>	宛先システムのカーネルとブートローダーコンソールを指定します。
--dest-device <path>	指定した宛先デバイスをインストールして上書きします。
--network-keyfile <path>	ライブシステムと宛先システムに指定された NetworkManager キーファイルを使用してネットワークを設定します。
--ignition-ca <path>	Ignition によって信頼される追加の TLS 認証局を指定します。

--pre-install <path>	インストールする前に、指定されたスクリプトを実行します。
post-install <path>	インストール後に指定されたスクリプトを実行します。
--installer-config <path>	指定されたインストーラー設定ファイルを適用します。
--live-ignition <path>	指定された Ignition 設定ファイルをライブ環境の新しい設定フラグメントにマージします。
-o, --output <path>	initramfs を新しい出力ファイルに書き込みます。  **注記** このオプションは、PXE 環境に必要です。
-h、--help	ヘルプ情報を表示します。

1.11.3.4.3. ISO または PXE インストールの**coreos.inst** ブートオプション

coreos.inst ブートパラメーターを RHCOS ライブインストーラーに渡して、ブート時に **coreos-installer** オプションを自動的に起動できます。これらは、標準のブート引数の追加として提供されます。

- ISO インストールの場合、ブートローダーメニューで自動ブートを中断して **coreos.inst** オプションを追加できます。**RHEL CoreOS (Live)** メニューオプションが強調表示されている状態で **TAB** を押すと、自動ブートを中断できます。
- PXE または iPXE インストールの場合、RHCOS ライブインストーラーのブート前に **coreos.inst** オプションを **APPEND** 行に追加する必要があります。

以下の表は、ISO および PXE インストールの RHCOS ライブインストーラーの **coreos.inst** ブートオプションを示しています。

表1.10 coreos.inst ブートオプション

引数	説明

引数	説明
coreos.inst.install_dev	必須。インストール先のシステムのブロックデバイス。  **注記** **sda** は許可されていますが、**/dev/sda** などの完全パスを使用することが推奨されます。
coreos.inst.ignition_url	オプション: インストール済みシステムに埋め込む Ignition 設定の URL。URL が指定されていない場合、Ignition 設定は埋め込まれません。HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.save_partlabel	オプション: インストール時に保存するパーティションのコンマ区切りのラベル。glob 形式のワイルドカードが許可されます。指定したパーティションは存在する必要はありません。
coreos.inst.save_partindex	オプション: インストール時に保存するパーティションのコンマ区切りのインデックス。範囲 **m-n** は許可され、**m** または **n** のいずれかを省略できます。指定したパーティションは存在する必要はありません。
coreos.inst.insecure	オプション: **coreos.inst.image_url** で署名なしと指定される OS イメージを許可します。

引数	説明
coreos.inst.image_url	オプション: 指定した RHCOS イメージをダウンロードし、インストールします。 ● この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。 ● この引数は、ライブメディアに一致しないバージョンの RHCOS をインストールするために使用できますが、インストールするバージョンに一致するメディアを使用することが推奨されます。 ● **coreos.inst.image_url** を使用している場合は、**coreos.inst.insecure** も使用する必要があります。これは、ベアメタルメディアが OpenShift Container Platform について GPG で署名されていないためです。 ● HTTP プロトコルおよび HTTPS プロトコルのみがサポートされます。
coreos.inst.skip_reboot	オプション: システムはインストール後に再起動しません。インストールが完了するとプロンプトが表示され、インストール時に生じる内容を検査できます。この引数は実稼働環境では使用できず、デバッグの目的でのみ使用することが意図されています。
coreos.inst.platform_id	オプション: RHCOS イメージがインストールされるプラットフォームの Ignition プラットフォーム ID。デフォルトは **metal** です。このオプションは、VMware などのクラウドプロバイダーから Ignition 設定を要求するかどうかを決定します。例: **coreos.inst.platform_id=vmware**
ignition.config.url	オプション: ライブ起動の Ignition 設定の URL。たとえば、これは **coreos-installer** の起動方法をカスタマイズしたり、インストール前後にコードを実行するために使用できます。これはインストール済みシステムの Ignition 設定である **coreos.inst.ignition_url** とは異なります。

1.12. 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。
- 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。

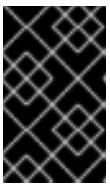
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.31.3 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。

1.13. 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。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 次のコマンドを実行して、**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. 次のコマンドを実行し、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
```

1.14. 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  v1.31.3
master-1 Ready    master  63m  v1.31.3
master-2 Ready    master  64m  v1.31.3
```

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
```

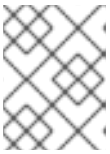
6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.31.3
master-1  Ready     master  73m  v1.31.3
```

```
master-2  Ready    master  74m  v1.31.3
worker-0  Ready    worker  11m  v1.31.3
worker-1  Ready    worker  11m  v1.31.3
```



