OpenShift Container Platform 4.3

インストール

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

Last Updated: 2020-11-13

OpenShift Container Platform 4.3 インストール

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

法律上の通知

概要

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

目次

第1章 インストールログの収集

OpenShift Container Platform のインストールした場合のトラブルシューティングのために、ブートストラップおよびコントロールプレーン、またはマスター、マシンからログを収集できます。

1.1. 前提条件

- OpenShift Container Platform クラスターのインストールを試みたが、インストールに失敗している。
- SSH キーをインストールプログラムに指定しており、そのキーは実行中の **ssh-agent** プロセスにある。

1.2. 失敗したインストールのログの収集

SSH キーをインストールプログラムに指定している場合、失敗したインストールについてのデータを収集することができます。

注記

実行中のクラスターからログを収集する場合とは異なるコマンドを使用して失敗したインストールについてのログを収集します。実行中のクラスターからログを収集する必要がある場合は、**oc adm must-gather** コマンドを使用します。

前提条件

- OpenShift Container Platform のインストールがブートストラッププロセスの終了前に失敗している。ブートストラップノードは実行中であり、SSH でアクセスできる必要がある。
- **ssh-agent** プロセスはコンピューター上でアクティブであり、**ssh-agent** プロセスとインストールプログラムの両方に同じ SSH キーを提供している。
- 独自にプロビジョニングしたインフラストラクチャーにクラスターのインストールを試行した場合には、コントロールプレーン、またはマスター、マシンの完全修飾ドメイン名があること。

手順

1. ブートストラップおよびコントロールプレーンマシンからインストールログを収集するために必要なコマンドを生成します。
   - インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合は、以下のコマンドを実行します。

     ```
     $ ./openshift-install gather bootstrap --dir=<directory> 1
     ```

     1 **installation_directory** は、インストールプログラムが作成する OpenShift Container Platform 定義ファイルを保存しているディレクトリーです。

     インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムは、ホスト名または IP アドレスを指定しなくてもよいようにクラスターについての情報を保存します。

- 独自にプロビジョニングしたインフラストラクチャーを使用している場合は、以下のコマンドを実行します。

```
$ ./openshift-install gather bootstrap --dir=<directory> \ 1
  --bootstrap <bootstrap_address> \ 2
  --master <master_1_address> \ 3
  --master <master_2_address> \ 4
  --master <master_3_address>" 5
```

1. **installation_directory** は、インストールプログラムが作成する OpenShift Container Platform 定義ファイルを保存しているディレクトリーです。
2. **<bootstrap_address>** は、クラスターのブートストラップマシンの完全修飾ドメイン名または IP アドレスです。
3. 4. 5. クラスター内のそれぞれのコントロールプレーン、またはマスター、マシンについては、**<master_*_address>** をその完全修飾ドメイン名または IP アドレスに置き換えます。

注記

デフォルトクラスターには 3 つのコントロールプレーンマシンが含まれます。クラスターが使用する数にかかわらず、表示されるようにすべてのコントロールプレーンマシンを一覧表示します。

コマンド出力は以下の例のようになります。

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<directory>/log-bundle-<timestamp>.tar.gz"
```

インストールの失敗についての Red Hat サポートケースを作成する場合は、圧縮したログをケースに含めるようにしてください。

1.3. ホストへの SSH アクセスによるログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

前提条件

- ホストへの SSH アクセスがあること。

手順

1. 以下を実行し、**journalctl** コマンドを使用してブートストラップホストから **bootkube.service** サービスログを収集します。

```
$ journalctl -b -f -u bootkube.service
```

2. Podman ログを使用して、ブートストラップホストのコンテナーログを収集します。これは、ホストからすべてのコンテナーログを取得するためにループで表示されます。

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. または、以下を実行し、**tail** コマンドを使用してホストのコンテナーログを収集します。

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. 以下を実行し、**journalctl** コマンドを使用して **kubelet.service** および **crio.service** サービスログをマスターホストから収集します。

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. 以下を実行し、**tail** コマンドを使用してマスターおよびワーカーホストコンテナーログを収集します。

```
$ sudo tail -f /var/log/containers/*
```

1.4. ホストへの SSH アクセスを使用しないログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

ノードへの SSH アクセスがない場合は、システムジャーナルにアクセスし、ホストで生じていることを調査できます。

前提条件

- OpenShift Container Platform のインストールが完了している。
- API サービスが機能している。
- システム管理者権限がある。

手順

1. 以下を実行し、**/var/log** の下にある **journald** ユニットログにアクセスします。

