OpenShift Container Platform 4.9

クラスターの更新

OpenShift Container Platform クラスターの更新

Last Updated: 2022-01-15

OpenShift Container Platform 4.9 クラスターの更新

OpenShift Container Platform クラスターの更新

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

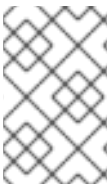
概要

本書では、OpenShift Container Platform クラスターを更新し、アップグレードする方法を説明します。クラスターの更新を、クラスターをオフラインにする必要のない単純なプロセスで実行できます。

目次

第1章 OPENSHIFT UPDATE SERVICE について

インターネットにアクセスできるクラスターの場合に、Red Hat は、パブリック API の背後にあるホスト型サービスとしてOpenShift Container Platform 更新サービスを介して OTA (over-the-air) 更新を提供します。



注記

非接続クラスターがパブリック API にアクセスできない、制限付きのネットワークを使用している場合には、OpenShift Update Service をローカルにインストールできます。OpenShift UpdateServiceのインストールと設定 を参照してください。

1.1. OPENSHIFT UPDATE SERVICE について

OpenShift Update Service (OSUS) は、Red Hat Enterprise Linux CoreOS (RHCOS) を含む OpenShift Container Platform に OTA (over-the-air) 更新を提供します。コンポーネント Operator のグラフ、または **頂点** とそれらを結ぶ **辺** を含む図表が提示されます。グラフのエッジでは、安全に更新できるバージョンが表示されます。頂点は、マネージドクラスターコンポーネントの意図された状態を指定する更新ペイロードです。

クラスター内の Cluster Version Operator (CVO) は、OpenShift Update Service をチェックして、グラフの現在のコンポーネントバージョンとグラフの情報に基づき、有効な更新および更新パスを確認します。ユーザーが更新をリクエストすると、CVO はその更新のリリースイメージを使ってクラスターをアップグレードします。リリースアーティファクトは、コンテナーイメージとして Quay でホストされます。

OpenShift Update Service が互換性のある更新のみを提供できるようにするために、リリース検証 Pipeline で自動化を支援します。それぞれのリリースアーティファクトについて、他のコンポーネントパッケージだけでなくサポートされているクラウドプラットフォームおよびシステムアーキテクチャーとの互換性の有無が検証されます。Pipeline がリリースの適合性を確認した後に、OpenShift Update Service は更新が利用可能であることを通知します。

重要

OpenShift Update Service は、現在のクラスターに推奨される更新をすべて表示します。OpenShift Update Service が推奨するアップグレードパスがない場合には、更新またはターゲットリリースに関連する既知の問題がある可能性があります。

連続更新モード中は、2 つのコントローラーが実行されます。1 つのコントローラーはペイロードマニフェストを絶えず更新し、そのマニフェストをクラスターに適用し、Operator が利用可能か、アップグレード中か、または失敗しているかに応じて Operator の制御されたロールアウトのステータスを出力します。2 つ目のコントローラーは OpenShift Update Service をポーリングして、更新が利用可能かどうかを判別します。

重要

サポートされているのは、新規バージョンへのアップグレードのみです。クラスターを以前のバージョンに戻すまたはロールバックすることはサポートされていません。アップグレードできない場合は、Red Hat サポートにお問い合わせください。

アップグレードプロセスで、Machine Config Operator (MCO) は新規設定をクラスターマシンに適用します。MCO は、マシン設定プールの **maxUnavailable** フィールドによって指定されるノードの数を分離し、それらを利用不可としてマークします。デフォルトで、この値は **1** に設定されます。次に、

MCO は新しい設定を適用して、マシンを再起動します。

Red Hat Enterprise Linux (RHEL) マシンをワーカーとして使用する場合、まず OpenShift API をそれらのマシンで更新する必要があるため、MCO は kubelet を更新しません。

新規バージョンの仕様は古い kubelet に適用されるため、RHEL マシンを **Ready** 状態に戻すことができません。マシンが利用可能になるまで更新を完了することはできません。ただし、利用不可のノードの最大数は、その数のマシンがサービス停止状態のマシンとして分離されても通常のクラスター操作が継続できるようにするために設定されます。

OpenShift Update Service は Operator および 1 つ以上のアプリケーションインスタンスで構成されます。

1.2. 管理外の OPERATOR のサポートポリシー

Operator の **管理状態** は、Operator が設計通りにクラスター内の関連するコンポーネントのリソースをアクティブに管理しているかどうかを定めます。Operator が **unmanaged** 状態に設定されている場合、これは設定の変更に応答せず、更新を受信しません。

これは非実稼働クラスターやデバッグ時に便利ですが、管理外の状態の Operator はサポートされず、クラスター管理者は個々のコンポーネント設定およびアップグレードを完全に制御していることを前提としています。

Operator は以下の方法を使用して管理外の状態に設定できます。

- **個別の Operator 設定**
  個別の Operator には、それらの設定に **managementState** パラメーターがあります。これは Operator に応じてさまざまな方法でアクセスできます。たとえば、Red Hat OpenShift Logging Operator は管理するカスタムリソース (CR) を変更することによってこれを実行しますが、Cluster Samples Operator はクラスター全体の設定リソースを使用します。

  managementState パラメーターを **Unmanaged** に変更する場合、Operator はそのリソースをアクティブに管理しておらず、コンポーネントに関連するアクションを取らないことを意味します。Operator によっては、クラスターが破損し、手動リカバリーが必要になる可能性があるため、この管理状態に対応しない可能性があります。

  警告

  個別の Operator を **Unmanaged** 状態に変更すると、特定のコンポーネントおよび機能がサポート対象外になります。サポートを継続するには、報告された問題を **Managed** 状態で再現する必要があります。

- **Cluster Version Operator (CVO) のオーバーライド**
  spec.overrides パラメーターを CVO の設定に追加すると、管理者はコンポーネントについての CVO の動作に対してオーバーライドの一覧を追加できます。コンポーネントについて **spec.overrides[].unmanaged** パラメーターを **true** に設定すると、クラスターのアップグレードがブロックされ、CVO のオーバーライドが設定された後に管理者にアラートが送信されます。

> Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.

警告

CVO のオーバーライドを設定すると、クラスター全体がサポートされない状態になります。サポートを継続するには、オーバーライドを削除した後に、報告された問題を再現する必要があります。

第2章 OPENSHIFT UPDATE SERVICE のインストールと設定

インターネットにアクセスできるクラスターの場合に、Red Hat は、パブリック API の背後にあるホスト型サービスとしてOpenShift Container Platform 更新サービスを介して OTA (over-the-air) 更新を提供します。ただし、ネットワークが制限された環境のクラスターは、パブリック API にアクセスして更新情報を取得する方法はありません。

ネットワークが制限された環境で同様のアップグレードエクスペリエンスを提供するには、OpenShift Update Service をローカルでインストールして、非接続環境で利用できるようにします。

以下のセクションでは、非接続クラスターとその基礎となるオペレーティングシステムの OTA (over-the-air) 更新を提供する方法を説明します。

2.1. 前提条件

- Operator のインストールについての詳細は、「Installing Operators from in your namespace 」を参照してください。

2.1.1. OpenShift Update Service向けの セキュリティー保護されたレジストリーへのアクセス設定

リリースイメージがセキュリティー保護されたレジストリーに含まれている場合には、「イメージレジストリーアクセスの追加トラストストアの設定」の手順を実行して、更新サービスに以下の変更を加えます。

OpenShift Update Service Operator では、設定マップのキー名 **updateservice-registry** がレジストリー CA 証明書に必要です。

更新サービス向けのイメージレジストリー CA の設定マップの例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  updateservice-registry: | 1
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | 2
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

1 OpenShift Update Service Operator では、設定マップのキー名 updateservice-registry がレジストリー CA 証明書に必要です。

2 レジストリーにポートがある場合 (例: **registry-with-port.example.com:5000**)、「**:**」は **..** に置き換える必要があります。

2.2. OPENSHIFT UPDATE SERVICE のインストール

OpenShift Update Service をインストールするには、まず OpenShift Container Platform Web コンソールまたは CLI を使用して OpenShift Update Service Operator をインストールする必要があります。

注記

ネットワークが制限された環境 (非接続クラスターとして知られる) にインストールされているクラスターの場合には、デフォルトで Operator Lifecycle Manager はリモートレジストリーでホストされる Red Hat が提供する OperatorHub ソースにアクセスできません。それらのリモートソースには完全なインターネット接続が必要であるためです。詳細は、「ネットワークが制限された環境での Operator Lifecycle Manager の使用」を参照してください。

2.2.1. Web コンソールを使用した OpenShift Update Service Operator のインストール

Web コンソールを使用して、OpenShift Update Service Operator をインストールできます。

手順

1. Web コンソールで **Operators → OperatorHub** をクリックします。

注記

Update Service と **Filter by keyword...** フィールドに入力し、素早く Operator を見つけます。

2. 利用可能な Operator の一覧から **OpenShift Update Service** を選択し、**Install** をクリックします。
   a. 本リリースで利用可能な唯一のチャネルであるため、チャネル **v1** が **Update Channel** として選択されます。
   b. **A specific namespace on the cluster**が **Installation Mode** で選択します。
   c. **Installed Namespace** の namespace を選択するか、推奨される namespace **openshift-update-service** を受け入れます。
   d. **Approval Strategy** を選択します。
      - **Automatic** ストラテジーにより、Operator Lifecycle Manager (OLM) は新規バージョンが利用可能になると Operator を自動的に更新できます。
      - **Manual** ストラテジーには、クラスター管理者が Operator の更新を承認する必要があります。
   e. **Install** をクリックします。
3. **Operators → Installed Operators** ページに切り替えて、OpenShift Update Service Operator がインストールされていることを確認します。
4. **Status** が **Succeeded** の **OpenShift Update Service** が選択された namespace に一覧表示されていることを確認します。

2.2.2. CLI を使用した OpenShift Update Service Operator のインストール

OpenShift CLI (**oc**) を使用して、OpenShift Update Service Operator をインストールできます。

手順

1. OpenShift Update Service Operator の namespace を作成します。
   a. OpenShift Update Service Operator の **namespace** オブジェクト YAML ファイル (**update-service-namespace.yaml** など) を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-update-service
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true" 1
```

   1 **openshift.io/cluster-monitoring** ラベルを設定して、kこの namespace で Operator が推奨するクラスターのモニタリングを有効にします。

   b. namespace を作成します。

```
$ oc create -f <filename>.yaml
```

   以下は例になります。

```
$ oc create -f update-service-namespace.yaml
```

2. 以下のオブジェクトを作成して OpenShift Update Service Operator をインストールします。
   a. **OperatorGroup** オブジェクト YAML ファイルを作成します (例: **update-service-operator-group.yaml**)。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: update-service-operator-group
spec:
  targetNamespaces:
  - openshift-update-service
```

   b. **OperatorGroup** オブジェクトを作成します。

```
$ oc -n openshift-update-service create -f <filename>.yaml
```

   以下は例になります。

```
$ oc -n openshift-update-service create -f update-service-operator-group.yaml
```

   c. **Subscription** オブジェクト YAML ファイルを作成します (例: **update-service-subscription.yaml**)。

   Subscription の例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: update-service-subscription
spec:
  channel: v1
  installPlanApproval: "Automatic"
  source: "redhat-operators" (1)
  sourceNamespace: "openshift-marketplace"
  name: "cincinnati-operator"
```

1. Operator を提供するカタログソースの名前を指定します。カスタム Operator Lifecycle Manager (OLM) を使用しないクラスターの場合には、**redhat-operators** を指定します。OpenShift Container Platform クラスターが、非接続クラスターとも呼ばれるネットワークが制限された環境でインストールされている場合、Operator Lifecycle Manager (OLM) の設定時に作成される **CatalogSource** オブジェクトの名前を指定します。

d. **Subscription** オブジェクトを作成します。

```
$ oc create -f <filename>.yaml
```

以下は例になります。

```
$ oc -n openshift-update-service create -f update-service-subscription.yaml
```

OpenShift Update Service Operator は **openshift-update-service** namespace にインストールされ、**openshift-update-service** namespace をターゲットにします。

3. Operator のインストールを確認します。

```
$ oc -n openshift-update-service get clusterserviceversions
```

出力例

```
NAME                         DISPLAY                VERSION   REPLACES   PHASE
update-service-operator.v4.6.0   OpenShift Update Service   4.6.0             Succeeded
...
```

OpenShift Update Service Operator が記載されている場合には、インストールが成功しています。バージョン番号が表示されるものと異なる場合があります。

2.2.3. OpenShift Update Service グラフデータコンテナーイメージの作成

OpenShift Update Service には、OpenShift Update Service がチャネルメンバーシップについての情報を取得し、更新エッジをブロックするグラフデータコンテナーイメージが必要です。通常、グラフデータはアップグレードグラフデータリポジトリーから直接取得します。インターネット接続が利用できない場合には、グラフデータを OpenShift Update Service で利用できるようにする別の方法として init コンテナーからこの情報を読み込むことができます。init コンテナーの役割として、グラフデータのローカルコピーを提供し、Pod の初期化時に init コンテナーはデータをサービスがアクセスできるボリュームにコピーすることが挙げられます。

手順

1. 以下を含む Dockerfile (**./Dockerfile** など) を作成します。

```
FROM registry.access.redhat.com/ubi8/ubi:8.1

RUN curl -L -o cincinnati-graph-data.tar.gz https://github.com/openshift/cincinnati-graph-data/archive/master.tar.gz

CMD exec /bin/bash -c "tar xvzf cincinnati-graph-data.tar.gz -C /var/lib/cincinnati/graph-data/ --strip-components=1"
```

2. 上記の手順で作成した docker ファイルを使用して、グラフデータコンテナーイメージ (例: **registry.example.com/openshift/graph-data:latest**) を構築します。

```
$ podman build -f ./Dockerfile -t registry.example.com/openshift/graph-data:latest
```

3. 前の手順で作成した graph-data コンテナーイメージを、OpenShift Update Service (例: **registry.example.com/openshift/graph-data:latest**) からアクセスできるリポジトリーにプッシュします。

```
$ podman push registry.example.com/openshift/graph-data:latest
```

注記

制限のあるネットワーク環境に存在するローカルレジストリーにグラフデータイメージをプッシュするには、前の手順で作成した graph-data コンテナーイメージを OpenShift Update Service がアクセスできるリポジトリーにコピーします。利用可能なオプションについては、**oc image mirror --help** を実行します。

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

OpenShift Update Service には、更新リリースペイロードを含む、ローカルでアクセス可能なレジストリーが必要です。

重要

OpenShift Update Service アプリケーションで、メモリーが過剰に使用されないようにするため、以下の手順に従って、リリースイメージを別のリポジトリーにミラーリングすることが推奨されます。

前提条件

- 「非接続インストールのイメージのミラーリング」から「**OpenShift Container Platform イメージリポジトリーのミラーリング**」というタイトルのセクションまでのステップを確認して完了している。
- 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。

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

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

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、アップグレードする必要のある 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。
   a. リリースバージョンをエクスポートします。

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

   <release_version> について、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: **4.6.4**)。

   b. ローカルレジストリー名とホストポートをエクスポートします。

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

   <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。

   c. ローカルリポジトリー名をエクスポートします。

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

   <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。

   d. 追加のローカルリポジトリー名をエクスポートして、リリースイメージを追加します。

```
$
LOCAL_RELEASE_IMAGES_REPOSITORY='<local_release_images_repository_name>'
```

   <local_release_images_repository_name> については、**ocp4/openshift4-release-images** 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。

   e. ミラーリングするリポジトリーの名前をエクスポートします。

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

   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。

   f. パスをレジストリープルシークレットにエクスポートします。

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

<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の絶対パスおよびファイル名を指定します。

g. リリースミラーをエクスポートします。

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

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。

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

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

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

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

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

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

iii. 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}
```

iv. 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} 1
```

1. **REMOVABLE_MEDIA_PATH** の場合は、リムーバブルメディアをマウントしたパスを使用する必要があります。

v. リリースイメージを別のリポジトリーにミラーリングします。