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

- 証明書署名要求

1.15. 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                             4.18.0    True        False         False      19m
baremetal                                  4.18.0    True        False         False      37m
cloud-credential                           4.18.0    True        False         False      40m
cluster-autoscaler                         4.18.0    True        False         False      37m
config-operator                            4.18.0    True        False         False      38m
console                                    4.18.0    True        False         False      26m
csi-snapshot-controller                    4.18.0    True        False         False      37m
dns                                        4.18.0    True        False         False      37m
etcd                                       4.18.0    True        False         False      36m
image-registry                             4.18.0    True        False         False      31m
ingress                                    4.18.0    True        False         False      30m
insights                                   4.18.0    True        False         False      31m
kube-apiserver                             4.18.0    True        False         False      26m
kube-controller-manager                    4.18.0    True        False         False      36m
kube-scheduler                             4.18.0    True        False         False      36m
kube-storage-version-migrator              4.18.0    True        False         False      37m
machine-api                                4.18.0    True        False         False      29m
machine-approver                           4.18.0    True        False         False      37m
machine-config                             4.18.0    True        False         False      36m
marketplace                                4.18.0    True        False         False      37m
monitoring                                 4.18.0    True        False         False      29m
network                                    4.18.0    True        False         False      38m
```

```
node-tuning                                4.18.0    True        False         False      37m
openshift-apiserver                        4.18.0    True        False         False      32m
openshift-controller-manager               4.18.0    True        False         False      30m
openshift-samples                          4.18.0    True        False         False      32m
operator-lifecycle-manager                 4.18.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.18.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.18.0    True        False         False      32m
service-ca                                 4.18.0    True        False         False      38m
storage                                    4.18.0    True        False         False      37m
```

2. 利用不可の Operator を設定します。

1.15.1. 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。

1.15.2. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、Image Registry Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。これが完了したら、ストレージを設定する必要があります。

1.15.3. 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。

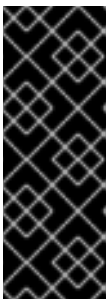
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

1.15.3.1. ベアメタルおよび他の手動インストール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。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- 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

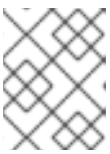
共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

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   4.18         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
```

1.15.3.2. 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。

1.15.3.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage
  namespace: openshift-image-registry
spec:
  accessModes:
  - ReadWriteOnce
  resources:
```

```
    requests:
      storage: 100Gi
```

ここでは、以下のようになります。

name

PersistentVolumeClaim オブジェクトを表す一意の名前を指定します。

namespace

PersistentVolumeClaim オブジェクトの **namespace** (**openshift-image-registry**) を指定します。

accessModes

永続ボリューム要求のアクセスモードを指定します。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

storage

永続ボリューム要求のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

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

カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

1.16. 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                             4.18.0    True        False         False      19m
baremetal                                  4.18.0    True        False         False      37m
cloud-credential                           4.18.0    True        False         False      40m
cluster-autoscaler                         4.18.0    True        False         False      37m
config-operator                            4.18.0    True        False         False      38m
console                                    4.18.0    True        False         False      26m
csi-snapshot-controller                    4.18.0    True        False         False      37m
dns                                        4.18.0    True        False         False      37m
etcd                                       4.18.0    True        False         False      36m
image-registry                             4.18.0    True        False         False      31m
ingress                                    4.18.0    True        False         False      30m
insights                                   4.18.0    True        False         False      31m
kube-apiserver                             4.18.0    True        False         False      26m
kube-controller-manager                    4.18.0    True        False         False      36m
kube-scheduler                             4.18.0    True        False         False      36m
kube-storage-version-migrator              4.18.0    True        False         False      37m
machine-api                                4.18.0    True        False         False      29m
machine-approver                           4.18.0    True        False         False      37m
machine-config                             4.18.0    True        False         False      36m
marketplace                                4.18.0    True        False         False      37m
monitoring                                 4.18.0    True        False         False      29m
network                                    4.18.0    True        False         False      38m
node-tuning                                4.18.0    True        False         False      37m
openshift-apiserver                        4.18.0    True        False         False      32m
openshift-controller-manager               4.18.0    True        False         False      30m
openshift-samples                          4.18.0    True        False         False      32m
operator-lifecycle-manager                 4.18.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.18.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.18.0    True        False         False      32m
service-ca                                 4.18.0    True        False         False      38m
storage                                    4.18.0    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。

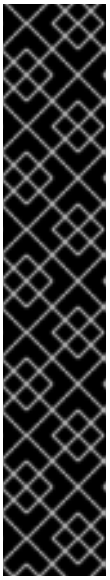
```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

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>
   ```

   - **<namespace>**: 前のコマンドの出力に示されているように、Pod 名と 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. FCP (Fibre Channel Protocol) を使用したインストールでは、マルチパスを有効にするために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** ドキュメントで、「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。

1.17. OPENSHIFT CONTAINER PLATFORM の TELEMETRY アクセス

OpenShift Container Platform 4.18 では、Telemetry サービスにもインターネットアクセスが必要です。Telemetry サービスは、クラスターの健全性と更新の成功に関するメトリクスを提供するためにデフォルトで実行されます。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 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 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 を参照してください。

1.18. 次のステップ

- 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。
- レジストリーをセットアップし、レジストリーストレージを設定 します。