```
$ oc adm node-logs --role=master -u kubelet
```

2. 以下を実行し、**/var/log** の下にあるホストファイルのパスにアクセスします。

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

第2章 FIPS 暗号のサポート

バージョン 4.3 以降、FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターをインストールすることができます。

クラスター内の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの場合、この変更は、ユーザーがクラスターのデプロイメント時に変更できるクラスターオプションを制御する **install-config.yaml** ファイルのオプションのステータスに基づいてマシンがデプロイされる際に適用されます。Red Hat Enterprise Linux マシンでは、ワーカーマシンとして使用する予定のマシンにオペレーティングシステムをインストールする場合に FIPS モードを有効にする必要があります。これらの設定方法により、クラスターが FIPS コンプライアンス監査の要件を満たすことを確認できます。初期システムの起動前は、FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号パッケージのみが有効になります。

FIPS はクラスターが使用するオペレーティングシステムの初回の起動前に有効にされている必要があり、クラスターをデプロイしてから FIPS を有効にすることはできません。

2.1. OPENSHIFT CONTAINER PLATFORM での FIPS 検証

OpenShift Container Platform は、それが使用するオペレーティングシステムのコンポーネント用に Red Hat Enterprise Linux (RHEL) および RHCOS 内の特定の FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用します。「RHEL7 core crypto components」を参照してください。たとえば、ユーザーが OpenShift Container Platform クラスターおよびコンテナーに対して SSH を実行する場合、それらの接続は適切に暗号化されます。

OpenShift Container Platform コンポーネントは Go で作成され、Red Hat の golang コンパイラーを使用してビルドされます。クラスターの FIPS モードを有効にすると、暗号署名を必要とするすべての OpenShift Container Platform コンポーネントは RHEL および RHCOS 暗号ライブラリーを呼び出します。

表2.1 OpenShift Container Platform 4.3 における FIPS モード属性および制限

属性	制限
RHEL 7 オペレーティングシステムの FIPS サポート。	FIPS 実装は、ハッシュ関数を計算し、そのハッシュに基づくキーを検証する単一の機能を提供しません。この制限については、今後の OpenShift Container Platform リリースで継続的に評価され、改善されます。
CRI-O ランタイムの FIPS サポート。	
OpenShift Container Platform サービスの FIPS サポート。	
RHEL 7 および RHCOS バイナリーおよびイメージから取得される FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号化モジュールおよびアルゴリズム。	
FIPS と互換性のある golang コンパイラーの使用。	TLS FIPS サポートは完全に実装されていませんが、今後の OpenShift Container Platform リリースで予定されています。

2.2. クラスターが使用するコンポーネントでの FIPS サポート

OpenShift Container Platform クラスター自体は FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用しますが、OpenShift Container Platform クラスターをサポートするシステムが暗号化の FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用していることを確認してください。

2.2.1. etcd

etcd に保存されるシークレットが FIPS 検証済み/進行中のモジュール (Modules in Process) の暗号を使用できるようにするには、ノードを FIPS モードで起動します。クラスターを FIPS モードでインストールした後に、 FIPS で承認される **aes cbc** 暗号アルゴリズムを使用して etcd データを暗号化できます。

2.2.2. ストレージ

ローカルストレージの場合は、RHEL が提供するディスク暗号化または RHEL が提供するディスク暗号化を使用する Container Native Storage を使用します。RHEL が提供するディスク暗号を使用するボリュームにすべてのデータを保存し、クラスター用に FIPS モードを有効にすることで、移動しないデータと移動するデータまたはネットワークデータは FIPS の検証済み/進行中のモジュール (Modules in Process) の暗号化によって保護されます。「ノードのカスタマイズ」で説明されているように、各ノードのルートファイルシステムを暗号化するようにクラスターを設定できます。

2.2.3. ランタイム

コンテナーに対して FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号モジュールを使用しているホストで実行されていることを認識させるには、CRI-O を使用してランタイムを管理します。CRI-O は FIPS-Mode をサポートします。このモードでは、コンテナーが FIPS モードで実行されていることを認識できるように設定されます。

2.3. FIPS モードでのクラスターのインストール

FIPS モードでクラスターをインストールするには、必要なインフラストラクチャーにカスタマイズされたクラスターをインストールする方法についての説明に従ってください。クラスターをデプロイする前に、**fips: true** を **install-config.yaml** ファイルに設定していることを確認します。

- Amazon Web Services
- Microsoft Azure
- ベアメタル
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

AES CBC 暗号化を etcd データストアに適用するには、クラスターのインストール後に etcd データの暗号化プロセスに従ってください。

RHEL ノードをクラスターに追加する場合は、初回の起動前に FIPS モードをマシン上で有効にしていることを確認してください。「RHEL コンピュートマシンの OpenShift Container Platform クラスターへの追加」および「FIPS モードの有効化 」を参照してください。

第3章 インストール設定

3.1. 各種プラットフォームのインストール方法

各種のプラットフォームで各種のインストールを実行できます。

注記

以下の表にあるように、すべてのプラットフォームですべてのインストールオプションが利用できる訳ではありません。

表3.1 インストーラーでプロビジョニングされるインフラストラクチャーのオプション

	AWS	Azure	GCP	OpenStack	ベアメタル	vSphere	IBM Z
デフォルト	X	X	X				
カスタム	X	X	X	X			
Network Operator	X	X	X				
プライベートクラスター	X	X	X				
既存の仮想プライベートネットワーク	X	X	X				

表3.2 ユーザーによってプロビジョニングされるインフラストラクチャーのオプション

	AWS	Azure	GCP	OpenStack	ベアメタル	vSphere	IBM Z
カスタム	X	X	X		X	X	
Network Operator					X	X	
ネットワークが制限されたインストール	X		X		X	X	

3.2. ノードのカスタマイズ

OpenShift Container Platform ノードへの直接の変更は推奨されませんが、必要とされる低レベルのセキュリティー、ネットワーク、またはパフォーマンス機能を実装することが必要になる場合があります。OpenShift Container Platform ノードへの直接の変更は、以下によって実行できます。

- **openshift-install** の実行時にクラスターを起動するためにマニフェストファイルに組み込まれる MachineConfig を作成します。
- Machine Config Operator を使用して実行中の OpenShift Container Platform ノードに渡される MachineConfig を作成します。

以下のセクションでは、この方法でノード上で設定する必要が生じる可能性のある機能について説明します。

3.2.1. day-1 カーネル引数の追加

多くの場合、カーネル引数を day-2 アクティビティーとして変更することが推奨されますが、初期クラスターのインストール時にすべてのマスターまたはワーカーノードにカーネル引数を追加することができます。以下は、クラスターのインストール時にカーネル引数を追加して、システムの初回起動前に有効にする必要が生じる可能性のある理由です。

- SELinux などの機能を無効にし、初回起動時にシステムに影響を与えないようにする必要がある場合。
- システムの起動前に、低レベルのネットワーク設定を実行する必要がある場合。

カーネル引数をマスターまたはワーカーノードに追加するには、MachineConfig オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

起動時に RHEL 8 カーネルに渡すことのできる引数の一覧については、Kernel.org カーネルパラメーターを参照してください。カーネル引数が OpenShift Container Platform の初回インストールを完了するために必要な場合は、この手順でカーネル引数のみを追加することが推奨されます。

手順

1. クラスターの Kubernetes マニフェストを生成します。

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

2. カーネル引数をワーカーまたはマスターノードに追加するかどうかを決定します。
3. **openshift** ディレクトリーでファイル（例: **99_openshift-machineconfig_master-kargs.yaml**）を作成し、カーネル設定を追加するために MachineConfig オブジェクトを定義します。以下の例では、**loglevel=7** カーネル引数をマスターノードに追加します。

```
$ cat << EOF > 99_openshift-machineconfig_master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99_openshift-machineconfig_master-kargs
spec:
```

```
  kernelArguments:
    - 'loglevel=7'
EOF
```

カーネル引数をワーカーノードに追加する場合は、**master** を **worker** に切り替えます。マスターおよびワーカーノードの両方に追加するために別々の YAML ファイルを作成します。

クラスターの作成を継続できます。

3.2.2. カーネルモジュールのノードへの追加

大半の一般的なハードウェアの場合、Linux カーネルには、コンピューターの起動時にそのハードウェアを使用するために必要となるデバイスドライバーモジュールが含まれます。ただし、一部のハードウェアの場合、Linux でモジュールを利用できません。したがって、各ホストコンピューターにこれらのモジュールを提供する方法を確保する必要があります。この手順では、OpenShift Container Platform クラスターのノードについてこれを実行する方法を説明します。

この手順に従ってカーネルモジュールを最初にデプロイする際、モジュールは現行のカーネルに対して利用可能になります。新規カーネルがインストールされると、kmods-via-containers ソフトウェアはモジュールを再ビルドし、デプロイしてそのモジュールの新規カーネルと互換性のあるバージョンが利用可能になるようにします。

この機能によって各ノードでモジュールが最新の状態に保てるようにするために、以下が実行されます。

- 新規カーネルがインストールされているかどうかを検出するために、システムの起動時に起動する各ノードに systemd サービスを追加します。
- 新規カーネルが検出されると、サービスはモジュールを再ビルドし、これをカーネルにインストールします。

この手順に必要なソフトウェアの詳細については、kmods-via-containers github サイトを参照してください。

以下の重要な点に留意してください。

- この手順はテクノロジープレビューです。
- ソフトウェアのツールおよびサンプルは公式の RPM 形式で利用できず、現時点ではこの手順に記載されている非公式の **github.com** サイトからしか取得できません。
- この手順で追加する必要がある可能性のあるサードパーティーのカーネルモジュールについては Red Hat はサポートしません。
- この手順では、カーネルモジュールのビルドに必要なソフトウェアは RHEL 8 コンテナーにデプロイされます。モジュールは、ノードが新規カーネルを取得する際に各ノードで自動的に再ビルドされることに注意してください。このため、各ノードには、モジュールの再ビルドに必要なカーネルと関連パッケージを含む **yum** リポジトリーへのアクセスが必要です。このコンテンツは、有効な RHEL サブスクリプションを使用して効果的に利用できます。

3.2.2.1. カーネルモジュールコンテナーのビルドおよびテスト

カーネルモジュールを OpenShift Container Platform クラスターにデプロイする前に、プロセスを別の RHEL システムでテストできます。カーネルモジュールのソースコード、KVC フレームワーク、および kmod-via-containers ソフトウェアを収集します。次にモジュールをビルドし、テストします。RHEL 8 システムでこれを行うには、以下を実行します。

手順

1. RHEL 8 システムを取得して、これを登録し、サブスクライブします。