```
$ oc image mirror -a ${LOCAL_SECRET_JSON}
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_REL
EASE}-${ARCHITECTURE}
```

- 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、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できます。

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

2.3. OPENSHIFT UPDATE SERVICE アプリケーションの作成

OpenShift Container Platform Web コンソールまたは CLI を使用し、OpenShift Update Service アプリケーションを作成できます。

2.3.1. Web コンソールを使用した OpenShift Update Service アプリケーションの作成

OpenShift Container Platform Web コンソールを使用して、OpenShift Update Service Operator で OpenShift Update Service アプリケーションを作成できます。

前提条件

- OpenShift Update Service Operator がインストールされている。
- OpenShift Update Service の graph-data コンテナーイメージを作成して、OpenShift Update Service がアクセスできるリポジトリーにプッシュしておく。
- 現在のリリースおよび更新ターゲットリリースがローカルアクセス可能なレジストリーにミラーリングされている。

手順

1. Web コンソールで **Operators → Installed Operators** をクリックします。
2. インストールされた Operator の一覧から **OpenShift Update Service** を選択します。
3. **Update Service** タブをクリックします。
4. **Create UpdateService** をクリックします。
5. **service** など、**Name** フィールドに名前を入力します。

6. **Graph Data Image** フィールドに「OpenShift Update Service グラフデータコンテナーイメージの作成」で作成した graph-data コンテナーイメージにローカルの pullspec を入力します (例: **registry.example.com/openshift/graph-data:latest**)。
7. **Releases** フィールドに、「OpenShift Container Platform イメージリポジトリーのミラーリング」でリリースイメージを含むように作成したローカルのレジストリーとリポジトリー (例: **registry.example.com/ocp4/openshift4-release-images**) を入力します。
8. **Replicas** フィールドに **2** と入力します。
9. **Create** をクリックして OpenShift Update Service アプリケーションを作成します。
10. OpenShift Update Service アプリケーションを検証します。
    - **Update Service** タブの **UpdateServices** 一覧から、作成した Update Service アプリケーションをクリックします。
    - **Resources** タブをクリックします。
    - 各アプリケーションリソースのステータスが **Created** であることを確認します。

2.3.2. CLI を使用した OpenShift Update Service アプリケーションの作成

OpenShift CLI (**oc**) を使用して、OpenShift Update Service アプリケーションを作成できます。

前提条件

- OpenShift Update Service Operator がインストールされている。
- OpenShift Update Service の graph-data コンテナーイメージを作成して、OpenShift Update Service がアクセスできるリポジトリーにプッシュしておく。
- 現在のリリースおよび更新ターゲットリリースがローカルアクセス可能なレジストリーにミラーリングされている。

手順

1. OpenShift Update Service ターゲット namespace を設定します (例: **openshift-update-service**)。

```
$ NAMESPACE=openshift-update-service
```

   namespace は Operator グループの **targetNamespaces** 値と一致する必要があります。

2. OpenShift Update Service アプリケーションの名前 (例: **service**) を設定します。

```
$ NAME=service
```

3. 「OpenShift Container Platform イメージリポジトリーの設ミラーリング」(例: **registry.example.com/ocp4/openshift4-release-images**) に設定されるように、リリースイメージのローカルレジストリーおよびリポジトリーを設定します。

```
$ RELEASE_IMAGES=registry.example.com/ocp4/openshift4-release-images
```

4. 「OpenShift Update Service グラフデータコンテナーイメージの作成」で作成した

4. 「OpenShift Update Service グラフデータコンテナーイメージの作成」で作成した graph-data コンテナーイメージにローカルの pullspec を入力します (例: **registry.example.com/openshift/graph-data:latest**)。

```
$ GRAPH_DATA_IMAGE=registry.example.com/openshift/graph-data:latest
```

5. OpenShift Update Service アプリケーションオブジェクトを作成します。

```
$ oc -n "${NAMESPACE}" create -f - <<EOF
apiVersion: updateservice.operator.openshift.io/v1
kind: UpdateService
metadata:
  name: ${NAME}
spec:
  replicas: 2
  releases: ${RELEASE_IMAGES}
  graphDataImage: ${GRAPH_DATA_IMAGE}
EOF
```

6. OpenShift Update Service アプリケーションを検証します。

   a. 以下のコマンドを使用してポリシーエンジンルートを取得します。

```
$ while sleep 1; do POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
"${NAME}")"; SCHEME="${POLICY_ENGINE_GRAPH_URI%%:*}"; if test "${SCHEME}"
= http -o "${SCHEME}" = https; then break; fi; done
```

   コマンドが成功するまでポーリングが必要になる場合があります。

   b. ポリシーエンジンからグラフを取得します。**チャネル** に有効なバージョンを指定してください。たとえば、OpenShift Container Platform 4.9 で実行している場合は、**stable-4.9** を使用します。

```
$ while sleep 10; do HTTP_CODE="$(curl --header Accept:application/json --output
/dev/stderr --write-out "%{http_code}" "${POLICY_ENGINE_GRAPH_URI}?
channel=stable-4.6")"; if test "${HTTP_CODE}" -eq 200; then break; fi; echo
"${HTTP_CODE}"; done
```

   これにより、グラフ要求が成功するまでポーリングされます。ただし、ミラーリングしたリリースイメージによっては、生成されるグラフが空白の場合があります。

注記

ポリシーエンジンのルート名は、RFC-1123 に基づき、64 文字以上を指定できません。**host must conform to DNS 1123 naming convention and must be no more than 63 characters** が原因で、**ReconcileCompleted** のステータスが **false**、理由が **CreateRouteFailed** となっている場合には、更新サービスをもう少し短い名前で作成してみてください。

2.3.3. Cluster Version Operator (CVO) の設定

OpenShift Update Service Operator をインストールして、OpenShift Update Service アプリケーションを作成した後に、ローカルインストールされた OpenShift Update Service からグラフデータをプルするように Cluster Version Operator (CVO) を更新できます。

前提条件

- OpenShift Update Service Operator がインストールされている。
- OpenShift Update Service の graph-data コンテナーイメージを作成して、OpenShift Update Service がアクセスできるリポジトリーにプッシュしておく。
- 現在のリリースおよび更新ターゲットリリースがローカルアクセス可能なレジストリーにミラーリングされている。
- OpenShift Update Service アプリケーションが作成されている。

手順

1. OpenShift Update Service ターゲット namespace を設定します (例: **openshift-update-service**)。

```
$ NAMESPACE=openshift-update-service
```

2. OpenShift Update Service アプリケーションの名前 (例: **service**) を設定します。

```
$ NAME=service
```

3. ポリシーエンジンルートを取得します。

```
$ POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
"${NAME}")"
```

4. プルグラフデータのパッチを設定します。

```
$ PATCH="{\"spec\":{\"upstream\":\"${POLICY_ENGINE_GRAPH_URI}\"}}"
```

5. CVO にパッチを適用して、ローカルの OpenShift Update Service を使用します。

```
$ oc patch clusterversion version -p $PATCH --type merge
```

注記

「クラスター全体のプロキシーの有効化」を参照して、CA が更新サーバーを信頼するように設定します。

2.4. OPENSHIFT UPDATE SERVICE アプリケーションの削除

OpenShift Container Platform Web コンソールまたは CLI を使用して OpenShift Update Service アプリケーションを削除できます。

2.4.1. Web コンソールを使用した OpenShift Update Service アプリケーションの削除

OpenShift Container Platform Web コンソールを使用して、OpenShift Update Service Operator で OpenShift Update Service アプリケーションを削除できます。

前提条件

- OpenShift Update Service Operator がインストールされている。

手順

1. Web コンソールで **Operators → Installed Operators** をクリックします。
2. インストールされた Operator の一覧から **OpenShift Update Service** を選択します。
3. **Update Service** タブをクリックします。
4. インストールされた OpenShift Update Service アプリケーションの一覧から、削除するアプリケーションを選択して、**Delete UpdateService** をクリックします。
5. **Delete UpdateService?** 確認ダイアログで、**Delete** をクリックし、削除を確定します。

2.4.2. CLI を使用した OpenShift Update Service アプリケーションの削除

OpenShift CLI (**oc**) を使用して、OpenShift Update Service アプリケーションを削除できます。

手順

1. OpenShift Update Service アプリケーションを作成した namespace を使用して OpenShift Update Service アプリケーション名を取得します (例: **openshift-update-service**)。

```
$ oc get updateservice -n openshift-update-service
```

出力例

```
NAME      AGE
service   6s
```

2. 直前の手順の **NAME** の値を使用して OpenShift Update Service アプリケーションと、OpenShift Update Service アプリケーションを作成した namespace (例: **openshift-update-service**) を削除します。

```
$ oc delete updateservice service -n openshift-update-service
```

出力例

```
updateservice.updateservice.operator.openshift.io "service" deleted
```

2.5. OPENSHIFT UPDATE SERVICE OPERATOR のアンインストール

OpenShift Update Service をアンインストールするには、まず OpenShift Container Platform Web コンソールまたは CLI を使用してすべての OpenShift Update Service アプリケーションを削除する必要があります。

2.5.1. Web コンソールを使用した OpenShift Update Service Operator のアンインストール

OpenShift Container Platform Web コンソールを使って OpenShift Update Service Operator をアンインストールすることができます。

前提条件

- OpenShift Update Service アプリケーションがすべて削除されている。

手順

1. Web コンソールで **Operators → Installed Operators** をクリックします。
2. インストールされた Operator の一覧から **OpenShift Update Service** を選択し、**Uninstall Operator** をクリックします。
3. **Uninstall Operator?** 確認ダイアログから **Uninstall** をクリックし、アンインストールを確定します。

2.5.2. CLI を使用した OpenShift Update Service Operator のアンインストール

OpenShift CLI (**oc**) を使用して、OpenShift Update Service Operator をアンインストールできます。

前提条件

- OpenShift Update Service アプリケーションがすべて削除されている。

手順

1. OpenShift Update Service Operator (例: **openshift-update-service**) が含まれるプロジェクトに切り替えます。

```
$ oc project openshift-update-service
```

出力例

```
Now using project "openshift-update-service" on server "https://example.com:6443".
```

2. OpenShift Update Service Operator Operator グループの名前を取得します。

```
$ oc get operatorgroup
```

出力例

```
NAME                              AGE
openshift-update-service-fprx2   4m41s
```

3. Operator グループを削除します (例: **openshift-update-service-fprx2**)。

```
$ oc delete operatorgroup openshift-update-service-fprx2
```

出力例

```
operatorgroup.operators.coreos.com "openshift-update-service-fprx2" deleted
```

4. OpenShift Update Service Operator サブスクリプションの名前を取得します。

```
$ oc get subscription
```

出力例

```
NAME                        PACKAGE                     SOURCE                      CHANNEL
update-service-operator   update-service-operator   updateservice-index-catalog   v1
```

5. 直前の手順で **Name** の値を使用して、**currentCSV** フィールドで、サブスクライブされた OpenShift Update Service Operator の現行バージョンを確認します。

```
$ oc get subscription update-service-operator -o yaml | grep " currentCSV"
```

出力例

```
  currentCSV: update-service-operator.v0.0.1
```

6. サブスクリプション (例: **update-service-operator**) を削除します。

```
$ oc delete subscription update-service-operator
```

出力例

```
subscription.operators.coreos.com "update-service-operator" deleted
```

7. 直前の手順の **currentCSV** 値を使用し、OpenShift Update Service Operator の CSV を削除します。

```
$ oc delete clusterserviceversion update-service-operator.v0.0.1
```

出力例

```
clusterserviceversion.operators.coreos.com "update-service-operator.v0.0.1" deleted
```

第3章 アップグレードチャネルおよびリリースについて

OpenShift Container Platform 4.1 で、Red Hat はクラスターのアップグレードの適切なリリースバージョンを推奨するためにチャネルという概念を導入しました。アップグレードのペースを制御することで、これらのアップグレードチャネルからアップグレードストラテジーを選択することができます。アップグレードチャネルは OpenShift Container Platform のマイナーバージョンに関連付けられます。たとえば、OpenShift Container Platform 4.9 アップグレードチャネルでは 4.9 リリースへのアップグレードおよび 4.9 内のアップグレードが推奨されます。また、4.8 内のアップグレードおよび 4.8 から 4.9 へのアップグレードが推奨されます。これにより、4.8 のクラスターを最終的に 4.9 にアップグレードできます。4.10 以降のリリースへのアップグレードは推奨されていません。このストラテジーにより、管理者は OpenShift Container Platform の次のマイナーバージョンへのアップグレードに関して明確な決定を行うことができます。

アップグレードチャネルはリリースの選択のみを制御し、インストールするクラスターのバージョンには影響を与えません。 OpenShift Container Platform の特定のバージョンの **openshift-install** バイナリーファイルは常に該当バージョンをインストールします。

OpenShift Container Platform 4.9 は以下のアップグレードチャネルを提供します。

- **candidate-4.9**
- **fast-4.9**
- **stable-4.9**
- **EUS-4.y** （4.8 などの数値が付けられたクラスターリリースを実行する場合のみ）

Cluster Version Operator をアップグレード推奨サービスから利用可能な更新を取得する必要がない場合は、OpenShift CLI で **oc adm upgrade channel** コマンドを使用して空のチャネルを設定できます。この設定は、クラスターがネットワークアクセスが制限された状況で、ローカルで到達可能なアップグレードに関する推奨サービスがない場合に役立ちます。

警告

Red Hat は、Openshift Update Service が提案するバージョンのみへのアップグレードを推奨しています。マイナーバージョンのアップグレードでは、バージョンは連続している必要があります。Red Hat は、連続していないバージョンへのアップグレードをテストしず、以前のバージョンとの互換性を保証しません。

3.1. アップグレードパスとリリースパス

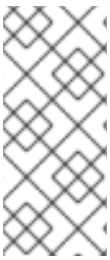
クラスター管理者は、Web コンソールからアップグレードチャネルを設定できます。

3.1.1. candidate-4.9 チャネル

candidate-4.9 チャネルには、z-stream (4.9.z) リリースの候補となるビルドが含まれます。リリース候補には、製品のすべての機能が含まれますが、それらがサポートされる訳ではありません。リリース候補を使用して機能の受け入れテストを実行し、OpenShift Container Platform の次のバージョンへの対応を支援します。リリース候補は、名前に **-rc** など、プレリリースバージョン を含まない、候補チャネルで利用可能なビルドを指します。候補チャネルでバージョンが利用可能になると、さらに品質の

チェックが行われます。品質基準を満たす場合には、これは **fast-4.9** または **stable-4.9** チャンネルにプロモートされます。このストラテジーにより、特定のリリースが **candidate-4.9** チャネルと **fast-4.9** または **stable-4.9** チャネルの両方で利用可能な場合、そのリリースは Red Hat でサポートされるバージョンということになります。**candidate-4.9** チャネルには、いずれのチャネルでも推奨されている更新のないリリースバージョンを含めることができます。

candidate-4.9 チャネルを使用して、OpenShift Container Platform の直前のマイナーバージョンからアップグレードできます。



注記

リリース候補は夜間ビルドとは異なります。夜間ビルドは各種機能への早期アクセスのために利用できますが、夜間ビルドへの/からの更新は推奨されておらず、サポートもされていません。夜間ビルドはいずれのアップグレードチャネルでも利用できません。ビルドについての詳細は、OpenShift Container Platform の リリースのステータス を参照できます。

3.1.2. fast-4.9 チャネル

fast-4.9 チャネルは、Red Hat が一般公開リリースとして指定のバージョンを宣言するとすぐに 4.9 の新規のバージョンおよびそれより前のマイナーバージョンで更新されます。そのため、これらのリリースは完全にサポートされ、実稼働用の品質があり、これらのリリースのプロモート元の **candidate-4.9** チャネルのリリース候補として利用可能であった間のパフォーマンスにも問題はありませんでした。リリースは **fast-4.9** チャネルに表示されてからしばらくすると、**stable-4.9** チャネルに追加されます。リリースは **stable-4.9** チャネルに表示される前に、**fast-4.9** チャネルに表示されることはありません。

fast-4.9 チャネルを使用して、OpenShift Container Platform の以前のマイナーバージョンからのアップグレードを実行できます。

3.1.3. stable-4.9 チャネル

fast-4.9 チャネルにはエラータの公開後すぐにリリースが組み込まれ、リリースの **stable-4.9** チャネルへの追加は遅延します。この期間中、接続環境のカスタマープログラム(Connected Customer Program) に関わる Red Hat SRE チーム、Red Hat サポートサービス、および実稼働前および実稼働環境からリリースの安定性についてのデータが収集されます。

stable-4.9 チャネルを使用して、OpenShift Container Platform の以前のマイナーバージョンからのアップグレードを実行できます。

3.1.4. EUS-4.y チャネル

stable チャネルのほかに、番号が偶数の OpenShift Container Platform マイナーバージョンはすべて Extended Update Support (延長アップデートサポート) (EUS) を提供します。これらの EUS バージョンでは、標準およびプレミアムサブスクリプションをお持ちのお客様のサポートフェーズを 18 カ月に拡張します。

OpenShift Container Platform 4.y が EUS フェーズに移行するまで **stable-4.y** と **eus-4.y** チャネル間に相違はありませんが、 EUS チャネルが利用可能になり次第、**eus-4.y** に切り換えることができます。

次の EUS チャンネルにアップグレードする場合は、次の EUS バージョンに到達するまで、次の EUS チャンネルに切り替えてアップグレードすることができます。

このアップグレードプロセスは、**eus-4.6** チャネルには適用されません。

3.1.5. アップグレードバージョンパス

OpenShift Container Platform では、インストールされた OpenShift Container Platform のバージョンと、次のリリースにアクセスするために選択したチャネル内のパスの確認を可能にするアップグレード推奨サービスが提供されます。

fast-4.9 チャネルでは以下を確認できます。

- 4.9.0
- 4.9.1
- 4.9.3
- 4.9.4

このサービスは、テスト済みの重大な問題のないアップグレードのみを推奨します。これは、既知の脆弱性を含む OpenShift Container Platform のバージョンへの更新を提案しません。クラスターが 4.9.1 にあり、OpenShift Container Platform が 4.9.4 を提案している場合、4.9.1 から 4.9.4 に更新しても問題がありません。パッチの連続する番号のみに依存しないようにしてください。たとえば、この例では 4.9.2 はチャネルで利用可能な状態ではなく、これまで利用可能になったことがありません。

更新の安定性は、チャネルによって異なります。**candidate-4.9** チャネルに更新についての推奨があるからといって、その更新が必ずしもサポートされる訳ではありません。つまり、更新について深刻な問題がまだ検出されていないものの、この更新の安定性についての提案を導くようなトラフィックの安定性はとくに確認されていない可能性があります。任意の時点で **fast-4.9** または **stable-4.9** チャネルの更新の推奨がある場合は、更新がサポートされていることを示します。リリースがチャネルから削除されることは決してありませんが、深刻な問題を示す更新の推奨はすべてのチャネルから削除されます。更新の推奨が削除された後に開始された更新は依然としてサポートされます。

Red Hat は最終的には、**fast-4.9** または **stable-4.9** チャネルのサポートされるリリースから 4.9.z の最新リリースへのサポートされる更新パスを提供します。ただし、問題のあるリリースからの安全なパスが構築され、検証される間に遅延が生じる可能性があります。

3.1.6. 高速かつ安定したチャネルの使用およびストラテジー

fast-4.9 および **stable-4.9** チャネルでは、一般公開リリースが利用可能になり次第これを受信するか、または Red Hat がそれらの更新のロールアウトを制御するようにするかを選択することができます。問題がロールアウト時またはロールアウト後に検出される場合、該当バージョンへのアップグレードは **fast-4.9** および **stable-4.9** チャネルの両方でブロックされ、新たに推奨されるアップグレード先の新規バージョンが導入される可能性があります。

fast-4.9 チャネルで実稼働前のシステムを設定し、**stable-4.9** チャネルで実稼働システムを設定してから Red Hat の接続環境のカスタマープログラム (Connected Customer Program) に参加することで、お客様のプロセスを改善することができます。Red Hat はこのプログラムを使用して、ご使用の特定のハードウェアおよびソフトウェア設定に対する更新の影響の有無を確認します。今後のリリースでは、更新が **fast-4.9** から **stable-4.9** チャネルに移行するペースが改善されるか、変更される可能性があります。

3.1.7. ネットワークが制限された環境のクラスター

OpenShift Container Platform クラスターのコンテナーイメージを独自に管理する場合には、製品リリースに関連する Red Hat エラータを確認し、アップグレードへの影響に関するコメントに留意する必要があります。アップグレード時に、インターフェースにこれらのバージョン間の切り替えについての警告が表示される場合があります。そのため、これらの警告を無視するかどうかを決める前に適切なバージョンを選択していることを確認する必要があります。

3.1.8. CLI プロファイル間の切り替え

チャネルは、Web コンソールまたは **adm upgrade channel** コマンドで切り換えることができます。

```
$ oc adm upgrade channel <channel>
```

Web コンソールは、現在のリリースを含まないチャネルに切り替えると、アラートを表示します。Web コンソールは、現在のリリースのないチャネルにある更新を推奨していません。ただし、任意の時点で元のチャネルに戻ることができます。

チャネルの変更は、クラスターのサポート可能性に影響を与える可能性があります。以下の条件が適用されます。

- **stable-4.9** チャネルから **fast-4.9** チャネルに切り換える場合も、クラスターは引き続きサポートされます。
- **candidate-4.9** チャネルに切り換えることはできますが、このチャネルの一部のリリースはサポートされない可能性があります。
- 現在のリリースが一般利用公開リリースの場合、**candidate-4.9** チャネルから **fast-4.9** チャネルに切り換えることができます。
- **fast-4.9** チャネルから **stable-4.9** チャネルに常に切り換えることができます。現在のリリースが最近プロモートされた場合は、リリースが **stable-4.9** にプロモートされるまでに最長 1 日分の遅延が生じる可能性があります。

第4章 OPENSHIFT CONTAINER PLATFORM 4.9 への更新の準備

OpenShift Container Platform 4.9 は Kubernetes 1.22 を使用します。これにより、非推奨となった **v1beta1** API が大幅に削除されました。

OpenShift Container Platform 4.8.14では、クラスターをOpenShift Container Platform 4.8から4.9にアップグレードする前に、管理者が手動で承認する必要があるという要件が導入されました。削除されたAPIが、クラスター上で実行されている、またはクラスターと対話しているワークロード、ツール、またはその他のコンポーネントによって引き続き使用されるOpenShift Container Platform 4.9にアップグレードした後の問題を防ぐ上で役立ちます。管理者は、削除する予定の使用中のAPIのクラスタを評価し、影響を受けるコンポーネントを移行して適切な新規APIバージョンを使用する必要があります。この評価および移行が完了したら、管理者は確認応答を提供できます。

OpenShift Container Platform 4.8 クラスターを 4.9 にアップグレードする前に、管理者の確認を提供する必要があります。

4.1. KUBERNETES API の削除

OpenShift Container Platform 4.9 は Kubernetes 1.22 を使用しますが、これにより、以下の非推奨となった **v1beta1** API が削除されました。**v1** API バージョンを使用するようにマニフェストおよび API クライアントを移行する必要があります。削除された API の移行についての詳細は、Kubernetes documentationを参照してください。

表4.1 v1beta1 API が Kubernetes 1.22 から削除

リソース	API	主な変更
APIService	**apiregistration.k8s.io/v1beta1**	不要
CertificateSigningRequest	**certificates.k8s.io/v1beta1**	必要
ClusterRole	**rbac.authorization.k8s.io/v1beta1**	不要
ClusterRoleBinding	**rbac.authorization.k8s.io/v1beta1**	不要
CSIDriver	**storage.k8s.io/v1beta1**	不要
CSINode	**storage.k8s.io/v1beta1**	不要
CustomResourceDefinition	**apiextensions.k8s.io/v1beta1**	必要
Ingress	**extensions/v1beta1**	必要
Ingress	**networking.k8s.io/v1beta1**	必要
IngressClass	**networking.k8s.io/v1beta1**	不要
Lease	**coordination.k8s.io/v1beta1**	不要
LocalSubjectAccessReview	**authorization.k8s.io/v1beta1**	必要

リソース	API	主な変更
MutatingWebhookConfiguration	**admissionregistration.k8s.io/v1beta1**	必要
PriorityClass	**scheduling.k8s.io/v1beta1**	不要
ロール	**rbac.authorization.k8s.io/v1beta1**	不要
RoleBinding	**rbac.authorization.k8s.io/v1beta1**	不要
SelfSubjectAccessReview	**authorization.k8s.io/v1beta1**	必要
StorageClass	**storage.k8s.io/v1beta1**	不要
SubjectAccessReview	**authorization.k8s.io/v1beta1**	必要
TokenReview	**authentication.k8s.io/v1beta1**	不要
ValidatingWebhookConfiguration	**admissionregistration.k8s.io/v1beta1**	必要
VolumeAttachment	**storage.k8s.io/v1beta1**	不要

4.2. 削除された API に関するクラスターの評価

削除される API が使用されている場所を管理者が特定するのに役立つ方法は複数あります。ただし、OpenShift Container Platform は、アイドル状態や外部ツールが使用されるワークロードなどのすべてのインスタンスを特定できません。すべてのワークロードと削除された API のインスタンスに対する他の統合を適切に評価することは管理者の責任です。

4.2.1. 削除された API の使用を特定するためのアラートの確認

次のリリースで削除予定の API が使用されている場合に 2 つのアラートが発生します。

- **APIRemovedInNextReleaseInUse**: OpenShift Container Platform の次のリリースで削除される API の場合
- **APIRemovedInNextEUSReleaseInUse**: 次の OpenShift Container Platform Extended Update Support (EUS) リリースで削除される API の場合

これらのアラートのいずれかがクラスターで実行している場合は、アラートを確認し、マニフェストおよび API クライアントを移行して新規 API バージョンを使用することによりアラートをクリアします。**APIRequestCount API** を使用して、使用中の API と、削除された API を使用しているワークロードに関する詳細情報を取得できます。

4.2.2. APIRequestCount を使用した削除された API の使用の特定

APIRequestCount APIを使用して API 要求を追跡し、それらのいずれかが削除された API のいずれかを使用しているかどうかを確認することができます。

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ようにする必要があります。

手順

- 以下のコマンドを実行して、出力の **REMOVEDINRELEASE** 列を確認して、現在使用中の削除済みの API を特定します。

```
$ oc get apirequestcounts
```

出力例

```
NAME                                REMOVEDINRELEASE   REQUESTSINCURRENTHOUR   REQUESTSINLAST24H
cloudcredentials.v1.operator.openshift.io                32                      111
ingresses.v1.networking.k8s.io                           28                      110
ingresses.v1beta1.extensions        1.22                 16                      66
ingresses.v1beta1.networking.k8s.io 1.22                 0                       1
installplans.v1alpha1.operators.coreos.com               93                      167
...
```

重要

結果に表示される以下のエントリーは無視しても問題ありません。

- **system:serviceaccount:kube-system:generic-garbage-collector** は、削除するリソースを検索するすべての登録済み API を説明するため、結果に表示されます。
- **system:kube-controller-manager** は、クォータの実施中にすべてのリソースをカウントするために全リソースを調べるため、結果に表示されます。

-o jsonpath を使用して結果をフィルターすることもできます。

```
$ oc get apirequestcounts -o jsonpath='{range .items[?(@.status.removedInRelease!="")]}{.status.removedInRelease}{"\t"}{.metadata.name}{"\n"}{end}'
```

出力例

```
1.22    certificatesigningrequests.v1beta1.certificates.k8s.io
1.22    ingresses.v1beta1.extensions
1.22    ingresses.v1beta1.networking.k8s.io
```

4.2.3. APIRequestCount を使用した、削除された API を使用しているワークロードを特定する

指定の API バージョンの **APIRequestCount** リソースを確認して、API を使用しているワークロードを特定することができます。

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ようにする必要があります。

手順

- 以下のコマンドを実行して、**username** および **userAgent** を確認して、API を使用しているワークロードを特定できるようにします。

```
$ oc get apirequestcounts <resource>.<version>.<group> -o yaml
```

以下は例になります。

```
$ oc get apirequestcounts ingresses.v1beta1.networking.k8s.io -o yaml
```

-o jsonpath を使用して、**APIRequestCount** リソースから **username** の値を抽出することもできます。

```
$ oc get apirequestcounts ingresses.v1beta1.networking.k8s.io -o jsonpath='{range
..username}{$}{"\n"}{end}' | sort | uniq
```

出力例

```
user1
user2
app:serviceaccount:delta
```

4.3. 削除された API インスタンスのインスタンスの移行

削除された Kubernetes API を移行する方法は、Kubernetes ドキュメントのDeprecated API Migration Guideを参照してください。

4.4. 管理者の確認の提供

削除された API についてクラスターを評価し、削除された API を移行すると、クラスターが OpenShift Container Platform 4.8 から 4.9 にアップグレードできることを確認できます。

警告

この管理者の確認を提供する前に、削除されたAPIのすべての使用が解決され、必要に応じて移行されたことを確認するすべての責任は管理者にあることに注意してください。OpenShift Container Platform はその評価を支援できますが、とくにアイドル状態のワークロードや外部ツールなど、削除された API の考えられるすべての用途を特定することはできません。

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ようにする必要があります。

手順

- 以下のコマンドを実行して、評価を完了し、クラスターが OpenShift Container Platform 4.9 にアップグレードできることを確認します。

```
$ oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.8-kube-1.22-api-removals-in-4.9":"true"}}' --type=merge
```

第5章 WEB コンソールを使用したマイナーバージョン内でのクラスターの更新

Web コンソールを使用して、OpenShift Container Platform クラスターの更新またはアップグレードを実行できます。以下の手順では、マイナーバージョン内でクラスターを更新します。マイナーバージョン間でクラスターを更新する場合は、同じ手順を使用できます。

注記

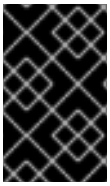
Web コンソールまたは **oc adm upgrade channel <channel>** を使用して更新チャネルを変更します。4.9 チャネルに更新を加えた後に更新を完了するために、CLI を使用してマイナーバージョン内でクラスターを更新する手順を実行できます。

5.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできること。RBAC の使用によるパーミッションの定義および適用を参照してください。
- アップグレードが失敗し、クラスターを直前の状態に復元する必要がある場合に最新の etcd バックアップがあること。
  OpenShift Container Platform 4.9 では、etcd バージョン 3.4 から 3.5 にアップグレードする必要があります。etcd Operator がアップグレードを停止すると、アラートがトリガーされます。このアラートをクリアするには、現在の etcd バックアップがあり、**--force** フラグを使用してアップグレードを再起動します。