```
# subscription-manager register
Username: yourname
Password: ***************
# subscription-manager attach --auto
```

2. ソフトウェアとコンテナーのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

3. kmod-via-containers リポジトリーのクローンを作成します。

```
$ mkdir kmods; cd kmods
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

4. RHEL 8 ビルドホストに KVC フレームワークインスタンスをインストールし、モジュールをテストします。これにより kmods-via-container systemd サービスが追加され、ロードされます。

```
$ cd kmods-via-containers/
$ sudo make install
$ sudo systemctl daemon-reload
```

5. カーネルモジュールのソースコードを取得します。ソースコードは、制御下になく、他から提供されるサードパーティーモジュールをビルドするために使用される可能性があります。システムに対してクローン作成できる以下の **kvc-simple-kmod** サンプルのコンテンツと同様のコンテンツが必要になります。

```
$ cd ..
$ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

6. この例では、設定ファイル **simple-kmod.conf** を編集し、Dockerfile の名前を **Dockerfile.rhel** に変更し、ファイルが以下のように表示されるようにします。

```
$ cd kvc-simple-kmod
$ cat simple-kmod.conf

KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-simple-kmod.git"
KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

7. この例ではカーネルモジュール **simple-kmod** の **kmods-via-containers@.service** のインスタンスを作成し、これを有効にします。

```
$ sudo make install
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

8. systemd サービスを有効にしてから起動し、ステータスを確認します。

```
-
```

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service
$ sudo systemctl start kmods-via-containers@simple-kmod.service
$ sudo systemctl status kmods-via-containers@simple-kmod.service
● kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
   Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
         enabled; vendor preset: disabled)
   Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

9. カーネルモジュールがロードされていることを確認するには、**lsmod** コマンドを使用してモジュールを一覧表示します。

```
$ lsmod | grep simple_
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

10. simple-kmod のサンプルでは、それが機能するかどうかをテストする他のいくつかの方法も使用できます。**dmesg** を使ってカーネルリングバッファーで「Hello world」メッセージを探します。

```
$ dmesg | grep 'Hello world'
[ 6420.761332] Hello world from simple_kmod.
```

/**proc** で **simple-procfs-kmod** の値を確認します。

```
$ sudo cat /proc/simple-procfs-kmod
simple-procfs-kmod number = 0
```

spkut コマンドを実行して、モジュールの詳細情報を取得します。

```
$ sudo spkut 44
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
+ podman run -i --rm --privileged
   simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

その後は、システムの起動時に、このサービスは新規カーネルが実行中であるかどうかをチェックします。新規カーネルがある場合は、サービスは新規バージョンのカーネルモジュールをビルドし、これをロードします。モジュールがすでにビルドされている場合は、これをロードします。

3.2.2.2. カーネルモジュールの OpenShift Container Platform へのプロビジョニング

OpenShift Container Platform クラスターの初回起動時にカーネルモジュールを有効にする必要があるかどうかに応じて、以下のいずれかの方法でデプロイするようにカーネルモジュールを設定できます。

- **クラスターインストール時のカーネルモジュールのプロビジョニング (day-1)**: コンテンツを MachineConfig として作成し、これをマニフェストファイルのセットと共に組み、**openshift-install** に提供できます。

- **Machine Config Operator によるカーネルモジュールのプロビジョニング (day-2)**: カーネルモジュールを追加する際にクラスターが稼働するまで待機できる場合、Machine Config Operator (MCO) を使用してカーネルモジュールソフトウェアをデプロイできます。

いずれの場合も、各ノードは、新規カーネルの検出時にカーネルパッケージと関連ソフトウェアパッケージを取得できる必要があります。該当するコンテンツを取得できるように各ノードをセットアップする方法はいくつかあります。

- 各ノードに RHEL エンタイトルメントを提供します。
- **/etc/pki/entitlement** ディレクトリーから、既存 RHEL ホストの RHEL エンタイトルメントを取得し、それらを Ignition 設定の作成時に提供する他のファイルと同じ場所にコピーします。
- Dockerfile 内で、カーネルおよびその他のパッケージを含む **yum** リポジトリーへのポインターを追加します。これには、新たにインストールされたカーネルと一致させる必要があるため、新規のカーネルパッケージが含まれている必要があります。

3.2.2.2.1. MachineConfig でのカーネルモジュールのプロビジョニング

MachineConfig でカーネルモジュールソフトウェアをパッケージ化することで、そのソフトウェアをインストール時に、または Machine Config Operator を使用してワーカーまたはマスターノードに配信できます。

まずに、使用するベース Ignition 設定を作成します。インストール時に、Ignition 設定にはクラスターの **core** ユーザーの **authorized_keys** ファイルに追加する ssh パブリックキーが含まれます。後で MCO を使用して MachineConfig を追加する場合、ssh パブリックキーは不要になります。どちらのタイプでも、サンプルの simple-kmod サービスは **kmods-via-containers@simple-kmod.service** を必要とする systemd ユニットファイルを作成します。

注記

systemd ユニットはアップストリームのバグに対する回避策であり、**kmods-via-containers@simple-kmod.service** が起動時に開始するようにします。

1. RHEL 8 システムを取得して、これを登録し、サブスクライブします。

```
# subscription-manager register
Username: yourname
Password: ***************
# subscription-manager attach --auto
```

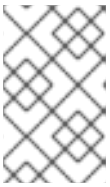
2. ソフトウェアのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

3. systemd ユニットファイルを作成する Ignition 設定ファイルを作成します。

```
$ mkdir kmods; cd kmods
$ cat <<EOF > ./baseconfig.ign
{
  "ignition": { "version": "2.2.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "groups": ["sudo"],
        "sshAuthorizedKeys": [
          "ssh-rsa AAAA"
```