  ```
  $ oc adm upgrade --force
  ```

- Operator Lifecycle Manager (OLM) で以前にインストールされたすべての Operator が、最新チャネルの最新バージョンに更新されていることを確認します。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターのアップグレード時に現行のマイナーバージョンから次のマイナーバージョンに切り替わる際、確実に有効なアップグレードパスがあるようにします。詳細は、「インストールされた Operator のアップグレード」を参照してください。
- すべてのマシン設定プール (MCP) が実行中であり、一時停止していないことを確認します。一時停止した MCP に関連付けられたノードは、更新プロセス中にスキップされます。カナリアロールアウト更新ストラテジーを実行している場合は、MCP を一時停止することができます。
- クラスターで手動で保守される認証情報を使用する場合は、Cloud Credential Operator (CCO) がアップグレード可能な状態であることを確認します。「AWS」、「Azure」、または「GCP」の **手動で保守される認証情報を使用したクラスターのアップグレード**」を参照してください。
- クラスターが、AWS Secure Token Service (STS)で手動でメンテナンスされる認証情報を使用する場合は、アップグレードするリリースイメージから **ccoctl** ユーティリティーのコピーを取得し、これを使用して更新された認証情報を処理します。詳細は、**Upgrading an OpenShift Container Platform cluster configured for manual mode with STS**を参照してください。
- Kubernetes 1.22 で削除された API の一覧を確認し、影響を受けるコンポーネントを移行して新しいAPIバージョンを使用し、管理者確認を提供します。詳細は、Preparing to update to OpenShift Container Platform 4.9を参照してください。



重要

unsupportedConfigOverrides セクションを使用して Operator の設定を変更することはサポートされておらず、クラスターのアップグレードをブロックする可能性があります。クラスターをアップグレードする前に、この設定を削除する必要があります。

重要

Prometheus の割り当てられた PVC を使用してクラスターモニタリングを実行している場合、クラスターのアップグレード時に OOM による強制終了が生じる可能性があります。永続ストレージが Prometheus 用に使用される場合、Prometheus のメモリー使用量はクラスターのアップグレード時、およびアップグレードの完了後の数時間で 2 倍になります。OOM による強制終了の問題を回避するには、ワーカーノードで、アップグレード前に利用可能なメモリーのサイズを 2 倍にできるようにします。たとえば、推奨される最小ノード（RAM が 8 GB の 2 コア）でモニタリングを実行している場合は、メモリーを 16 GB に増やします。詳細は、BZ#1925061 を参照してください。

関連情報

- 「管理外の Operator のサポートポリシー」

5.2. カナリアロールアウト更新の実行

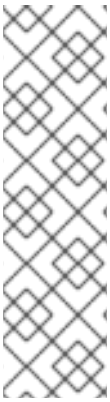
特定のユースケースでは、特定ノードを残りのクラスターと同時に更新しない、制御された更新プロセスが必要になる場合があります。これらのユースケースには、以下のようなものがありますが、これに限定されません。

- 更新時に利用できないミッションクリティカルなアプリケーションがあります。更新後の小規模なバッチで、ノードのアプリケーションを徐々にテストすることができます。
- すべてのノードを更新することができない小規模なメンテナンス期間がある場合や、複数のメンテナンスウィンドウがあります。

ローリングアップデートのプロセスは、通常の更新ワークフロー **ではありません**。大規模なクラスターの場合は、複数のコマンドを実行する必要がある時間のかかるプロセスになります。この複雑さにより、クラスター全体に影響を与える可能性のあるエラーが発生する場合があります。組織がローリングアップデートを使用し、開始前にプロセスの実装を慎重に計画するかどうかを慎重に検討することが推奨されます。

本トピックで説明されているローリングアップデートプロセスでは、以下が関係します。

- 1 つ以上のカスタムマシン設定プール (MCP) の作成。
- これらのノードをカスタム MCP に移動するためにすぐに更新しない各ノードのラベル付け。
- カスタム MCP の一時停止。これにより、それらのノードへの更新が回避されます。
- クラスターの更新の実行。
- それらのノードで更新をトリガーする 1 つのカスタム MCP の一時停止解除。
- これらのノードでアプリケーションをテストし、新たに更新されたノードでアプリケーションが想定どおりに機能していることを確認。
- 必要に応じて、小規模なバッチの残りのノードからカスタムラベルを削除し、それらのノードでアプリケーションのテスト。



注記

MCP を一時停止にすると、Machine Config Operator が関連付けられたノードに設定変更を適用できなくなります。MCP を一時停止することにより、**kube-apiserver-to-kubelet-signer** CA 証明書の自動 CA ローテーションを含め、自動的にローテーションされる証明書が関連付けられたノードにプッシュされないようにします。MCP が **kube-apiserver-to-kubelet-signer** CA 証明書の期限が切れ、MCO が証明書を自動的に更新しようとすると、新規証明書が作成されますが、適切なマシン設定プールのノード全体では適用されません。これにより、**oc debug**、**oc logs**、**oc exec**、**oc attach** など、複数の **oc** コマンドで問題が発生します。MCP の一時停止は、**kube-apiserver-to-kubelet-signer** CA 証明書の有効期限を慎重に考慮して、短期間のみ行う必要があります。

カナリアロールアウト更新プロセスを使用する場合は、「カナリアロールアウト更新の実行」を参照してください。

5.3. MACHINEHEALTHCHECK リソースの一時停止

アップグレードプロセスで、クラスター内のノードが一時的に利用できなくなる可能性があります。ワーカーノードの場合、マシンのヘルスチェックにより、このようなノードは正常ではないと識別され、それらが再起動される場合があります。このようなノードの再起動を回避するには、クラスターを更新する前にすべての **MachineHealthCheck** リソースを一時停止します。

前提条件

- OpenShift CLI (**oc**) をインストールすること。

手順

1. 一時停止する利用可能なすべての **MachineHealthCheck** リソースを一覧表示するには、以下のコマンドを実行します。

```
$ oc get machinehealthcheck -n openshift-machine-api
```

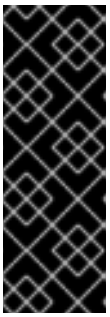
2. マシンヘルスチェックを一時停止するには、**cluster.x-k8s.io/paused=""** アノテーションを **MachineHealthCheck** リソースに追加します。以下のコマンドを実行します。

```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused=""
```

注釈付きの **MachineHealthCheck** リソースは以下の YAML ファイルのようになります。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
  annotations:
    cluster.x-k8s.io/paused: ""
spec:
  selector:
    matchLabels:
      role: worker
  unhealthyConditions:
  - type:    "Ready"
    status:  "Unknown"
```

```
    timeout: "300s"
  - type:    "Ready"
    status:  "False"
    timeout: "300s"
  maxUnhealthy: "40%"
status:
  currentHealthy: 5
  expectedMachines: 5
```



重要

クラスターの更新後にマシンヘルスチェックを再開します。チェックを再開するには、以下のコマンドを実行して **MachineHealthCheck** リソースから pause アノテーションを削除します。

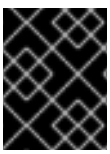
```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused-
```

5.4. 単一ノードの OPENSHIFT CONTAINER PLATFORM の更新

コンソールまたは CLI のいずれかを使用して、単一ノードの OpenShift Container Platform クラスターを更新またはアップグレードできます。

ただし、以下の制限事項に注意してください。

- 他にヘルスチェックを実行するノードがないので、**MachineHealthCheck** リソースを一時停止する時に課される前提条件は必要ありません。
- etcd バックアップを使用した単一ノードの OpenShift Container Platform クラスターの復元は、正式にはサポートされていません。ただし、アップグレードに失敗した場合には、etcd バックアップを実行することが推奨されます。コントロールプレーンが正常である場合には、バックアップを使用してクラスターを以前の状態に復元できる場合があります。
- 単一ノードの OpenShift Container Platform クラスターを更新するには、ダウンタイムが必要です。更新には、自動再起動も含まれる可能性があります。ダウンタイムの時間は、以下のシナリオのように更新ペイロードによって異なります。
  - 更新ペイロードに再起動が必要なオペレーティングシステムの更新が含まれる場合には、ダウンタイムは、クラスター管理およびユーザーのワークロードに大きく影響します。
  - アップデートに含まれるマシン設定の変更で、再起動の必要がない場合には、ダウンタイムは少なくなり、クラスター管理およびユーザーワークロードへの影響は低くなります。この場合、クラスターに、ワークロードの再スケジューリングするノードが他にないため、単一ノードの OpenShift Container Platform でノードのドレイン（解放）のステップが省略されます。
  - 更新ペイロードにオペレーティングシステムの更新またはマシン設定の変更が含まれていない場合は、API が短時間してすぐに解決します。



重要

更新パッケージのバグなどの制約があり、再起動後に単一ノードが再起動されないことがあります。この場合、更新は自動的にロールバックされません。

関連情報

- 再起動が必要なマシン設定の変更については、「Machine Config Operator について」を参照してください。

5.5. WEB コンソールを使用したクラスターの更新

更新が利用可能な場合、Web コンソールからクラスターを更新できます。

利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータルの エラータ のセクションを参照してください。

前提条件

- **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。
- すべての **MachineHealthCheck** リソースを一時停止します。

手順

1. Web コンソールから、**Administration → Cluster Settings** をクリックし、**Details** タブの内容を確認します。
2. 実稼働クラスターの場合、**Channel** が **stable-4.9** などの更新する必要のあるバージョンの正しいチャネルに設定されていることを確認します。

重要

実稼働クラスターの場合、stable-* または fast-* チャネルにサブスクライブする必要があります。

   - **Update Status** が **Updates Available** ではない場合、クラスターをアップグレードすることはできません。
   - **Select Channel** は、クラスターが実行されているか、または更新されるクラスターのバージョンを示します。

3. 更新するバージョンを選択し、**Save** をクリックします。
入力チャネルの **Update Status** が **Update to <product-version> in progress**切り替わり、Operator およびノードの進捗バーを監視して、クラスター更新の進捗を確認できます。

注記

バージョン 4.y から 4.(y+1) などの次のマイナーバージョンにクラスターをアップグレードする場合、新たな機能に依存するワークロードをデプロイする前にノードがアップグレードされていることを確認することが推奨されます。更新されていないワーカーノードを持つプールは **Cluster Settings** ページに表示されます。

4. 更新が完了し、Cluster Version Operator が利用可能な更新を更新したら、追加の更新が現在のチャネルで利用可能かどうかを確認します。
   - 更新が利用可能な場合は、更新ができなくなるまで、現在のチャネルでの更新を継続します。

- 利用可能な更新がない場合は、**Channel** を次のマイナーバージョンの stable-* または fast-* チャネルに切り替え、そのチャネルで必要なバージョンに更新します。

必要なバージョンに達するまで、いくつかの中間更新を実行する必要がある場合があります。

5.6. WEB コンソールを使用した更新サーバーの変更

更新サーバーの変更は任意です。OpenShift Update Service (OSUS)がローカルにインストールされ、設定されている場合は、更新時にローカルサーバーを使用できるようにサーバーの URL を **upstream** として設定する必要があります。

手順

1. **Administration → Cluster Settings** に移動し、**version** をクリックします。
2. **YAML** タブをクリックし、**upstream** パラメーター値を編集します。

   出力例

```
...
spec:
  clusterID: db93436d-7b05-42cc-b856-43e11ad2d31a
  upstream: '<update-server-url>' (1)
...
```

   (1) **<update-server-url>** 変数は、更新サーバーの URL を指定します。

   デフォルトの **upstream** は **https://api.openshift.com/api/upgrades_info/v1/graph** です。

3. **保存** をクリックします。

第6章 CLI を使用したマイナーバージョン内でのクラスターの更新

OpenShift CLI (**oc**) を使用して OpenShift Container Platform クラスターをマイナーバージョン内で更新するか、またはアップグレードすることができます。

6.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできること。RBAC の使用によるパーミッションの定義および適用を参照してください。
- アップグレードが失敗し、クラスターを直前の状態に復元する必要がある場合に最新の etcd バックアップがあること。
- Operator Lifecycle Manager (OLM) で以前にインストールされたすべての Operator が、最新チャネルの最新バージョンに更新されていることを確認します。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターのアップグレード時に現行のマイナーバージョンから次のマイナーバージョンに切り替わる際、確実に有効なアップグレードパスがあるようにします。詳細は、「インストールされた Operator のアップグレード」を参照してください。
- すべてのマシン設定プール (MCP) が実行中であり、一時停止していないことを確認します。一時停止した MCP に関連付けられたノードは、更新プロセス中にスキップされます。カナリアロールアウト更新ストラテジーを実行している場合は、MCP を一時停止することができます。
- クラスターで手動で保守される認証情報を使用する場合は、Cloud Credential Operator (CCO) がアップグレード可能な状態であることを確認します。「AWS」、「Azure」、または「GCP」の **手動で保守される認証情報を使用したクラスターのアップグレード**」を参照してください。
- クラスターが、AWS Secure Token Service (STS)で手動でメンテナンスされる認証情報を使用する場合は、アップグレードするリリースイメージから **ccoctl** ユーティリティーのコピーを取得し、これを使用して更新された認証情報を処理します。詳細は、**Upgrading an OpenShift Container Platform cluster configured for manual mode with STS**を参照してください。
- クラスターで次のマイナーバージョンへのアップグレードができるように、すべての **Upgradeable=False** 条件に対応してください。**oc adm upgrade**コマンドを実行して、すべての**Upgradeable=False**条件の出力と、マイナーバージョンアップグレードの準備に役立つ条件推論を実行できます。

重要

unsupportedConfigOverrides セクションを使用して Operator の設定を変更することはサポートされておらず、クラスターのアップグレードをブロックする可能性があります。クラスターをアップグレードする前に、この設定を削除する必要があります。

重要

Prometheus の割り当てられた PVC を使用してクラスターモニタリングを実行している場合、クラスターのアップグレード時に OOM による強制終了が生じる可能性があります。永続ストレージが Prometheus 用に使用される場合、Prometheus のメモリー使用量はクラスターのアップグレード時、およびアップグレードの完了後の数時間で 2 倍になります。OOM による強制終了の問題を回避するには、ワーカーノードで、アップグレード前に利用可能なメモリーのサイズを 2 倍にできるようにします。たとえば、推奨される最小ノード（RAM が 8 GB の 2 コア）でモニタリングを実行している場合は、メモリーを 16 GB に増やします。詳細は、BZ#1925061 を参照してください。

関連情報

- 「管理外の Operator のサポートポリシー」

6.2. MACHINEHEALTHCHECK リソースの一時停止

アップグレードプロセスで、クラスター内のノードが一時的に利用できなくなる可能性があります。ワーカーノードの場合、マシンのヘルスチェックにより、このようなノードは正常ではないと識別され、それらが再起動される場合があります。このようなノードの再起動を回避するには、クラスターを更新する前にすべての **MachineHealthCheck** リソースを一時停止します。

前提条件

- OpenShift CLI (**oc**) をインストールすること。

手順

1. 一時停止する利用可能なすべての **MachineHealthCheck** リソースを一覧表示するには、以下のコマンドを実行します。

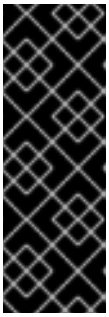
```
$ oc get machinehealthcheck -n openshift-machine-api
```

2. マシンヘルスチェックを一時停止するには、**cluster.x-k8s.io/paused=""** アノテーションを **MachineHealthCheck** リソースに追加します。以下のコマンドを実行します。

```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused=""
```

注釈付きの **MachineHealthCheck** リソースは以下の YAML ファイルのようになります。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
  annotations:
    cluster.x-k8s.io/paused: ""
spec:
  selector:
    matchLabels:
      role: worker
  unhealthyConditions:
  - type:    "Ready"
    status:  "Unknown"
    timeout: "300s"
  - type:    "Ready"
    status:  "False"
    timeout: "300s"
  maxUnhealthy: "40%"
status:
  currentHealthy: 5
  expectedMachines: 5
```



重要

クラスターの更新後にマシンヘルスチェックを再開します。チェックを再開するには、以下のコマンドを実行して **MachineHealthCheck** リソースから pause アノテーションを削除します。

```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused-
```

6.3. 単一ノードの OPENSHIFT CONTAINER PLATFORM の更新

コンソールまたは CLI のいずれかを使用して、単一ノードの OpenShift Container Platform クラスターを更新またはアップグレードできます。

ただし、以下の制限事項に注意してください。

- 他にヘルスチェックを実行するノードがないので、**MachineHealthCheck** リソースを一時停止する時に課される前提条件は必要ありません。
- etcd バックアップを使用した単一ノードの OpenShift Container Platform クラスターの復元は、正式にはサポートされていません。ただし、アップグレードに失敗した場合には、etcd バックアップを実行することが推奨されます。コントロールプレーンが正常である場合には、バックアップを使用してクラスターを以前の状態に復元できる場合があります。
- 単一ノードの OpenShift Container Platform クラスターを更新するには、ダウンタイムが必要です。更新には、自動再起動も含まれる可能性があります。ダウンタイムの時間は、以下のシナリオのように更新ペイロードによって異なります。
    - 更新ペイロードに再起動が必要なオペレーティングシステムの更新が含まれる場合には、ダウンタイムは、クラスター管理およびユーザーのワークロードに大きく影響します。
    - アップデートに含まれるマシン設定の変更で、再起動の必要がない場合には、ダウンタイムは少なくなり、クラスター管理およびユーザーワークロードへの影響は低くなります。この場合、クラスターに、ワークロードの再スケジューリングするノードが他にないため、単一ノードの OpenShift Container Platform でノードのドレイン（解放）のステップが省略されます。
    - 更新ペイロードにオペレーティングシステムの更新またはマシン設定の変更が含まれていない場合は、API が短時間してすぐに解決します。

重要

更新パッケージのバグなどの制約があり、再起動後に単一ノードが再起動されないことがあります。この場合、更新は自動的にロールバックされません。

関連情報

再起動が必要なマシン設定の変更については、「Machine Config Operator について」を参照してください。

6.4. CLI を使用したクラスターの更新

更新が利用可能な場合、OpenShift CLI (**oc**) を使用してクラスターを更新できます。

利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータルの エラータ のセクションを参照してください。

前提条件

- お使いの更新バージョンのバージョンに一致する OpenShift CLI (**oc**) をインストールします。
- **cluster-admin** 権限を持つユーザーとしてクラスターにログインします。
- **jq** パッケージをインストールします。
- すべての **MachineHealthCheck** リソースを一時停止します。

手順

1. クラスターが利用可能であることを確認します。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.8.13    True        False         158m    Cluster version is 4.8.13
```