```
        ]
       }
      ]
    },
    "systemd": {
      "units": [{
        "name": "require-kvc-simple-kmod.service",
        "enabled": true,
        "contents": "[Unit]\nRequires=kmods-via-containers@simple-
kmod.service\n[Service]\nType=oneshot\nExecStart=/usr/bin/true\n\n[Install]\nWantedBy=multi-
user.target"
      }]
    }
}
EOF
```



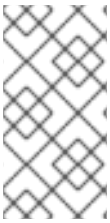
注記

openshift-install の実行時に、パブリック SSH キーを使用する **baseconfig.ign** ファイルに追加する必要があります。MCO を使用して MachineConfig を作成する場合、パブリック SSH キーは必要ありません。

4. 以下のようなベース MCO YAML スニペットを作成します。

```
$ cat <<EOF > mc-base.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 10-kvc-simple-kmod
spec:
  config:
EOF
```

+



注記

mc-base.yaml は、**worker** ノードにカーネルモジュールをデプロイするように設定されます。マスターノードでデプロイするには、ロールを **worker** から **master**に変更します。どちらの方法でも、デプロイメントの種類ごとに異なるファイル名を使用して手順全体を繰り返すことができます。

1. **kmods-via-containers** ソフトウェアを取得します。

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
$ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

2. モジュールソフトウェアを取得します。この例では、**kvc-simple-kmod** が使用されます。

3. fakeroot ディレクトリーを作成し、先にクローン作成したリポジトリーを使用して Ignition で配信するファイルを使用してこれを設定します。

```
$ FAKEROOT=$(mktemp -d)
$ cd kmods-via-containers
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
$ cd ../kvc-simple-kmod
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

4. **filetranspiler** というツールおよび依存するソフトウェアを取得します。

```
$ cd ..
$ sudo yum install -y python3
git clone https://github.com/ashcrow/filetranspiler.git
```

5. 最終的な MachineConfig YAML（**mc.yaml**）を生成し、これに配信するファイルと共にベース Ignition 設定、ベース MachineConfig および fakeroot ディレクトリーを含めます。

```
$ ./filetranspiler/filetranspile -i ./baseconfig.ign \
    -f ${FAKEROOT} --format=yaml --dereference-symlinks \
    | sed 's/^/     /' | (cat mc-base.yaml -) > 99_simple-kmod.yaml
```

6. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、そのファイルを **openshift** ディレクトリーに追加します。クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

```
$ oc create -f 99_simple-kmod.yaml
```

ノードは **kmods-via-containers@simple-kmod.service** サービスを起動し、カーネルモジュールがロードされます。

7. カーネルモジュールがロードされていることを確認するには、ノードにログインすることができます (**oc debug node/<openshift-node>** を使用してから **chroot /host**を使用します)。モジュールを一覧表示するには、**lsmod** コマンドを使用します。

```
$ lsmod | grep simple_
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

3.2.3. インストール時のディスクの暗号化

OpenShift Container Platform のインストール時に、すべてのマスターおよびノードノードでディスクの暗号化を有効にできます。この機能には以下の特徴があります。

- インストーラーでプロビジョニングされるインフラストラクチャーおよびユーザーによってプロビジョニングされるインフラストラクチャーのデプロイメントで利用可能である。
- Red Hat Enterprise Linux CoreOS (RHCOS) システムのみでサポートされる。
- マニフェストのインストールフェーズでディスク暗号化が設定される。これにより、初回起動時からディスクに書き込まれたすべてのデータが暗号化されます。
- ルートファイルシステムのみでデータを暗号化する (**/dev/mapper/coreos-luks-root** は / マウントポイントを表す)。
- パスフレーズを提供するのにユーザーの介入を必要としない。

- AES-256-CBC 暗号化を使用する。
- クラスターで FIPS をサポートするために有効にされている必要がある。

サポートされている暗号化モードとして、以下の 2 つのモードを使用できます。

- TPM v2: これは優先されるモードです。TPM v2 はパスフレーズを安全な暗号プロセッサーに保存します。TPM v2 ディスク暗号化を実装するには、以下で説明されているように Ignition 設定ファイルを作成します。
- Tang: Tang を使用してクラスターを暗号化するには、Tang サーバーを使用する必要があります。Clevis はクライアント側に復号化を実装します。Tang 暗号化モードは、ベアメタルのインストールの場合にのみサポートされます。

クラスター内のノードのディスク暗号化を有効にするには、以下の 2 つの手順の内のいずれかに従います。

3.2.3.1. TPM v2 ディスク暗号化の有効化

以下の手順を使用して、OpenShift Container Platform のデプロイメント時に TPM v2 モードのディスク暗号化を有効にします。

手順

1. TPM v2 暗号化を各ノードの BIOS で有効にする必要があるかどうかを確認します。これは、ほとんどの Dell システムで必要になります。コンピューターのマニュアルを確認してください。
2. 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>
```

3. **openshift** ディレクトリーで、マスターおよび/またはワーカーファイルを作成し、それらのノードのディスクを暗号化します。以下は、それらの 2 つのファイルの例になります。

```
$ cat << EOF > ./99_openshift-worker-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tpm
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

```
$ cat << EOF > ./99_openshift-master-tpmv2-encryption.yaml
```

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tpm
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

4. YAML ファイルのバックアップコピーを作成します。ファイルはクラスターの作成時に削除されるため、これを実行する必要があります。

5. 残りの OpenShift Container Platform のデプロイメントを継続します。

3.2.3.2. Tang ディスク暗号化の有効化

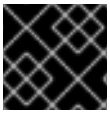
以下の手順を使用して、OpenShift Container Platform のデプロイメント時に Tang モードのディスク暗号化を有効にします。

手順

1. 暗号化の設定を構成し、**openshift-install** を実行してクラスターをインストールし、**oc** を使用してクラスターを操作するために Red Hat Enterprise Linux サーバーにアクセスします。

2. 既存の Tang サーバーを設定するか、またはこれにアクセスします。手順については、「NBDE (Network-Bound Disk Encryption)」を参照してください。タグの表示についての詳細は、Securing Automated Decryption New Cryptography and Techniques を参照してください。

3. クラスターについて Red Hat Enterprise Linux CoreOS (RHCOS) インストールを実行する際にネットワークを設定するためにカーネル引数を追加します。たとえば、DHCP ネットワークを設定するには、**ip=dhcp** を特定するか、またはカーネルコマンドラインにパラメーターを追加する際に静的ネットワークを設定します。DHCP と静的ネットワークの両方の場合、**rd.neednet=1** カーネル引数も指定する必要があります。



重要

このステップを省略すると、2 番目の起動に失敗します。

4. サムプリントを生成します。(まだインストールされていない場合は) clevis パッケージをインストールし、Tang サーバーからサムプリントを生成します。**url** の値を Tang サーバーの URL に置き換えます。

```
$ sudo yum install clevis -y
$ echo nifty random wordwords \
    | clevis-encrypt-tang \
```

```
      '{"url":"https://tang.example.org"}'

The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9

Do you wish to trust these keys? [ynYN] y
eyJhbmc3SlRyMXpPenc3ajhEQ01tZVJiTi1oM...
```

5. Base64 でエンコードされたファイルを作成します。値を Tang サーバーの URL (**url**) と生成したサムプリント (**thp**) で置き換えます。

```
$ (cat <<EOM
{
 "url": "https://tang.example.com",
 "thp": "PLjNyRdGw03zlRoGjQYMahSZGu9"
}
EOM
) | base64 -w0

ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRocCI6ICJaUk1leTFjR3cw
N3psVExxHYlhuUWFoUzBHdTAiCn0K
```

6. TPM2 の例の「source」を、ワーカーおよび/またはマスターノードのサンプルのいずれか、またはそれら両方の Base64 でエンコードされたファイルに置き換えます。

重要

rd.neednet=1 カーネル引数を追加する必要があります。

```
$ cat << EOF > ./99-openshift-worker-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tang
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRoc
CI6ICJaUk1leTFjR3cwN3psVExxHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
  kernelArguments:
    - rd.neednet=1 <.>
EOF
```

必須。

```
$ cat << EOF > ./99_openshift-master-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tang
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRoc
CI6ICJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
  kernelArguments:
    - rd.neednet=1 <.>
EOF
```

必須。

7. 残りの OpenShift Container Platform のデプロイメントを継続します。

3.2.4. chrony タイムサービスの設定

chrony タイムサービス (chronyd) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツを MachineConfig としてノードに渡して設定できます。

手順

1. **chrony.conf** ファイルのコンテンツを作成し、これを base64 でエンコードします。以下は例になります。

```
$ cat << EOF | base64
    server clock.redhat.com iburst
    driftfile /var/lib/chrony/drift
    makestep 1.0 3
    rtcsync
    logdir /var/log/chrony
    EOF

ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXIvbGli
```

```
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2
RpciAv
dmFyL2xvZy9jaHJvbnkK
```

2. MachineConfig ファイルを作成します。base64 文字列を独自に作成した文字列に置き換えます。この例では、ファイルを **master** ノードに追加します。これを **worker** に切り替えたり、**worker** ロールの追加の MachineConfig を作成したりできます。

```
$ cat << EOF > ./99_masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,c2VydmVyIGNsb2NrLnJlZGhhdC5jb20gaWJ1cnN0CmRyaWZ0ZmlsZSAvdmFyL2xp
Yi9jaHJvbnkvZHJpZnQKbWFrZXN0ZXAgMS4wIDMKcnRjc3luYwpsb2dkaXIgL3Zhci9sb2cvY2h
yb255Cg==
        verification: {}
      filesystem: root
      mode: 420
      path: /etc/chrony.conf
  osImageURL: ""
EOF
```

3. 設定ファイルのバックアップコピーを作成します。

4. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、そのファイルを **openshift** ディレクトリーに追加してから、クラスターの作成を継続します。

5. クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

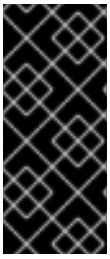
```
$ oc apply -f ./masters-chrony-configuration.yaml
```

3.2.5. 追加リソース

FIPS サポートについての詳細は、「FIPS 暗号のサポート」を参照してください。

3.3. ネットワークが制限された環境でのインストール用のミラーレジストリーの作成

ネットワークが制限された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールする前に、必要なコンテナーイメージをその環境にミラーリングする必要があります。ネットワークが制限された環境でのインストールは、インストーラーでプロビジョニングされるインフラストラクチャーではなく、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でのみサポートされます。この手順を無制限のネットワークで使用して、クラスターが外部コンテンツにちて組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。



重要

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

3.3.1. ミラーレジストリーについて

OpenShift Container Platform インストールおよびミラーレジストリーに対する後続の製品の更新に必要なイメージをミラーリングできます。これらのアクションは同じプロセスを使用します。コンテンツの説明が含まれるリリースイメージ、およびこれが参照するイメージがすべてミラーリングされます。さらに、Operator カタログソースイメージおよびこれが参照するイメージは、使用する Operator ごとにミラーリングする必要があります。コンテンツをミラーリングした後に、各クラスターをミラーレジストリーからこのコンテンツを取得するように設定します。

ミラーレジストリーには、最新のコンテナーイメージ API (**schema2** と呼ばれる) をサポートするすべてのコンテナーレジストリーを使用できます。すべての主要なクラウドプロバイダーレジストリー、および Red Hat Quay、Artifactory、およびオープンソースの Docker ディストリビューションレジストリー には、必要なサポートがあります。これらのレジストリーの 1 つを使用すると、OpenShift Container Platform で非接続環境で各イメージの整合性を検証できるようになります。

ミラーレジストリーは、プロビジョニングするクラスター内のすべてのマシンから到達できる必要があります。レジストリーに到達できない場合、インストール、更新、またはワークロードの再配置などの通常の操作が失敗する可能性があります。そのため、ミラーレジストリーは可用性の高い方法で実行し、ミラーレジストリーは少なくとも OpenShift Container Platform クラスターの実稼働環境の可用性の条件に一致している必要があります。

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

3.3.2. ミラーホストの準備

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

3.3.2.1. バイナリーのダウンロードによる CLI のインストール

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

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.3 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.3.2.1.1. Linux への 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 OpenShift Cluster Manager サイトの「Infrastructure Provider」ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

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

5. **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
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.3.2.1.2. Windows での 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 OpenShift Cluster Manager サイトの「Infrastructure Provider」ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

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

3.3.2.1.3. macOS への 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 OpenShift Cluster Manager サイトの「Infrastructure Provider」ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
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
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.3.3. イメージのミラーリングを可能にする認証情報の設定

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

警告

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

警告

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

重要

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

前提条件

- ネットワークが制限された環境で使用するミラーレジストリーを設定していること。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。
- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. Red Hat OpenShift Cluster Manager サイトの「Pull Secret」ページから **registry.redhat.io** プルシークレットをダウンロードし、これを **.json** ファイルに保存します。
2. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

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

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

3. JSON 形式でプルシークレットのコピーを作成します。

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

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

ファイルの内容は以下の例のようになります。

```
{
```

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

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

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

1 **<mirror_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000**

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

ファイルは以下の例のようになります。

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

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

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

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

1 **<mirror_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000**

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

ファイルは以下の例のようになります。

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

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

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

前提条件

- ミラーホストがインターネットにアクセスできる。
- ネットワークが制限された環境で使用するミラーレジストリーを設定し、設定した証明書および認証情報にアクセスできる。
- Red Hat OpenShift Cluster Manager のサイトの「Pull Secret」ページからプルシークレットをダウンロードしており、ミラーリポジトリーに認証を組み込むようにこれを変更している。

手順

ミラーホストで以下の手順を実行します。

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