2. 現在の更新チャネル情報を確認し、チャネルが **stable-4.9**に設定されていることを確認します。

```
$ oc get clusterversion -o json|jq ".items[0].spec"
```

出力例

```
{
  "channel": "stable-4.9",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff"
}
```

重要

実稼働クラスターの場合、**stable-*** または **fast-*** チャネルにサブスクライブする必要があります。

3. 利用可能な更新を確認し、適用する必要のある更新のバージョン番号をメモします。

```
$ oc adm upgrade
```

出力例

```
Cluster version is 4.8.13

Updates:
```

```
VERSION IMAGE
4.9.0   quay.io/openshift-release-dev/ocp-release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b
```

4. 更新を適用します。
   - 最新バージョンに更新するには、以下を実行します。

     ```
     $ oc adm upgrade --to-latest=true 1
     ```

   - 特定のバージョンに更新するには、以下を実行します。

     ```
     $ oc adm upgrade --to=<version> 1
     ```

     1 1 **<version>** は、直前のコマンドの出力から得られる更新バージョンです。

5. クラスターバージョン Operator を確認します。

   ```
   $ oc get clusterversion -o json|jq ".items[0].spec"
   ```

   出力例

   ```
   {
     "channel": "stable-4.9",
     "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff",
     "desiredUpdate": {
       "force": false,
       "image": "quay.io/openshift-release-dev/ocp-release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b",
       "version": "4.9.0" 1
     }
   }
   ```

   1 **desiredUpdate** スタンザの **version** 番号が指定した値と一致する場合、更新は進行中です。

6. クラスターバージョン履歴で、更新のステータスをモニターします。すべてのオブジェクトの更新が終了するまでに時間がかかる可能性があります。

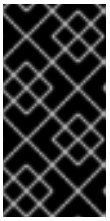
   ```
   $ oc get clusterversion -o json|jq ".items[0].status.history"
   ```

   出力例

   ```
   [
     {
       "completionTime": null,
       "image": "quay.io/openshift-release-dev/ocp-release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b7",
       "startedTime": "2021-01-28T20:30:50Z",
   ```

```
    "state": "Partial",
    "verified": true,
    "version": "4.9.0"
  },
  {
    "completionTime": "2021-01-28T20:30:50Z",
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b
7",
    "startedTime": "2021-01-28T17:38:10Z",
    "state": "Completed",
    "verified": false,
    "version": "4.8.13"
  }
]
```

履歴には、クラスターに適用された最新バージョンの一覧が含まれます。この値は、CVO が更新を適用する際に更新されます。この一覧は日付順に表示され、最新の更新は一覧の先頭に表示されます。履歴の更新には、ロールアウトが完了した場合には **Completed** と表示され、更新が失敗したか、または完了しなかった場合には **Partial** と表示されます。



重要

アップグレードに失敗する場合、Operator は停止し、失敗しているコンポーネントのステータスを報告します。クラスターの以前のバージョンへのロールバックはサポートされていません。アップグレードできない場合は、Red Hat サポートにお問い合わせください。

7. 更新が完了したら、クラスターのバージョンが新たなバージョンに更新されていることを確認できます。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION     AVAILABLE   PROGRESSING   SINCE     STATUS
version   4.9.0       True        False         2m        Cluster version is 4.9.0
```

8. バージョン 4.y から 4.(y+1) などの次のマイナーバージョンにクラスターをアップグレードする場合、新たな機能に依存するワークロードをデプロイする前にノードがアップグレードされていることを確認することが推奨されます。

```
$ oc get nodes
```

出力例

```
NAME                         STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.22.1
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.22.1
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.22.1
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.22.1
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.22.1
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.22.1
```

6.5. CLI を使用した更新サーバーの変更

更新サーバーの変更は任意です。OpenShift Update Service (OSUS)がローカルにインストールされ、設定されている場合は、更新時にローカルサーバーを使用できるようにサーバーの URL を **upstream** として設定する必要があります。**upstream** のデフォルト値は **https://api.openshift.com/api/upgrades_info/v1/graph** です。

手順

- クラスターバージョンで **upstream** パラメーター値を変更します。

```
$ oc patch clusterversion/version --patch '{"spec":{"upstream":"<update-server-url>"}}' --type=merge
```

<update-server-url> 変数は、更新サーバーの URL を指定します。

出力例

```
clusterversion.config.openshift.io/version patched
```

第7章 カナリアロールアウト更新の実行

更新プロセスによってアプリケーションが失敗した場合でも、更新全体を通じてミッションクリティカルなアプリケーションを利用できるようにするために、ワーカーノードへの更新のより制御されたロールアウトが必要なシナリオがいくつかある場合があります。組織のニーズによっては、ワーカーノードの小規模なサブセットを更新し、一定期間でクラスターおよびワークロードの正常性を評価し、残りのノードを更新する必要が生じる場合があります。通常、これは **カナリア** 更新と呼ばれます。または、クラスター全体を一度に更新するために大きなメンテナンスウィンドウを使用できない場合は、ホストの再起動が必要になることが多いワーカーノードの更新を、定義済みの小さなメンテナンスウィンドウに収めることもできます。

これらのシナリオでは、複数のカスタムマシン構成プール (MCP) を作成して、クラスターを更新するときに特定のワーカーノードが更新されないようにすることができます。残りのクラスターが更新されたら、それらのワーカーノードをバッチで随時更新できます。

たとえば、クラスターに 10% 超過する容量が 100 個あるクラスターがあり、4 時間を超えないようにメンテナンスウィンドウがあり、ワーカーノードをドレイン (解放) および再起動するのに 8 分未満になったことが分かっている場合は、MCP を使用して目標を達成できます。たとえば、**workerpool-canary**、**workerpool-A**、**workerpool-B**、および **workerpool-C** という名前の 4 つの MCP を、それぞれ 10、30、30、30ノードで定義できます。

最初のメンテナンス期間中、**workerpool-A**、**workerpool-B**、および **workerpool-C** の MCP を一時停止してから、クラスターの更新を開始します。これにより、プールが一時停止されていないため、OpenShift Container Platform の上部で実行されるコンポーネントと **workerpool-canary** MCP のメンバーである 10 ノードが更新されます。他の 3 つの MCP は一時停止されているため、更新されません。何らかの理由で、クラスターまたはワークロードの正常性が **workerpool-canary** 更新によって悪影響を受けると判断すると、問題を診断するまで十分な容量を維持しながら、そのプールのすべてのノードを遮断およびドレイン (解放) します。すべてが期待どおりに機能している場合は、一時停止を解除することを決定する前にクラスターおよびワークロードの状態を評価し、追加のメンテナンスウィンドウごとに **workerpool-A**、**workerpool-B**、および **workerpool-C** を連続して更新します。

カスタム MCP を使用してワーカーノードの更新を管理する一方で、複数のコマンドを実行する必要がある時間のかかるプロセスがある場合があります。この複雑さにより、クラスター全体に影響を与える可能性のあるエラーが発生する場合があります。組織のニーズを考慮し、開始する前にプロセスの実装を慎重に検討することが推奨されます。

注記

MCP を異なる OpenShift Container Platform バージョンに更新することは推奨されません。たとえば、ある MCP を 4.y.10 から 4.y.11 に更新せず、もう 1 つの MCP を 4.y.12 に更新しないでください。このシナリオはテストされておらず、未定義のクラスターの状態になる可能性があります。

重要

マシン設定プールを一時停止にすると、Machine Config Operator が関連付けられたノードに設定変更を適用できなくなります。MCP を一時停止することにより、**kube-apiserver-to-kubelet-signer** CA 証明書の自動 CA ローテーションを含め、自動的にローテーションされる証明書が関連付けられたノードにプッシュされないようにします。MCP が **kube-apiserver-to-kubelet-signer** CA 証明書の期限が切れ、MCO が証明書を自動的に更新しようとすると、新規証明書が作成されますが、適切なマシン設定プールのノード全体では適用されません。これにより、**oc debug**、**oc logs**、**oc exec**、**oc attach** など、複数の **oc** コマンドで問題が発生します。MCP の一時停止は、**kube-apiserver-to-kubelet-signer** CA 証明書の有効期限を慎重に考慮して、短期間のみ行う必要があります。

7.1. カナリアロールアウト更新プロセスおよび MCP について

OpenShift Container Platform では、ノードを個別に考慮しません。ノードはマシン設定プール (MCP) にグループ化されます。デフォルトの OpenShift Container Platform クラスターには 2 つの MCP があります。1 つはコントロールプレーンノード用であり、もう 1 つはワーカーノードになります。OpenShift Container Platform の更新は、すべての MCP を同時に影響します。

更新中、Machine Config Operator (MCO) は、MCP 内のすべてのノードを、指定された **maxUnavailable** ノード数 (指定されている場合) (デフォルトでは **1**) までドレイン (解放) および遮断します。ノードがドレイン (解放) および遮断し、ノード上のすべての Pod のスケジュールを解除し、ノードをスケジュール対象外としてマークします。ノードがドレイン (解放) されると、Machine Config Daemon は新規マシン設定を適用します。これには、オペレーティングシステム (OS) の更新を含めることができます。OS を更新するには、ホストを再起動する必要があります。

特定のノードが更新されないようにするために、つまり、ドレイン (解放)、遮断、および更新されないようにするために、カスタム MCP を作成できます。次に、これらの MCP を一時停止して、それらの MCP に関連付けられたノードが更新されないようにします。MCO は一時停止された MCP を更新しません。1 つ以上のカスタム MCP を作成して、それらのノードを更新するシーケンスをより詳細に制御できます。最初の MCP でノードを更新した後に、アプリケーションの互換性を確認し、残りのノードを新規バージョンに段階的に更新できます。

注記

コントロールプレーンの安定性を確保するには、コントロールプレーンノードからカスタム MCP の作成はサポートされません。Machine Config Operator (MCO) は、コントロールプレーンノード用に作成されるカスタム MCP を無視します。

ワークロードのデプロイメントトポロジーに基づいて、作成する MCP の数および各 MCP のノード数について慎重に考慮する必要があります。たとえば、更新を特定のメンテナンスウィンドウに合わせる必要がある場合は、OpenShift Container Platform がウィンドウ内で更新できるノードの数を把握しておく必要があります。この数は、一意のクラスターおよびワークロードの特性によって異なります。

また、クラスターで利用可能な容量の数を考慮する必要があります。たとえば、アプリケーションが更新されたノードで予想通りに機能しない場合は、プール内のそれらのノードを遮断およびドレイン (解放) できます。これにより、アプリケーション Pod を他のノードに移動します。必要なカスタム MCP の数および各 MCP のノード数を判別するために、利用可能な追加容量を考慮する必要があります。たとえば、ノードが各プールにある 2 つのカスタム MCP と 50% を使用する場合は、ノードの 50% が実行されているかどうかを判断する必要があります。この場合は、アプリケーション用に十分な QoS (quality-of-service) が提供されます。

この更新プロセスは、文書化されたすべての OpenShift Container Platform 更新プロセスで使用できます。ただし、このプロセスは、Ansible Playbook を使用して更新される Red Hat Enterprise Linux (RHEL) マシンでは機能しません。

7.2. カナリアロールアウト更新の実行について

このトピックでは、このカナリアロールアウト更新プロセスの一般的なワークフローについて説明します。ワークフローで各タスクを実行する手順は、以下のセクションで説明します。

1. ワーカープールに基づいて MCP を作成します。各 MCP のノード数は、各 MCP のメンテナンス期間や予約容量 (つまりクラスターで利用可能な追加のワーカーノード) など、いくつかの要素に依存します。

注記

MCP の **maxUnavailable** 設定を変更して、任意の時点で更新できるパーセンテージまたはマシン数を指定できます。デフォルトでは 1 回です。

2. ノードセレクターをカスタム MCP に追加します。残りのクラスターと同時に更新しない各ノードに、一致するラベルをノードに追加します。このラベルは、ノードを MCP に関連付けます。

注記

ノードからデフォルトのワーカーラベルを削除しないでください。クラスターで適切に機能するには、ノードにロールラベルが **必要** です。

3. 更新プロセスの一部として更新しない MCP を一時停止します。

注記

MCP を一時停止すると、kube-apiserver-to-kubelet-signer 自動 CA 証明書のローテーションも一時停止します。新しい CA 証明書は、インストール日と古い証明書の 292 日で生成され、インストール日から 365 日は削除されます。次の自動 CA 証明書のローテーションまでの所要時間については、「Understanding CA cert auto updates in Red Hat OpenShift 4」を参照してください。CA 証明書のローテーションが行われると、プールが一時停止されていないことを確認します。MCP が一時停止すると、証明書のローテーションが発生しません。これにより、クラスターは劣化し、複数の **oc** コマンドで失敗の原因となります。これには、**oc debug**、**oc logs**、**oc exec**、および **oc attach** が含まれますが、これに限定されません。

4. クラスターの更新を実行します。更新プロセスでは、コントロールプレーンノードを含む、一時停止されない MCP を更新します。

5. 更新されたノードでアプリケーションをテストし、想定通りに機能していることを確認します。

6. 残りの MCP を 1 つずつ一時停止解除し、すべてのワーカーノードが更新されるまでそれらのノードでアプリケーションをテストします。MCP の一時停止を解除すると、その MCP に関連付けられたノードの更新プロセスが開始されます。**Administration → Cluster settings** をクリックして、Web コンソールから更新の進捗を確認できます。または、**oc get machineconfigpools** CLI コマンドを使用します。

7. 必要に応じて、更新されたノードからカスタムラベルを削除し、カスタム MCP を削除します。

7.3. カナリアロールアウト更新を実行するためのマシン設定プールの作成

このカナリアロールアウト更新を実行する最初のタスクは、1 つ以上のマシン設定プール (MCP) を作成することです。

1. ワーカーノードから MCP を作成します。

   a. クラスターのワーカーノードを一覧表示します。

```
$ oc get -l 'node-role.kubernetes.io/master!=' -o 'jsonpath={range .items[*]}
{.metadata.name}{"\n"}{end}' nodes
```

出力例

```
ci-ln-pwnll6b-f76d1-s8t9n-worker-a-s75z4
ci-ln-pwnll6b-f76d1-s8t9n-worker-b-dglj2
ci-ln-pwnll6b-f76d1-s8t9n-worker-c-lldbm
```

b. 遅延させるノードの場合は、カスタムラベルをノードに追加します。

```
$ oc label node <node name> node-role.kubernetes.io/<custom-label>=
```

以下は例になります。

```
$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-
role.kubernetes.io/workerpool-canary=
```

出力例

```
node/ci-ln-gtrwm8t-f76d1-spbl7-worker-a-xk76k labeled
```

c. 新規 MCP を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: workerpool-canary (1)
spec:
  machineConfigSelector:
    matchExpressions: (2)
      - {
        key: machineconfiguration.openshift.io/role,
        operator: In,
        values: [worker,workerpool-canary]
        }
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/workerpool-canary: "" (3)
```

(1) MCP の名前を指定します。

(2) **worker** およびカスタム MCP 名を指定します。

(3) このプールで必要なノードに追加したカスタムラベルを指定します。

```
$ oc create -f <file_name>
```

出力例

```
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary created
```

d. クラスター内の MCP およびそれらの現在の状態を表示します。

```
$ oc get machineconfigpool
```

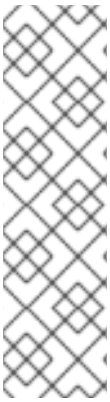
出力例

```
NAME              CONFIG                                             UPDATED   UPDATING
DEGRADED   MACHINECOUNT   READYMACHINECOUNT
UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
master            rendered-master-b0bb90c4921860f2a5d8a2f8137c1867            True
False      False      3            3                3                   0                   97m
workerpool-canary rendered-workerpool-canary-87ba3dec1ad78cb6aecebf7fbb476a36
True       False      False      1            1                1                   0                   2m42s
worker            rendered-worker-87ba3dec1ad78cb6aecebf7fbb476a36            True
False      False      2            2                2                   0                   97m
```