```
$ export OCP_RELEASE=<release_version> 1
$ export LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>' 2
$ export LOCAL_REPOSITORY='<local_repository_name>' 3
$ export PRODUCT_REPO='openshift-release-dev' 4
$ export LOCAL_SECRET_JSON='<path_to_pull_secret>' 5
$ export RELEASE_NAME="ocp-release" 6
$ export ARCHITECTURE=<server_architecture> 7
$ REMOVABLE_MEDIA_PATH=<path> 8
```

1. **<release_version>** について、ご使用のアーキテクチャー用にインストールする OpenShift Container Platform のバージョンに対応するタグを指定します (例: **4.3.0**)。
2. **<local_registry_host_name>** については、ミラーレジストリーのレジストリードメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。
3. **<local_repository_name>** については、**ocp4/openshift4** などのレジストリーに作成するリポジトリーの名前を指定します。
4. ミラーリングするリポジトリー。実稼働環境のリリースの場合には、**openshift-release-dev**を指定する必要があります。
5. **<path_to_pull_secret>**については、作成したミラーレジストリーのプルシークレットおよびファイル名の絶対パスを指定します。
6. リリースミラー。実稼働環境のリリースについては、**ocp-release**を指定する必要があります。
7. **server_architecture** について、サーバーのアーキテクチャー (例: **x86_64**) を指定します。
8. **<path>** について、ミラーリングされたイメージをホストするためのディレクトリーへのパスを指定します。

3. バージョンイメージを内部コンテナーレジストリーにミラーリングします。

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

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

  ii. ミラーリングするイメージおよび設定マニフェストを確認します。

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

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

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

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

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

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

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

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

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

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

  ii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に

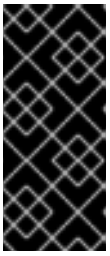
imageContentSources セクションを **install-config.yaml** ファイルに追加する必要があります。

注記

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

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

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



重要

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

インターネット接続のあるマシンで、このステップを実行する必要があります。

3.3.5. サポートデータを収集するためのクラスターの準備

ネットワークが制限された環境を使用するクラスターは、Red Hat サポート用のデバッグデータを収集するために、デフォルトの must-gather イメージをインポートする必要があります。must-gather イメージはデフォルトでインポートされず、ネットワークが制限された環境のクラスターは、リモートリポジトリーから最新のイメージをプルするためにインターネットにアクセスできません。

手順

1. インストールペイロードからデフォルトの must-gather イメージをインポートします。

```
$ oc import-image is/must-gather -n openshift
```

3.3.6. 代替のレジストリーまたはミラーリングされたレジストリーでの Samples Operator イメージストリームの使用

Samples Operator によって管理される OpenShift namespace のほとんどのイメージストリームは、Red Hat レジストリーの registry.redhat.io にあるイメージを参照します。ミラーリングはこれらのイメージストリームには適用されません。

重要

jenkins、**jenkins-agent-maven**、および **jenkins-agent-nodejs** イメージストリームは、インストールペイロードからのもので、Samples Operator によって管理されるため、これらのイメージストリームには追加のミラーリングの手順は必要ありません。

Sample Operator 設定ファイルの **samplesRegistry** フィールドの registry.redhat.io への設定は、これはすでに Jenkins イメージおよびイメージストリーム以外のすべての registry.redhat.io に送信されているため不要になります。また、Jenkins イメージストリームのインストールペイロードも破損します。

Samples Operator は Jenkins イメージストリームについて以下のレジストリーの使用を防ぎます。

- docker.io
- registry.redhat.io
- registry.access.redhat.com
- quay.io

注記

cli、**installer**、**must-gather**、および **tests** イメージストリームはインストールペイロードの一部ですが、Samples Operator によって管理されません。これらについては、この手順で扱いません。

前提条件

- **cluster-admin** ロールを持つユーザーとしてのクラスターへのアクセス。
- ミラーレジストリーのプルシークレットの作成。

手順

1. ミラーリングする特定のイメージストリームのイメージにアクセスします。

```
$ oc get is <imagestream> -n openshift -o json | jq .spec.tags[].from.name | grep
registry.redhat.io
```

2. ネットワークが制限された環境で必要とするイメージストリームに関連付けられた registry.redhat.io のイメージを定義されたミラーのいずれかにミラーリングします。

```
$ oc image mirror registry.redhat.io/rhscl/ruby-25-rhel7:latest ${MIRROR_ADDR}/rhscl/ruby-
25-rhel7:latest
```

3. クラスターのイメージ設定オブジェクトに、ミラーに必要な信頼される CA を追加します。

```
$ oc create configmap registry-config --from-
file=${MIRROR_ADDR_HOSTNAME}..5000=$path/ca.crt -n openshift-config
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"registry-config"}}}' --type=merge
```

4. Samples Operator 設定オブジェクトの **samplesRegistry** フィールドを、ミラー設定で定義されたミラーの場所の **hostname** の部分を含むように更新します。

```
$ oc edit configs.samples.operator.openshift.io -n openshift-cluster-samples-operator
```

注記

これは、イメージストリームのインポートプロセスでミラーまたは検索メカニズムが使用されないので必要になります。

5. Samples Operator 設定オブジェクトの **skippedImagestreams** フィールドにミラーリングされないイメージストリームを追加します。または、サンプルイメージストリームのいずれもサポートする必要がない場合は、Samples Operator を Samples Operator 設定オブジェクトの **Removed** に設定します。

注記

省略されないミラーリングされないイメージがあるか、または Samples Operator が **Removed** に変更されない場合、 Samples Operator はイメージストリームのインポートが失敗し始めてから 2 時間後に **Degraded** ステータスを報告します。

OpenShift namespace のテンプレートの多くはイメージストリームを参照します。そのため、**Removed** を使用してイメージストリームとテンプレートの両方を除去すると、イメージストリームのいずれかが欠落しているためにテンプレートが正常に機能しない場合にテンプレートの使用を試行する可能性がなくなります。

3.3.7. 次のステップ

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

3.4. 利用可能なクラスターのカスタマイズ

OpenShift Container Platform クラスターのデプロイ後は、大半のクラスター設定およびカスタマイズが終了していることになります。数多くの**設定リソース**が利用可能です。

イメージレジストリー、ネットワーク設定、イメージビルドの動作およびアイデンティティープロバイダーなどのクラスターの主要な機能を設定するために設定リソースを変更します。

これらのリソースを使用して制御する設定の現在の記述については、 **oc explain** コマンドを使用します (例: **oc explain builds --api-version=config.openshift.io/v1**)。

3.4.1. クラスター設定リソース

すべてのクラスター設定リソースはグローバルにスコープが設定され (namespace は設定されない)、**cluster**という名前が付けられます。

リソース名	説明
apiserver.config.openshift.io	証明書および認証局などの api-server 設定を提供します。

リソース名	説明
authentication.config.openshift.io	クラスターのアイデンティティープロバイダーおよび認証設定を制御します。
build.config.openshift.io	クラスターのすべてのビルドについてのデフォルトおよび有効にされている設定を制御します。
console.config.openshift.io	ログアウト動作を含む Web コンソールインターフェースの動作を設定します。
featuregate.config.openshift.io	FeatureGates を有効にして、テクノロジープレビュー機能を使用できるようにします。
image.config.openshift.io	特定の イメージレジストリー が処理される方法を設定します (allowed、disallowed、insecure、CA details)。
ingress.config.openshift.io	ルートのデフォルトドメインなどのルーティングに関連する設定の詳細。
oauth.config.openshift.io	内部 OAuth サーバーフローに関連するアイデンティティープロバイダーおよび他の動作を設定します。
project.config.openshift.io	プロジェクトテンプレートを含む、プロジェクトの作成方法 を設定します。
proxy.config.openshift.io	外部ネットワークアクセスを必要とするコンポーネントで使用されるプロキシーを定義します。注: すべてのコンポーネントがこの値を使用する訳ではありません。
scheduler.config.openshift.io	ポリシーやデフォルトノードセレクターなどのスケジューラーの動作を設定します。

3.4.2. Operator 設定リソース

これらの設定リソースは、**cluster**という名前のクラススコープのインスタンスです。これは、特定の Operator によって所有される特定コンポーネントの動作を制御します。

リソース名	説明
console.operator.openshift.io	ブランドのカスタマイズなどのコンソールの外観の制御
config.imageregistry.operator.openshift.io	パブリックルーティング、プロキシー設定、リソース設定、レプリカ数およびストレージタイプなどの 内部イメージレジストリー設定を設定します。
config.samples.operator.openshift.io	Samples Operator を設定して、クラスターにインストールされるイメージストリームとテンプレートのサンプルを制御します。

3.4.3. 追加の設定リソース

これらの設定リソースは、特定コンポーネントの単一インスタンスを表します。場合によっては、リソースの複数のインスタンスを作成して、複数のインスタンスを要求できます。他の場合には、Operator は特定の namespace の特定のリソースインスタンス名のみを使用できます。追加のリソースインスタンスの作成方法や作成するタイミングについての詳細は、コンポーネント固有のドキュメントを参照してください。

リソース名	インスタンス名	Namespace	説明
alertmanager.monitoring.coreos.com	main	openshift-monitoring	alertmanager デプロイメントパラメーターを制御します。
ingresscontroller.operator.openshift.io	default	openshift-ingress-operator	ドメイン、レプリカ数、証明書、およびコントローラーの配置などの Ingress Operator 動作を設定します。

3.4.4. 情報リソース

これらのリソースを使用して、クラスターについての情報を取得します。これらのリソースは直接編集しないでください。

リソース名	インスタンス名	説明
clusterversion.config.openshift.io	version	OpenShift Container Platform 4.3 では、実稼働クラスターの ClusterVersion リソースをカスタマイズすることはできません。その代わりとして、 クラスターの更新プロセスを実行します。
dns.config.openshift.io	cluster	クラスターの DNS 設定を変更することはできません。DNS Operator ステータスを表示できます。
infrastructure.config.openshift.io	cluster	クラスターはそのクラウドプロバイダーとの対話を可能にする設定の詳細。
network.config.openshift.io	cluster	インストール後にクラスターのネットワークを変更することはできません。ネットワークをカスタマイズするには、インストール時にネットワークをカスタマイズするプロセスを実行します。

3.5. ファイアウォールの設定

ファイアウォールを使用する場合、OpenShift Container Platform が機能するために必要なサイトにアクセスできるように設定する必要があります。一部のサイトにはアクセスを常に付与し、クラスターをホストするために Red Hat Insights、Telemetry サービス、クラウドを使用したり、特定のビルドストラテジーをホストする場合に追加のアクセスを付与する必要があります。

3.5.1. OpenShift Container Platform のファイアウォールの設定

OpenShift Container Platform をインストールする前に、ファイアウォールを、OpenShift Container Platform が必要とするサイトへのアクセスを付与するように設定する必要があります。

コントローラーノードのみで実行されるサービスとワーカーノードで実行されるサービスの設定に関する特別な考慮事項はありません。

手順

1. 以下のレジストリー URL を許可リストに指定します。

URL	関数
registry.redhat.io	コアコンテナーイメージを指定します。
quay.io	コアコンテナーイメージを指定します。
sso.redhat.com	**https://cloud.redhat.com/openshift** サイトでは、**sso.redhat.com** からの認証を使用します。
openshift.org	Red Hat Enterprise Linux CoreOS (RHCOS) イメージを提供します。

2. ビルドに必要な言語またはフレームワークのリソースを提供するサイトを許可リストに指定します。

3. Telemetry を無効にしていない場合は、以下の URL へのアクセスを許可して Red Hat Insights にアクセスできるようにする必要があります。

URL	関数
cert-api.access.redhat.com	Telemetry で必須
api.access.redhat.com	Telemetry で必須
infogw.api.openshift.com	Telemetry で必須
https://cloud.redhat.com/api/ingress	Telemetry および **insights-operator** で必須

4. Amazon Web Services (AWS) 、Microsoft Azure、または Google Cloud Platform (GCP) を使用してクラスターをホストする場合、クラウドプロバイダー API およびそのクラウドの DNS を提供する URL へのアクセスを付与する必要があります。

クラウド	URL	関数

クラウド	URL	関数
AWS	***.amazonaws.com**	AWS サービスおよびリソースへのアクセスに必要です。AWS ドキュメントの「AWS Service Endpoints」を参照し、使用するリージョンを許可するエンドポイントを判別します。
GCP	***.googleapis.com**	GCP サービスおよびリソースへのアクセスに必要です。GCP ドキュメントの「Cloud Endpoints」を参照し、API を許可するエンドポイントを判別します。
	accounts.google.com	GCP アカウントへのアクセスに必要です。
Azure	**management.azure.com**	Azure サービスおよびリソースへのアクセスに必要です。Azure ドキュメントで「Azure REST API Reference」を参照し、API を許可するエンドポイントを判別します。

5. 以下の URL を許可リストに指定します。

URL	関数
mirror.openshift.com	ミラーリングされたインストールのコンテンツおよびイメージへのアクセスに必要。Cluster Version Operator には単一の機能ソースのみが必要ですが、このサイトはリリースイメージ署名のソースでもあります。
storage.googleapis.com/openshift-release	リリースイメージ署名のソース (ただし、Cluster Version Operator には単一の機能ソースのみが必要)。
***.apps.<cluster_name>.<base_domain>**	Ingress ワイルドカードをインストール時に設定しない限り、デフォルトのクラスタールートへのアクセスに必要。
quay-registry.s3.amazonaws.com	AWS で Quay イメージコンテンツにアクセスするために必要。
api.openshift.com	クラスターに更新が利用可能かどうかを確認するために必要。
art-rhcos-ci.s3.amazonaws.com	Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードするために必要。
api.openshift.com	クラスタートークンに必須

URL	関数
cloud.redhat.com/openshift	クラスタートークンに必須

Operator にはヘルスチェックを実行するためのルートアクセスが必要です。具体的には、認証および Web コンソール Operator は 2 つのルートに接続し、ルートが機能することを確認します。クラスター管理者として操作を実行しており、***.apps.<cluster_name>.<base_domain>** を許可しない場合は、これらのルートを許可します。

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**、またはフィールドが空でない場合に **consoles.operator/cluster** オブジェクトの **spec.route.hostname** フィールドに指定されるホスト名。

3.6. プライベートクラスターの設定

OpenShift Container Platform バージョン 4.3 クラスターのインストール後に、そのコアコンポーネントの一部を private に設定できます。

重要

この変更は、クラウドプロバイダーに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クラスターにのみ設定できます。

3.6.1. プライベートクラスター

デフォルトで、OpenShift Container Platform は一般にアクセス可能な DNS およびエンドポイントを使用してプロビジョニングされます。クラスターのデプロイ後に DNS、Ingress コントローラー、および API サーバーを private に設定できます。

DNS
OpenShift Container Platform をインストーラーでプロビジョニングされるインフラストラクチャーにインストールする場合、インストールプログラムは既存のパブリックゾーンにレコードを作成し、可能な場合はクラスター独自の DNS 解決用のプライベートゾーンを作成します。パブリックゾーンおよびプライベートゾーンの両方で、インストールプログラムまたはクラスターが Ingress の ***.apps**、および API サーバーの **api** の DNS エントリーを作成します。

***.apps** レコードはパブリックゾーンとプライベートゾーンのどちらでも同じであるため、パブリックゾーンを削除する際に、プライベートゾーンではクラスターのすべての DNS 解決をシームレスに提供します。

Ingress コントローラー
デフォルトの Ingress オブジェクトはパブリックとして作成されるため、ロードバランサーはインターネットに接続され、パブリックサブネットで使用されます。デフォルト Ingress コントローラーは内部コントローラーに置き換えることができます。

API サーバー
デフォルトでは、インストールプログラムは内部トラフィックと外部トラフィックの両方で使用するための API サーバーの適切なネットワークロードバランサーを作成します。

Amazon Web Services (AWS) では、個別のパブリックロードバランサーおよびプライベートロードバランサーが作成されます。ロードバランサーは、クラスター内で使用するために追加ポートが内部で利

用可能な場合を除き、常に同一です。インストールプログラムは API サーバー要件に基づいてロードバランサーを自動的に作成または破棄しますが、クラスターはそれらを管理または維持しません。クラスターの API サーバーへのアクセスを保持する限り、ロードバランサーを手動で変更または移動できます。パブリックロードバランサーの場合、ポート 6443 は開放され、ヘルスチェックが HTTPS について **/readyz** パスに対して設定されます。

Google Cloud Platform では、内部および外部 API トラフィックの両方を管理するために単一のロードバランサーが作成されるため、ロードバランサーを変更する必要はありません。

Microsoft Azure では、パブリックおよびプライベートロードバランサーの両方が作成されます。ただし、現在の実装には制限があるため、プライベートクラスターで両方のロードバランサーを保持します。

3.6.2. DNS をプライベートに設定する

クラスターのデプロイ後に、プライベートゾーンのみを使用するように DNS を変更できます。

手順

1. クラスターの DNS カスタムリソースを確認します。

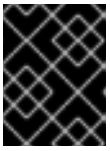
```
$ oc get dnses.config.openshift.io/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>: owned
  publicZone:
    id: Z2XXXXXXXXXXA4
status: {}
```

spec セクションには、プライベートゾーンとパブリックゾーンの両方が含まれることに注意してください。

2. DNS カスタムリソースにパッチを適用して、パブリックゾーンを削除します。