新規マシン設定プールの **workerpool-canary** が作成され、カスタムラベルが追加されたノード数がマシン数に表示されます。ワーカー MCP マシン数は同じ数で縮小されます。マシン数の更新に数分かかることがあります。この例では、1 つのノードが **worker** MCP から **workerpool-canary** MCP に移動しました。

7.4. マシン設定プールの一時停止

このカナリアロールアウト更新プロセスでは、OpenShift Container Platform クラスターの残りの部分で更新しないノードにラベルを付け、マシン設定プール (MCP) を作成し、それらの MCP を一時停止します。MCP を一時停止にすると、Machine Config Operator (MCO) がその MCP に関連付けられたノードを更新できなくなります。



注記

MCP を一時停止すると、kube-apiserver-to-kubelet-signer 自動 CA 証明書のローテーションも一時停止します。新しい CA 証明書は、インストール日と古い証明書の 292 日で生成され、インストール日から 365 日は削除されます。次の自動 CA 証明書のローテーションまでの所要時間については、「Understanding CA cert auto updates in Red Hat OpenShift 4」を参照してください。CA 証明書のローテーションが行われると、プールが一時停止されていないことを確認します。MCP が一時停止すると、証明書のローテーションが発生しません。これにより、クラスターは劣化し、複数の **oc** コマンドで失敗の原因となります。これには、**oc debug**、**oc logs**、**oc exec**、および **oc attach** が含まれますが、これに限定されません。

MCP を一時停止するには、以下を実行します。

1. 一時停止する MCP にパッチを適用します。

```
$ oc patch mcp/<mcp_name> --patch '{"spec":{"paused":true}}' --type=merge
```

以下は例になります。

```
$  oc patch mcp/workerpool-canary --patch '{"spec":{"paused":true}}' --type=merge
```

出力例

```
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
```

7.5. クラスターの更新の実行

MCP が ready 状態に入ると、クラスターの更新が可能になります。クラスターに合わせて、以下の更新方法のいずれかを参照してください。

- クラスターのマイナーバージョン間の更新
- CLI の使用によるマイナーバージョン内でのクラスターの更新

更新が完了したら、MCP の 1 回の一時停止を解除することができます。

7.6. マシン設定プールの一時停止の解除

このカナリアロールアウト更新プロセスでは、OpenShift Container Platform の更新が完了した後にカスタム MCP の一時停止を 1 つずつ解除します。MCP の一時停止を解除すると、Machine Config Operator(MCO)はその MCP に関連付けられたノードを更新できます。

MCP の一時停止を解除するには、以下を実行します。

1. 一時停止を解除する MCP にパッチを適用します。

```
$ oc patch mcp/<mcp_name> --patch '{"spec":{"paused":false}}' --type=merge
```

以下は例になります。

```
$  oc patch mcp/workerpool-canary --patch '{"spec":{"paused":false}}' --type=merge
```

出力例

```
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
```

oc get machineconfigpools コマンドを使用して更新の進捗を確認できます。

2. 更新されたノードでアプリケーションをテストし、想定通りに機能していることを確認します。
3. 一時停止した他の MCP の一時停止を解除すると、1 回目でアプリケーションが機能することを確認します。

7.6.1. アプリケーション障害発生時

更新されたノードでアプリケーションが機能しないなどの障害が発生した場合は、プール内のノードを遮断してドレイン (解放) できます。これにより、アプリケーション Pod が他のノードに移動され、アプリケーションのサービス品質を維持できます。この最初の MCP は追加の容量よりも大きくすることはできません。

7.7. ノードを元のマシン設定プールに移行

このカナリアロールアウト更新プロセスでは、カスタムマシン設定プール (MCP) の一時停止を解除し、その MCP に関連付けられたノード上のアプリケーションが期待どおりに機能していることを確認した後、ノードに追加したカスタムラベルを削除して、元の MCP に戻す必要があります。

重要

ノードには、クラスター内で適切に機能するロールが必要です。

ノードを元の MCP に移動するには、以下を実行します。

1. ノードからカスタムラベルを削除します。

```
$ oc label node <node_name> node-role.kubernetes.io/<custom-label>-
```

以下は例になります。

```
$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-
role.kubernetes.io/workerpool-canary-
```

出力例

```
node/ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz labeled
```

MCO は、ノードを元の MCP に戻し、ノードを MCP 設定に調整します。

2. クラスター内の MCP およびそれらの現在の状態を表示します。

```
$oc get mcp
```

```
NAME              CONFIG                                        UPDATED   UPDATING
DEGRADED   MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
master            rendered-master-1203f157d053fd987c7cbd91e3fbc0ed        True      False
False      3            3                   3                   0                   61m
workerpool-canary   rendered-mcp-noupdate-5ad4791166c468f3a35cd16e734c9028   True
False      False      0            0                   0                   0                   21m
worker            rendered-worker-5ad4791166c468f3a35cd16e734c9028        True      False
False      3            3                   3                   0                   61m
```

ノードはカスタム MCP から削除され、元の MCP に戻ります。マシン数の更新に数分かかることがあります。この例では、1 つのノードが削除された **workerpool-canary** MCP から 'worker'MCP に移動しました。

3. 必要に応じて、カスタム MCP を削除します。

```
$ oc delete mcp <mcp_name>
```

第8章 RHEL コンピュートマシンを含むクラスターの更新

OpenShift Container Platform クラスターの更新またはアップグレードを実行できます。クラスターに Red Hat Enterprise Linux (RHEL) マシンが含まれる場合は、それらのマシンを更新するために追加の手順を実行する必要があります。

8.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできること。RBAC の使用によるパーミッションの定義および適用を参照してください。
- アップグレードが失敗し、クラスターを直前の状態に復元する必要がある場合に最新の etcd バックアップがあること。
- クラスターで手動で保守される認証情報を使用する場合は、Cloud Credential Operator (CCO) がアップグレード可能な状態であることを確認します。「AWS」、「Azure」、または「GCP」の **手動で保守される認証情報を使用したクラスターのアップグレード**」を参照してください。
- クラスターが、AWS Secure Token Service (STS)で手動でメンテナンスされる認証情報を使用する場合は、アップグレードするリリースイメージから **ccoctl** ユーティリティーのコピーを取得し、これを使用して更新された認証情報を処理します。詳細は、**Upgrading an OpenShift Container Platform cluster configured for manual mode with STS**を参照してください。

重要

Prometheus の割り当てられた PVC を使用してクラスターモニタリングを実行している場合、クラスターのアップグレード時に OOM による強制終了が生じる可能性があります。永続ストレージが Prometheus 用に使用される場合、Prometheus のメモリー使用量はクラスターのアップグレード時、およびアップグレードの完了後の数時間で 2 倍になります。OOM による強制終了の問題を回避するには、ワーカーノードで、アップグレード前に利用可能なメモリーのサイズを 2 倍にできるようにします。たとえば、推奨される最小ノード（RAM が 8 GB の 2 コア）でモニタリングを実行している場合は、メモリーを 16 GB に増やします。詳細は、BZ#1925061 を参照してください。

関連情報

- 「管理外の Operator のサポートポリシー」

8.2. WEB コンソールを使用したクラスターの更新

更新が利用可能な場合、Web コンソールからクラスターを更新できます。

利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータルの エラータ のセクションを参照してください。

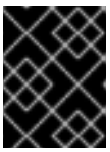
前提条件

- **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。
- すべての **MachineHealthCheck** リソースを一時停止します。

手順

1. Web コンソールから、**Administration → Cluster Settings** をクリックし、**Details** タブの内容を確認します。

2. 実稼働クラスターの場合、**Channel** が **stable-4.9** などの更新する必要のあるバージョンの正しいチャネルに設定されていることを確認します。

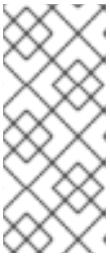


重要

実稼働クラスターの場合、stable-* または fast-* チャネルにサブスクライブする必要があります。

- **Update Status** が **Updates Available** ではない場合、クラスターをアップグレードすることはできません。
- **Select Channel** は、クラスターが実行されているか、または更新されるクラスターのバージョンを示します。

3. 更新するバージョンを選択し、**Save** をクリックします。
入力チャネルの **Update Status** が **Update to <product-version> in progress**切り替わり、Operator およびノードの進捗バーを監視して、クラスター更新の進捗を確認できます。



注記

バージョン 4.y から 4.(y+1) などの次のマイナーバージョンにクラスターをアップグレードする場合、新たな機能に依存するワークロードをデプロイする前にノードがアップグレードされていることを確認することが推奨されます。更新されていないワーカーノードを持つプールは **Cluster Settings** ページに表示されます。

4. 更新が完了し、Cluster Version Operator が利用可能な更新を更新したら、追加の更新が現在のチャネルで利用可能かどうかを確認します。

- 更新が利用可能な場合は、更新ができなくなるまで、現在のチャネルでの更新を継続します。
- 利用可能な更新がない場合は、**Channel** を次のマイナーバージョンの stable-* または fast-* チャネルに切り替え、そのチャネルで必要なバージョンに更新します。

必要なバージョンに達するまで、いくつかの中間更新を実行する必要がある場合があります。

注記

Red Hat Enterprise Linux (RHEL) ワーカーマシンを含むクラスターを更新する場合、それらのワーカーは、更新プロセス時に一時的に使用できなくなります。クラスターの更新の終了において各 RHEL マシンがのステートが **NotReady** になる際に、アップグレード Playbook を各 RHEL マシンに対して実行する必要があります。

8.3. オプション: RHEL マシンで ANSIBLE タスクを実行するためのフックの追加

OpenShift Container Platform の更新時に **フック** を使用し、RHEL コンピュートマシンで Ansible タスクを実行できます。

8.3.1. アップグレード用の Ansible Hook について

OpenShift Container Platform の更新時に **フック** を使用し、特定操作の実行中に Red Hat Enterprise Linux (RHEL) ノードでカスタムタスクを実行できます。フックを使用して、特定の更新タスクの前後に実行するタスクを定義するファイルを指定できます。OpenShift Container Platform クラスターで RHEL コンピュートノードを更新する際に、フックを使用してカスタムインフラストラクチャーを検証したり、変更したりすることができます。

フックが失敗すると操作も失敗するため、フックはべき等性があるか、または複数回実行でき、同じ結果を出せるように設計する必要があります。

フックには以下のような重要な制限があります。まず、フックには定義された、 またはバージョン付けされたインターフェースがありません。フックは内部の **openshift-ansible** 変数を使用できますが、これらの変数は今後の OpenShift Container Platform のリリースで変更されるか、または削除される予定です。次に、フックにはエラー処理機能がないため、フックにエラーが生じると更新プロセスが中止されます。エラーの発生時には、まず問題に対応してからアップグレードを再び開始する必要があります。

8.3.2. Ansible インベントリーファイルでのフックを使用する設定

Red Hat Enterprise Linux (RHEL) コンピュートマシン (ワーカーマシンとしても知られている) の更新時に使用するフックを、**all:vars** セクションの下にある **hosts** インベントリーファイルで定義します。

前提条件

- RHEL コンピュートマシンクラスターの追加に使用したマシンへのアクセスがあること。RHEL マシンを定義する **hosts** Ansible インベントリーファイルにアクセスできる必要があります。

手順

1. フックの設計後に、フック用に Ansible タスクを定義する YAML ファイルを作成します。このファイルは、以下に示すように一連のタスクで構成される必要があり、Playbook にすることはできません。

```
---
# Trivial example forcing an operator to acknowledge the start of an upgrade
# file=/home/user/openshift-ansible/hooks/pre_compute.yml

- name: note the start of a compute machine update
  debug:
    msg: "Compute machine upgrade of {{ inventory_hostname }} is about to start"

- name: require the user agree to start an upgrade
  pause:
    prompt: "Press Enter to start the compute machine update"
```

2. **hosts** Ansible インベントリーファイルを変更してフックファイルを指定します。フックファイルは、以下に示すように **[all:vars]** セクションのパラメーター値として指定されます。

インベントリーファイルのフック定義の例

```
[all:vars]
openshift_node_pre_upgrade_hook=/home/user/openshift-ansible/hooks/pre_node.yml
openshift_node_post_upgrade_hook=/home/user/openshift-ansible/hooks/post_node.yml
```

フックへのパスでの曖昧さを避けるために、それらの定義では相対パスの代わりに絶対パスを使用します。

8.3.3. RHEL コンピュートマシンで利用できるフック

Red Hat Enterprise Linux (RHEL) コンピュートマシンを OpenShift Container Platform クラスターで更新する際に、以下のフックを使用できます。

フック名	説明
openshift_node_pre_cordon_hook	• 各ノードの遮断 (cordon) **前** に実行されます。 • このフックは **各ノード** に対して連続して実行されます。 • タスクが異なるホストに対して実行される必要がある場合、そのタスクは **delegate_to** または **local_action** を使用する必要があります。
openshift_node_pre_upgrade_hook	• 各ノードの遮断 (cordon) **後**、更新 **前** に実行されます。 • このフックは **各ノード** に対して連続して実行されます。 • タスクが異なるホストに対して実行される必要がある場合、そのタスクは **delegate_to** または **local_action** を使用する必要があります。
openshift_node_pre_uncordon_hook	• 各ノードの更新 **後**、遮断の解除 (uncordon) **前に** 実行されます。 • このフックは **各ノード** に対して連続して実行されます。 • タスクが異なるホストに対して実行される必要がある場合、そのタスクは **delegate_to** または **local_action** を使用する必要があります。

フック名	説明
openshift_node_post_upgrade_hook	• 各ノードの遮断の解除 (uncordon) **後** に実行されます。これは、**最後の** ノード更新アクションになります。 • このフックは **各ノード** に対して連続して実行されます。 • タスクが異なるホストに対して実行される必要がある場合、そのタスクは **delegate_to** または **local_action** を使用する必要があります。

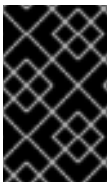
8.4. クラスター内の RHEL コンピュートマシンの更新

クラスターの更新後は、クラスター内の Red Hat Enterprise Linux (RHEL) コンピュートマシンを更新する必要があります。

重要

Red Hat Enterprise Linux(RHEL)バージョン 7.9 およびバージョン 8.4 は、RHEL ワーカー（コンピュート）マシンでサポートされます。

RHEL をオペレーティングシステムとして使用する場合は、コンピュートマシンを別の OpenShift Container Platform のマイナーバージョンに更新することもできます。マイナーバージョンの更新の実行時に、RHEL から RPM パッケージを除外する必要はありません。

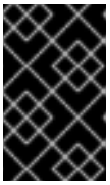


重要

RHEL 7 コンピュートマシンを RHEL 8 にアップグレードすることはできません。新しい RHEL 8 ホストをデプロイする必要があり、古い RHEL 7 ホストを削除する必要があります。

前提条件

- クラスターが更新されていること。

  重要

  RHEL マシンには、更新プロセスを完了するためにクラスターで生成されるアセットが必要になるため、クラスターを更新してから、クラスター内の RHEL ワーカーマシンを更新する必要があります。

- RHEL コンピュートマシンクラスターの追加に使用したマシンへのローカルアクセスがあること。RHEL マシンを定義する **hosts** Ansible インベントリーファイルおよび **upgrade** Playbook にアクセスできる必要があります。
- マイナーバージョンへの更新の場合、RPM リポジトリーはクラスターで実行しているのと同じバージョンの OpenShift Container Platform を使用します。

手順

1. ホストで firewalld を停止し、無効にします。

```
# systemctl disable --now firewalld.service
```

注記

デフォルトでは、「最小」インストールオプションを持つベース OS RHEL により、firewalld スパッドが有効になります。ホストで firewalld サービスを有効にすると、ワーカーで OpenShift Container Platform ログにアクセスできなくなります。ワーカーの OpenShift Container Platform ログへのアクセスを継続する場合は、firewalld を後で有効にしないでください。

2. OpenShift Container Platform 4.9 で必要なリポジトリーを有効にします。

   a. Ansible Playbook を実行するマシンで、必要なリポジトリーを更新します。

```
# subscription-manager repos --disable=rhel-7-server-ose-4.8-rpms \
                           --enable=rhel-7-server-ansible-2.9-rpms \
                           --enable=rhel-7-server-ose-4.9-rpms
```

   b. Ansible Playbook を実行するマシンで、**openshift-ansible** を含む必要なパッケージを更新します。

```
# yum update openshift-ansible openshift-clients
```

   c. 各 RHEL コンピュートノードで、必要なリポジトリーを更新します。

```
# subscription-manager repos --disable=rhel-7-server-ose-4.8-rpms \
                           --enable=rhel-7-server-ose-4.9-rpms  \
                           --enable=rhel-7-fast-datapath-rpms   \
                           --enable=rhel-7-server-optional-rpms
```

3. RHEL ワーカーマシンを更新します。

   a. 現在のノードステータスを確認し、更新する RHEL ワーカーを判別します。

```
# oc get node
```

   出力例

```
NAME                    STATUS                     ROLES    AGE    VERSION
mycluster-control-plane-0   Ready                      master   145m   v1.22.1
mycluster-control-plane-1   Ready                      master   145m   v1.22.1
mycluster-control-plane-2   Ready                      master   145m   v1.22.1
mycluster-rhel7-0           NotReady,SchedulingDisabled   worker   98m    v1.22.1
mycluster-rhel7-1           Ready                      worker   98m    v1.22.1
mycluster-rhel7-2           Ready                      worker   98m    v1.22.1
mycluster-rhel7-3           Ready                      worker   98m    v1.22.1
```

   ステータスが **NotReady,SchedulingDisabled** のマシンに留意してください。

   b. /**path_/inventory/hosts** で Ansible インベントリーファイルを確認し、以下の例に示され

b. **/<path>/inventory/hosts** で Ansible インベントリーファイルを確認し、以下の例に示されるように、ステータスが **NotReady,SchedulingDisabled** のマシンのみが **[workers]** セクションに一覧表示されるようにそのコンテンツを更新します。

```
[all:vars]
ansible_user=root
#ansible_become=True

openshift_kubeconfig_path="~/.kube/config"

[workers]
mycluster-rhel7-0.example.com
```

c. **openshift-ansible** ディレクトリーに移動します。

```
$ cd /usr/share/ansible/openshift-ansible
```

d. **upgrade** Playbook を実行します。

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/upgrade.yml 1
```

1 **<path>** については、作成した Ansible インベントリーファイルへのパスを指定します。

注記

upgrade Playbook は OpenShift Container Platform パッケージのみをアップグレードします。オペレーティングシステムパッケージは更新されません。

4. 直前の手順で実行したプロセスに従って、クラスター内の各 RHEL ワーカーマシンを更新します。

5. すべてのワーカーを更新したら、すべてのクラスターノードが新規バージョンに更新されていることを確認します。

```
# oc get node
```

出力例

```
NAME                    STATUS                        ROLES    AGE    VERSION
mycluster-control-plane-0   Ready                         master   145m   v1.22.1
mycluster-control-plane-1   Ready                         master   145m   v1.22.1
mycluster-control-plane-2   Ready                         master   145m   v1.22.1
mycluster-rhel7-0           NotReady,SchedulingDisabled   worker   98m    v1.22.1
mycluster-rhel7-1           Ready                         worker   98m    v1.22.1
mycluster-rhel7-2           Ready                         worker   98m    v1.22.1
mycluster-rhel7-3           Ready                         worker   98m    v1.22.1
```

6. オプション: **upgrade** Playbook で更新されていないオペレーティングシステムパッケージを更新します。4.9 にないパッケージを更新するには、以下のコマンドを使用します。

```
# yum update
```

注記

4.9 のインストール時に使用したものと同じ RPM リポジトリーを使用している場合は、RPM パッケージを除外する必要はありません。

第9章 ネットワークが制限された環境でのクラスターの更新

oc コマンドラインインターフェース (CLI) を使用してネットワークが制限された OpenShift Container Platform クラスターをアップグレードできます。

ネットワークが制限された環境とは、クラスターノードがインターネットにアクセスできない環境のことです。このため、レジストリーにはインストールイメージを設定する必要があります。レジストリーホストがインターネットとクラスターの両方にアクセスできない場合、その環境から切断されたファイルシステムにイメージをミラーリングし、そのホストまたはリムーバブルメディアを非接続環境に置きます。ローカルコンテナーレジストリーとクラスターがミラーレジストリーのホストに接続されている場合、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できます。

複数のクラスターがネットワークが制限されたネットワークに存在する場合、必要なリリースイメージを単一のコンテナーイメージレジストリーにミラーリングし、そのレジストリーを使用してすべてのクラスターを更新します。

9.1. 前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがあること。
- イメージをプッシュおよびプルするために、ネットワークが制限された環境でコンテナーレジストリーへの書き込みアクセスがあること。コンテナーレジストリーは Docker レジストリー API v2 と互換性がある必要があります。
- **oc** コマンドツールインターフェース (CLI) ツールがインストールされていること。
- **admin** 権限を持つユーザーとしてクラスターにアクセスできること。RBAC の使用によるパーミッションの定義および適用を参照してください。
- アップグレードが失敗し、クラスターを直前の状態に復元する必要がある場合に最新の etcd バックアップがあること。
- すべてのマシン設定プール (MCP) が実行中であり、一時停止していないことを確認します。一時停止した MCP に関連付けられたノードは、更新プロセス中にスキップされます。カナリアロールアウト更新ストラテジーを実行している場合は、MCP を一時停止することができます。
- クラスターで手動で保守される認証情報を使用する場合は、Cloud Credential Operator (CCO) がアップグレード可能な状態であることを確認します。「AWS」、「Azure」、または「GCP」の **手動で保守される認証情報を使用したクラスターのアップグレード**」を参照してください。

重要

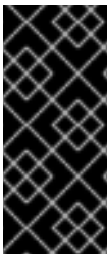
Prometheus の割り当てられた PVC を使用してクラスターモニタリングを実行している場合、クラスターのアップグレード時に OOM による強制終了が生じる可能性があります。永続ストレージが Prometheus 用に使用される場合、Prometheus のメモリー使用量はクラスターのアップグレード時、およびアップグレードの完了後の数時間で 2 倍になります。OOM による強制終了の問題を回避するには、ワーカーノードで、アップグレード前に利用可能なメモリーのサイズを 2 倍にできるようにします。たとえば、推奨される最小ノード（RAM が 8 GB の 2 コア）でモニタリングを実行している場合は、メモリーを 16 GB に増やします。詳細は、BZ#1925061 を参照してください。

9.2. 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。

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

コマンドラインインターフェースを使用して 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 4.9 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。ネットワークが制限された環境でクラスターをアップグレードする場合は、アップグレードする予定の **oc** バージョンをインストールします。

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.9 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
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
```

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

```
$ oc <command>
```

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。

3. **OpenShift v4.9 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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>
```

macOC 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. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.9 MacOSX 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. 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
```

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>
```

9.3. 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。

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_in_json> (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": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHAtqXs=",
      "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"
    }
  }
}
```

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

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のクラスターをアップグレードする前に、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。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。

手順

1. Red Hat OpenShift Container Platform Upgrade Graph visualizer および update planner を使用して、あるバージョンから別のバージョンへのアップグレードを計画します。OpenShift Upgrade Graph はチャネルのグラフと、現行バージョンと意図されるクラスターのバージョン間に更新パスがあることを確認する方法を提供します。

2. 必要な環境変数を設定します。

   a. リリースバージョンをエクスポートします。

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

   <release_version> について、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: **4.5.4**)。

   b. ローカルレジストリー名とホストポートをエクスポートします。

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

   <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。

   c. ローカルリポジトリー名をエクスポートします。

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

   <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。

   d. ミラーリングするリポジトリーの名前をエクスポートします。

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

   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。

   e. パスをレジストリープルシークレットにエクスポートします。

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

   <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の絶対パスおよびファイル名を指定します。

注記

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

f. リリースミラーをエクスポートします。

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

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。

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

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

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

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

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

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

4. 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. イメージおよび設定マニフェスト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}
```

  iii. 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} 1
```

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

- ローカルコンテナーレジストリーとクラスターがミラーホストに接続されている場合、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、以下のコマンドを使用して設定マップをクラスターに適用します。

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

注記

--apply-release-image-signature オプションが含まれる場合は、イメージ署名の検証用に設定マップを作成しません。

9.5. イメージ署名設定マップの作成

クラスターを更新する前に、使用するリリースイメージの署名が含まれる設定マップを手動で作成する必要があります。この署名により、Cluster Version Operator (CVO) では、予想されるイメージと実際のイメージの署名を比較することでリリースイメージが変更されていないことを確認できます。

バージョン 4.4.8 以降からアップグレードする場合は、**oc** CLI を使用して設定マップを作成できます。以前のバージョンからアップグレードする場合は、手動の方法を使用する必要があります。

9.5.1. oc CLI の使用によるイメージ署名の検証用の設定マップの作成

oc CLI を使用してイメージ署名設定マップの作成を単純化します。

前提条件

- OpenShift CLI (**oc**) をインストールすること。

手順

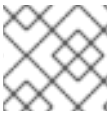
1. mirror.openshift.com または Google Cloud Storage (GCS) のいずれかからアップグレードするバージョンのイメージ署名を取得します。
2. **oc** コマンドラインインターフェース (CLI) を使用して、アップグレードしているクラスターにログインします。
3. ミラーリングされたリリースイメージ署名設定マップを接続されたクラスターに適用します。

```
$ oc apply -f <image_signature_file> 1
```

1 **<image_signature_file>** について、ファイルのパスおよび名前を指定します (例: **mirror/config/signature-sha256-81154f5c03294534.yaml**)。

9.5.2. イメージ署名設定マップの手動での作成

イメージ署名設定マップを作成し、更新するクラスターに適用します。



注記

クラスターを更新するたびに以下の手順を実行する必要があります。

手順

1. OpenShift Container Platform アップグレードパス についてのナレッジベースの記事を参照し、クラスターの有効なパスを判別します。
2. バージョンを **OCP_RELEASE_NUMBER** 環境変数に追加します。

```
$ OCP_RELEASE_NUMBER=<release_version> 1
```

1 **<release_version>** について、クラスターを更新する OpenShift Container Platform のバージョンに対応するタグを指定します (例: **4.4.0**)。

3. クラスターのシステムアーキテクチャーを **ARCHITECTURE** 環境変数に追加します。

```
$ ARCHITECTURE=<server_architecture> 1
```

1 **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。

4. Quay からリリースイメージダイジェストを取得します。

```
$ DIGEST="$(oc adm release info quay.io/openshift-release-dev/ocp-release:${OCP_RELEASE_NUMBER}-${ARCHITECTURE} | sed -n 's/Pull From: .*@//p')"
```

5. ダイジェストアルゴリズムを設定します。

```
$ DIGEST_ALGO="${DIGEST%%:*}"
```

6. ダイジェスト署名を設定します。

```
$ DIGEST_ENCODED="${DIGEST#*:}"
```

7. イメージ署名を mirror.openshift.com Web サイトから取得します。

```
$ SIGNATURE_BASE64=$(curl -s "https://mirror.openshift.com/pub/openshift-v4/signatures/openshift/release/${DIGEST_ALGO}=${DIGEST_ENCODED}/signature-1" | base64 -w0 && echo)
```

8. 設定マップを作成します。

```
$ cat >checksum-${OCP_RELEASE_NUMBER}.yaml <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: release-image-${OCP_RELEASE_NUMBER}
```

```
  namespace: openshift-config-managed
  labels:
    release.openshift.io/verification-signatures: ""
binaryData:
  ${DIGEST_ALGO}-${DIGEST_ENCODED}: ${SIGNATURE_BASE64}
EOF
```

9. 設定マップをクラスターに適用し、更新します。

```
$ oc apply -f checksum-${OCP_RELEASE_NUMBER}.yaml
```

9.6. MACHINEHEALTHCHECK リソースの一時停止

アップグレードプロセスで、クラスター内のノードが一時的に利用できなくなる可能性があります。ワーカーノードの場合、マシンのヘルスチェックにより、このようなノードは正常ではないと識別され、それらが再起動される場合があります。このようなノードの再起動を回避するには、クラスターを更新する前にすべての **MachineHealthCheck** リソースを一時停止します。

前提条件

- OpenShift CLI (**oc**) をインストールすること。

手順

1. 一時停止する利用可能なすべての **MachineHealthCheck** リソースを一覧表示するには、以下のコマンドを実行します。

```
$ oc get machinehealthcheck -n openshift-machine-api
```

2. マシンヘルスチェックを一時停止するには、**cluster.x-k8s.io/paused=""** アノテーションを **MachineHealthCheck** リソースに追加します。以下のコマンドを実行します。

```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused=""
```

注釈付きの **MachineHealthCheck** リソースは以下の YAML ファイルのようになります。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
  annotations:
    cluster.x-k8s.io/paused: ""
spec:
  selector:
    matchLabels:
      role: worker
  unhealthyConditions:
  - type:    "Ready"
    status:  "Unknown"
    timeout: "300s"
  - type:    "Ready"
    status:  "False"
```

```
    timeout: "300s"
  maxUnhealthy: "40%"
status:
  currentHealthy: 5
  expectedMachines: 5
```

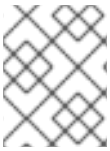
重要

クラスターの更新後にマシンヘルスチェックを再開します。チェックを再開するには、以下のコマンドを実行して **MachineHealthCheck** リソースから pause アノテーションを削除します。

```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused-
```

9.7. ネットワークが制限された環境のクラスターのアップグレード

ネットワークが制限された環境のクラスターを、ダウンロードしたリリースイメージの OpenShift Container Platform バージョンに更新します。



注記

ローカルの OpenShift Update Service がある場合は、この手順ではなく、接続された Web コンソールまたは CLI の手順を使用して更新できます。

前提条件

- 新規リリースのイメージをレジストリーに対してミラーリングしている。
- 新規リリースのリリースイメージ署名 ConfigMap をクラスターに適用している。
- イメージ署名 ConfigMap からリリースの sha256 合計値を取得している。
- OpenShift CLI (**oc**) をインストールすること。
- すべての **MachineHealthCheck** リソースを一時停止します。

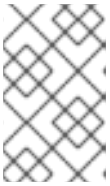
手順

- クラスターを更新します。