```
$ oc patch dnses.config.openshift.io/cluster --type=merge --patch='{"spec": {"publicZone": null}}'
dns.config.openshift.io/cluster patched
```

Ingress コントローラーは Ingress オブジェクトの作成時に DNS 定義を参照するため、Ingress オブジェクトを作成または変更する場合、プライベートレコードのみが作成されます。



重要

既存の Ingress オブジェクトの DNS レコードは、パブリックゾーンの削除時に変更されません。

3. オプション: クラスターの DNS カスタムリソースを確認し、パブリックゾーンが削除されていることを確認します。

```
$ oc get dnses.config.openshift.io/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>-wfpg4: owned
status: {}
```

3.6.3. Ingress コントローラーをプライベートに設定する

クラスターのデプロイ後に、その Ingress コントローラーをプライベートゾーンのみを使用するように変更できます。

手順

1. 内部エンドポイントのみを使用するようにデフォルト Ingress コントローラーを変更します。

```
$ oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: Internal
EOF
ingresscontroller.operator.openshift.io "default" deleted
ingresscontroller.operator.openshift.io/default replaced
```

パブリック DNS エントリーが削除され、プライベートゾーンエントリーが更新されます。

3.6.4. API サーバーをプライベートに制限する

クラスターを Amazon Web Services (AWS) または Microsoft Azure にデプロイした後に、プライベートゾーンのみを使用するように API サーバーを再設定することができます。

前提条件

- OpenShift CLI (**oc**) のインストール。
- **admin** 権限を持つユーザーとして Web コンソールにアクセスできること。

手順

1. AWS または Azure の Web ポータルまたはコンソールで、以下のアクションを実行します。
   a. 適切なロードバランサーコンポーネントを見つけ、削除します。
      - AWS の場合は、外部ロードバランサーを削除します。プライベートゾーンの API DNS エントリーは、同一の設定を使用する内部ロードバランサーをすでに参照するため、内部ロードバランサーを変更する必要はありません。
      - Azure の場合、ロードバランサーの **api-internal** ルールを削除します。
   b. パブリックゾーンの **api.$clustername.$yourdomain** DNS エントリーを削除します。
2. ターミナルで、クラスターマシンを一覧表示します。

```
$ oc get machine -n openshift-machine-api
NAME                          STATE     TYPE       REGION      ZONE         AGE
lk4pj-master-0                running   m4.xlarge  us-east-1   us-east-1a   17m
lk4pj-master-1                running   m4.xlarge  us-east-1   us-east-1b   17m
lk4pj-master-2                running   m4.xlarge  us-east-1   us-east-1a   17m
lk4pj-worker-us-east-1a-5fzfj running   m4.xlarge  us-east-1   us-east-1a   15m
lk4pj-worker-us-east-1a-vbghs running   m4.xlarge  us-east-1   us-east-1a   15m
lk4pj-worker-us-east-1b-zgpzg running   m4.xlarge  us-east-1   us-east-1b   15m
```

   以下の手順で、名前に **master** が含まれるコントロールプレーンマシンを変更します。

3. 各コントロールプレーンマシンから外部ロードバランサーを削除します。
   a. **master** Machine オブジェクトを編集し、外部ロードバランサーへの参照を削除します。

```
$ oc edit machines -n openshift-machine-api <master_name> 1
```

   1 変更するコントロールプレーン、またはマスター、マシンの名前を指定します。

   b. 以下の例でマークが付けられている外部ロードバランサーを記述する行を削除し、オブジェクト仕様を保存し、終了します。

```
...
spec:
  providerSpec:
    value:
      ...
      loadBalancers:
      - name: lk4pj-ext 1
```

```
    type: network (2)
  - name: lk4pj-int
    type: network
```

(1) (2) この行を削除します。

c. 名前に **master** が含まれるマシンにこのプロセスを繰り返します。