```
$ oc adm upgrade --allow-explicit-upgrade --to-image ${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}<sha256_sum_value> 1
```

1 **<sha256_sum_value>** 値は、イメージ署名 ConfigMap からのリリースの sha256 合計値です (例: **@sha256:81154f5c03294534e1eaf0319bef7a601134f891689ccede5d705ef659aa8c92**)。

ミラーレジストリーに **ImageContentSourcePolicy** を使用する場合、**LOCAL_REGISTRY** の代わりに正規レジストリー名を使用できます。



注記

ImageContentSourcePolicy オブジェクトを持つクラスターのグローバルプルシークレットのみを設定できます。プロジェクトにプルシークレットを追加することはできません。

9.8. イメージレジストリーのリポジトリーミラーリングの設定

コンテナーレジストリーのリポジトリーミラーリングの設定により、以下が可能になります。

- ソースイメージのレジストリーのリポジトリーからイメージをプルする要求をリダイレクトするように OpenShift Container Platform クラスターを設定し、これをミラーリングされたイメージレジストリーのリポジトリーで解決できるようにします。
- 各ターゲットリポジトリーに対して複数のミラーリングされたリポジトリーを特定し、1つのミラーがダウンした場合に別のミラーを使用できるようにします。

以下は、OpenShift Container Platform のリポジトリーミラーリングの属性の一部です。

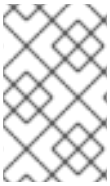
- イメージプルには、レジストリーのダウンタイムに対する回復性があります。
- ネットワークが制限された環境のクラスターは、重要な場所 (quay.io など) からイメージをプルでき、会社のファイアウォールの背後にあるレジストリーが要求されたイメージを提供するようにできます。
- イメージのプル要求時にレジストリーへの接続が特定の順序で試行され、通常は永続レジストリーが最後に試行されます。
- 入力したミラー情報は、OpenShift Container Platform クラスターの全ノードの **/etc/containers/registries.conf** ファイルに追加されます。
- ノードがソースリポジトリーからイメージの要求を行うと、要求されたコンテンツを見つけるまで、ミラーリングされた各リポジトリーに対する接続を順番に試行します。すべてのミラーで障害が発生した場合、クラスターはソースリポジトリーに対して試行します。成功すると、イメージはノードにプルされます。

リポジトリーミラーリングのセットアップは次の方法で実行できます。

- OpenShift Container Platform のインストール時:
  OpenShift Container Platform が必要とするコンテナーイメージをプルし、それらのイメージを会社のファイアウォールの内側に配置すると、制限されたネットワーク内にあるデータセンターに OpenShift Container Platform をインストールできます。
- OpenShift Container Platform の新規インストール後:
  OpenShift Container Platform インストール時にミラーリングを設定しなくても、**ImageContentSourcePolicy** オブジェクトを使用して後で設定することができます。

以下の手順では、インストール後のミラーを設定し、以下を識別する **ImageContentSourcePolicy** オブジェクトを作成します。

- ミラーリングするコンテナーイメージリポジトリーのソース
- ソースリポジトリーから要求されたコンテンツを提供する各ミラーリポジトリーの個別のエントリー。



注記

ImageContentSourcePolicy オブジェクトを持つクラスターのグローバルプルシークレットのみを設定できます。プロジェクトにプルシークレットを追加することはできません。

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があること。

手順

1. ミラーリングされたリポジトリーを設定します。以下のいずれかを実行します。
   - 「Repository Mirroring in Red Hat Quay」で説明されているように、Red Hat Quay でミラーリングされたリポジトリーを設定します。Red Hat Quay を使用すると、あるリポジトリーから別のリポジトリーにイメージをコピーでき、これらのリポジトリーを一定期間繰り返し自動的に同期することもできます。
   - **skopeo** などのツールを使用して、ソースディレクトリーからミラーリングされたリポジトリーにイメージを手動でコピーします。
     たとえば、Red Hat Enterprise Linux (RHEL 7 または RHEL 8) システムに skopeo RPM パッケージをインストールした後、以下の例に示すように **skopeo** コマンドを使用します。

     ```
     $ skopeo copy \
     docker://registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6 \
     docker://example.io/example/ubi-minimal
     ```

     この例では、**example.io** いう名前のコンテナーイメージレジストリーと **example** という名前のイメージリポジトリーがあり、そこに **registry.access.redhat.com** から **ubi8/ubi-minimal** イメージをコピーします。レジストリーを作成した後、OpenShift Container Platform クラスターを設定して、ソースリポジトリーで作成される要求をミラーリングされたリポジトリーにリダイレクトできます。
2. OpenShift Container Platform クラスターにログインします。
3. **ImageContentSourcePolicy** ファイル (例: **registryrepomirror.yaml**) を作成し、ソースとミラーを固有のレジストリー、およびリポジトリーのペアとイメージのものに置き換えます。

   ```
   apiVersion: operator.openshift.io/v1alpha1
   kind: ImageContentSourcePolicy
   metadata:
     name: ubi8repo
   spec:
     repositoryDigestMirrors:
     - mirrors:
       - example.io/example/ubi-minimal (1)
       source: registry.access.redhat.com/ubi8/ubi-minimal (2)
     - mirrors:
       - example.com/example/ubi-minimal
       source: registry.access.redhat.com/ubi8/ubi-minimal
   ```

```
- mirrors:
  - mirror.example.com/redhat
  source: registry.redhat.io/openshift4 3
```

1 イメージレジストリーおよびリポジトリーの名前を示します。

2 ミラーリングされているコンテンツが含まれるレジストリーおよびリポジトリーを示します。

3 レジストリー内の namespace を、その namespace の任意のイメージを使用するように設定できます。レジストリードメインをソースとして使用する場合、**ImageContentSourcePolicy** リソースはレジストリーからすべてのリポジトリーに適用されます。

4. 新しい **ImageContentSourcePolicy** オブジェクトを作成します。

```
$ oc create -f registryrepomirror.yaml
```

ImageContentSourcePolicy オブジェクトが作成されると、新しい設定が各ノードにデプロイされ、クラスターはソースリポジトリーへの要求のためにミラーリングされたリポジトリーの使用を開始します。

5. ミラーリングされた設定が適用されていることを確認するには、ノードのいずれかで以下を実行します。

   a. ノードの一覧を表示します。

```
$ oc get node
```

出力例

```
NAME                         STATUS                     ROLES    AGE  VERSION
ip-10-0-137-44.ec2.internal    Ready                      worker   7m   v1.22.1
ip-10-0-138-148.ec2.internal   Ready                      master   11m  v1.22.1
ip-10-0-139-122.ec2.internal   Ready                      master   11m  v1.22.1
ip-10-0-147-35.ec2.internal    Ready,SchedulingDisabled   worker   7m   v1.22.1
ip-10-0-153-12.ec2.internal    Ready                      worker   7m   v1.22.1
ip-10-0-154-10.ec2.internal    Ready                      master   11m  v1.22.1
```

変更が適用されているため、各ワーカーノードのスケジューリングが無効にされていることを確認できます。

   b. デバッグプロセスを開始し、ノードにアクセスします。

```
$ oc debug node/ip-10-0-147-35.ec2.internal
```

出力例

```
Starting pod/ip-10-0-147-35ec2internal-debug ...
To use host binaries, run `chroot /host`
```

   c. ノードのファイルにアクセスします。

```
sh-4.2# chroot /host
```

d. **/etc/containers/registries.conf** ファイルをチェックして、変更が行われたことを確認します。

```
sh-4.2# cat /etc/containers/registries.conf
```

出力例

```
unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
[[registry]]
  location = "registry.access.redhat.com/ubi8/"
  insecure = false
  blocked = false
  mirror-by-digest-only = true
  prefix = ""

  [[registry.mirror]]
    location = "example.io/example/ubi8-minimal"
    insecure = false

  [[registry.mirror]]
    location = "example.com/example/ubi8-minimal"
    insecure = false
```

e. ソースからノードにイメージダイジェストをプルし、ミラーによって解決されているかどうかを確認します。**ImageContentSourcePolicy** オブジェクトはイメージダイジェストのみをサポートし、イメージタグはサポートしません。

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6
```

リポジトリーのミラーリングのトラブルシューティング

リポジトリーのミラーリング手順が説明どおりに機能しない場合は、リポジトリーミラーリングの動作方法についての以下の情報を使用して、問題のトラブルシューティングを行うことができます。

- 最初に機能するミラーは、プルされるイメージを指定するために使用されます。
- メインレジストリーは、他のミラーが機能していない場合にのみ使用されます。
- システムコンテキストによって、**Insecure** フラグがフォールバックとして使用されます。
- **/etc/containers/registries.conf** ファイルの形式が最近変更されました。現在のバージョンはバージョン 2 で、TOML 形式です。

9.9. クラスターノードの再起動の頻度を減らすために、ミラーイメージカタログの範囲を拡大

リポジトリーレベルまたはより幅広いレジストリーレベルでミラーリングされたイメージカタログのスコープを設定できます。幅広いスコープの **ImageContentSourcePolicy** リソースにより、リソースの変更に対応するためにノードが再起動する必要のある回数が減ります。

ImageContentSourcePolicy リソースのミラーイメージカタログの範囲を拡大するには、以下の手順を実行します。

前提条件

- OpenShift Container Platform CLI (**oc**) をインストールします。
- **cluster-admin** 権限を持つユーザーとしてログインすること。
- 非接続クラスターで使用するようにミラーリングされたイメージカタログを設定します。

手順

1. **<local_registry>**, **<pull_spec>**, and **<pull_secret_file>** の値を指定して、以下のコマンドを実行します。

```
$ oc adm catalog mirror <local_registry>/<pull_spec> <local_registry> -a <pull_secret_file> --icsp-scope=registry
```

ここでは、以下のようになります。

<local_registry>
: 非接続クラスター (例: **local.registry:5000**) 用に設定したローカルレジストリーです。

<pull_spec>
: 非接続レジストリーで設定されるプル仕様です (例: **redhat/redhat-operator-index:v4.9**)。

<pull_secret_file>
: Red Hat OpenShift Cluster Manager サイトの「Pull Secret」ページからダウンロードした **.json** ファイル形式の **registry.redhat.io** プルシークレットです。

oc adm catalog mirror コマンドは、**/redhat-operator-index-manifests** ディレクトリーを作成し、**imageContentSourcePolicy.yaml**、**catalogSource.yaml**、および **mapping.txt** ファイルを生成します。

2. 新しい **ImageContentSourcePolicy** リソースをクラスターに適用します。

```
$ oc apply -f imageContentSourcePolicy.yaml
```

検証

- **oc apply** が **ImageContentSourcePolicy** に変更を正常に適用していることを確認します。

```
$ oc get ImageContentSourcePolicy -o yaml
```

出力例

```
apiVersion: v1
items:
- apiVersion: operator.openshift.io/v1alpha1
  kind: ImageContentSourcePolicy
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |
```

```
{"apiVersion":"operator.openshift.io/v1alpha1","kind":"ImageContentSourcePolicy","metadata":
{"annotations":{},"name":"redhat-operator-index"},"spec":{"repositoryDigestMirrors":
[{"mirrors":["local.registry:5000"],"source":"registry.redhat.io"}]}}
...
```

ImageContentSourcePolicy リソースを更新した後に、OpenShift Container Platform は新しい設定を各ノードにデプロイし、クラスターはソースリポジトリーへの要求のためにミラーリングされたリポジトリーの使用を開始します。

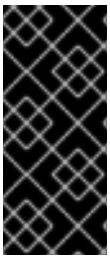
9.10. 関連情報

- ネットワークが制限された環境での Operator Lifecycle Manager の使用
- マシン設定の概要
- OpenShift Update Service のインストールと設定

第10章 VSPHERE で稼働するノードでのハードウェアの更新

vSphere で実行されているノードが OpenShift Container Platform でサポート対象のハードウェアバージョンで実行されていることを確認する必要があります。現時点で、ハードウェアバージョン 13 以降は、クラスター内の vSphere 仮想マシンでサポートされます。

仮想ハードウェアを直ちに更新したり、vCenter で更新をスケジュールしたりできます。



重要

vSphere で実行しているクラスターノード用にハードウェアバージョン 13 を使用することは非推奨となりました。これらのバージョンは引き続き完全にサポートされていますが、サポートは OpenShift Container Platform の今後のバージョンで削除されます。ハードウェアバージョン 15 が、OpenShift Container Platform の vSphere 仮想マシンのデフォルトになりました。

10.1. VSPHERE での仮想ハードウェアの更新

VMware vSphere 上の仮想マシンのハードウェアを更新するには、仮想マシンを個別に更新し、クラスターのダウンタイムのリスクを軽減します。

10.1.1. vSphere でのコントロールプレーンノードの仮想ハードウェアの更新

ダウンタイムのリスクを軽減するには、コントロールプレーンノードを順次アップグレードすることが推奨されます。これにより、Kubernetes API が利用可能な状態を保ち、etcd はクォーラム（定足数）を維持します。

前提条件

- OpenShift Container Platform クラスターをホストする vCenter インスタンスで必要なパーミッションを実行するためのクラスター管理者パーミッションがある。
- vSphere ESXi ホストがバージョン 6.7U3 以降を使用している。

手順

1. クラスターのコントロールプレーンノードを一覧表示します。

```
$ oc get nodes -l node-role.kubernetes.io/master
```

出力例

```
NAME                 STATUS   ROLES    AGE   VERSION
control-plane-node-0   Ready    master   75m   v1.22.1
control-plane-node-1   Ready    master   75m   v1.22.1
control-plane-node-2   Ready    master   75m   v1.22.1
```

コントロールプレーンノードの名前を書き留めておきます。

2. コントロールプレーンノードにスケジュール対象外(unschedulable)のマークを付けます。

```
$ oc adm cordon <control_plane_node>
```

3. コントロールプレーンノードに関連付けられた仮想マシンをシャットダウンします。仮想マシンを右クリックし、**Power → Shut Down Guest OS**を選択して、vSphere クライアントでこれを実行します。安全にシャットダウンされない場合があるため、**Power Off** を使用して仮想マシンをシャットダウンしないでください。

4. vSphere クライアントで仮想マシンをアップグレードします。詳細は、VMware ドキュメントの Upgrading a virtual machine to the latest hardware version を参照してください。

5. コントロールプレーンノードに関連付けられた仮想マシンの電源を入れます。仮想マシンを右クリックし、**Power On** を選択して、vSphere クライアントでこれを実行します。

6. ノードが **Ready** として報告されるまで待機します。

```
$ oc wait --for=condition=Ready node/<control_plane_node>
```

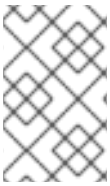
7. コントロールプレーンノードを再度スケジュール対象としてマークします。

```
$ oc adm uncordon <control_plane_node>
```

8. クラスター内のコントロールプレーンノードごとに、この手順を繰り返します。

10.1.2. vSphere でのコンピュートノードの仮想ハードウェア更新

ダウンタイムのリスクを軽減するには、コンピュートノードを順次アップグレードすることが推奨されます。



注記

ワークロードでは、**NotReady** の状態の複数のノードに対応できるという前提で、複数のコンピュートノードを並行してアップグレードできます。管理者が責任を持って、必要なコンピュートノードを利用できる状態にしてください。

前提条件

- OpenShift Container Platform クラスターをホストする vCenter インスタンスで必要なパーミッションを実行するためのクラスター管理者パーミッションがある。
- vSphere ESXi ホストがバージョン 6.7U3 以降を使用している。

手順

1. クラスターのコンピュートノードを一覧表示します。

```
$ oc get nodes -l node-role.kubernetes.io/worker
```

出力例

```
NAME           STATUS   ROLES    AGE   VERSION
compute-node-0   Ready    worker   30m   v1.22.1
compute-node-1   Ready    worker   30m   v1.22.1
compute-node-2   Ready    worker   30m   v1.22.1
```

コンピュートノードの名前を書き留めておきます。

2. コンピュートノードにスケジュール対象外 (unschedulable) のマークを付けます。

```
$ oc adm cordon <compute_node>
```

3. コンピュートノードから Pod を退避します。これにはいくつかの方法があります。たとえば、ノードですべてまたは選択した Pod を退避できます。

```
$ oc adm drain <compute_node> [--pod-selector=<pod_selector>]
```

ノードから Pod を退避させる方法は、「ノードの Pod を退避する方法」のセクションを参照してください。

4. コンピュートノードに関連付けられた仮想マシンをシャットダウンします。仮想マシンを右クリックし、**Power → Shut Down Guest OS**を選択して、vSphere クライアントでこれを実行します。安全にシャットダウンされない場合があるため、**Power Off** を使用して仮想マシンをシャットダウンしないでください。

5. vSphere クライアントで仮想マシンをアップグレードします。詳細は、VMware ドキュメントの Upgrading a virtual machine to the latest hardware version を参照してください。

6. コンピュートノードに関連付けられた仮想マシンの電源を入れます。仮想マシンを右クリックし、**Power On** を選択して、vSphere クライアントでこれを実行します。

7. ノードが **Ready** として報告されるまで待機します。

```
$ oc wait --for=condition=Ready node/<compute_node>
```

8. コンピュートノードを再度スケジュール対象としてマークします。

```
$ oc adm uncordon <compute_node>
```

9. クラスター内のコンピュートノードごとに、この手順を繰り返します。

関連情報

- ノード上の Pod を退避させる方法

10.2. VSPHERE での仮想ハードウェアの更新のスケジューリング

仮想マシンの電源がオンまたは再起動時に、仮想ハードウェアの更新をスケジュールできます。VMware ドキュメントの「仮想マシンの互換性アップグレードのスケジュール」に従い、仮想ハードウェアの更新だけを vCenter でスケジュールできます。

OpenShift Container Platform のアップグレード実行前に、アップグレードをスケジュールする場合には、OpenShift Container Platform のアップグレード中にノードが再起動されると、仮想ハードウェアが更新されます。