



OpenShift Container Platform 4.6

クラスタの更新

OpenShift Container Platform クラスタの更新

OpenShift Container Platform 4.6 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

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Updating_clusters.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

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 クラスターの更新の概要	4
1.1. OPENSIFT UPDATE SERVICE について	4
1.2. OPENSIFT UPDATE SERVICE のインストールと設定	4
1.3. アップグレードチャンネルとリリースを理解	4
1.4. WEB コンソールを使用してクラスターを更新	4
1.5. CLI を使用したクラスターの更新	5
1.6. RHEL コンピュータマシンを含むクラスターの更新	5
1.7. ネットワークが制限された環境でのクラスターの更新	5
第2章 OPENSIFT UPDATE SERVICE について	6
2.1. OPENSIFT UPDATE SERVICE について	6
2.2. 管理外の OPERATOR のサポートポリシー	7
第3章 OPENSIFT UPDATE SERVICE のインストールと設定	9
3.1. 前提条件	9
3.1.1. 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	9
3.1.2. グローバルクラスターのプルシークレットの更新	9
3.2. OPENSIFT UPDATE SERVICE のインストール	11
3.2.1. Web コンソールを使用した OpenShift Update Service Operator のインストール	11
3.2.2. CLI を使用した OpenShift Update Service Operator のインストール	12
3.2.3. OpenShift Update Service グラフデータコンテナイメージの作成	14
3.2.4. OpenShift Container Platform イメージリポジトリーのミラーリング	14
3.3. OPENSIFT UPDATE SERVICE アプリケーションの作成	17
3.3.1. 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	17
3.3.2. CLI を使用した OpenShift Update Service アプリケーションの作成	18
3.3.3. Cluster Version Operator (CVO) の設定	20
3.4. OPENSIFT UPDATE SERVICE アプリケーションの削除	21
3.4.1. 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	21
3.4.2. CLI を使用した OpenShift Update Service アプリケーションの削除	21
3.5. OPENSIFT UPDATE SERVICE OPERATOR のアンインストール	22
3.5.1. Web コンソールを使用した OpenShift Update Service Operator のアンインストール	22
3.5.2. CLI を使用した OpenShift Update Service Operator のアンインストール	22
第4章 アップグレードチャンネルとリリースを理解	24
4.1. チャンネルおよびリリースパスのアップグレード	24
4.1.1. candidate-4.6 チャンネル	24
4.1.2. fast-4.6 チャンネル	24
4.1.3. stable-4.6 チャンネル	25
4.1.4. EUS-4.y チャンネル	25
4.1.5. アップグレードバージョンパス	25
4.1.6. 高速かつ安定したチャンネルの使用およびストラテジー	26
4.1.7. ネットワークが制限された環境のクラスター	26
4.1.8. CLI プロファイル間の切り替え	26
第5章 WEB コンソールを使用してクラスターを更新	28
5.1. 前提条件	28
5.2. WEB コンソールを使用したクラスターの更新	28
5.3. WEB コンソールを使用した更新サーバーの変更	29
第6章 CLI を使用したクラスターの更新	31
6.1. 前提条件	31
6.2. 関連情報	31

6.3. CLI を使用したクラスターの更新	31
6.4. CLI を使用した更新サーバーの変更	34
第7章 RHEL コンピュータマシンを含むクラスターの更新	36
7.1. 前提条件	36
7.2. WEB コンソールを使用したクラスターの更新	36
7.3. 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	37
7.3.1. アップグレード用の Ansible Hook について	37
7.3.2. Ansible インベントリファイルでのフックを使用する設定	37
7.3.3. RHEL コンピュータマシンで利用できるフック	38
7.4. クラスター内の RHEL コンピュータマシンの更新	39
第8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	43
8.1. 前提条件	43
8.2. ミラーホストの準備	43
8.2.1. バイナリーのダウンロードによる OpenShift CLI のインストール	43
8.2.1.1. Linux への OpenShift CLI のインストール	44
8.2.1.2. Windows への OpenShift CLI のインストール	44
8.2.1.3. macOS への OpenShift CLI のインストール	45
8.3. イメージのミラーリングを可能にする認証情報の設定	45
8.4. OPENSIFT CONTAINER PLATFORM イメージリポジトリのミラーリング	48
8.5. イメージ署名設定マップの作成	50
8.5.1. イメージ署名設定マップの手動での作成	50
8.6. 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	51
8.7. イメージレジストリのリポジトリミラーリングの設定	52
8.8. 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	56
8.9. 関連情報	57

第1章 クラスターの更新の概要

Web コンソールまたは OpenShift CLI (oc) を使用して、1回の操作で OpenShift Container Platform 4 クラスターを更新できます。

1.1. OPENSIFT UPDATE SERVICE について

[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 Update Service を介して OTA (over-the-air) 更新を提供します。

1.2. OPENSIFT UPDATE SERVICE のインストールと設定

[OpenShift Update Service のインストールおよび設定](#): 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。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。詳細については、次の手順を参照してください。

- [OpenShift Update Service のインストール](#)
- [OpenShift Update Service アプリケーションの作成](#)
- [OpenShift Update Service アプリケーションの削除](#)
- [OpenShift Update Service Operator のアンインストール](#)

1.3. アップグレードチャンネルとリリースを理解

[アップグレードチャンネルとリリースの理解](#): アップグレードチャンネルを使用すると、アップグレード戦略を選択できます。アップグレードチャンネルは OpenShift Container Platform のマイナーバージョン固有のもので、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、常にそのマイナーバージョンをインストールします。マイナーバージョンに加えて、**candidate**、**fast**、**stable**、および **eus** チャンネル層から選択して、環境のニーズに合った更新戦略を有効にすることができます。詳細は、以下を参照してください。

- [チャンネルおよびリリースパスのアップグレード](#)
- [バージョンパスのアップグレード](#)
- [高速かつ安定したチャンネルの使用およびストラテジーの理解](#)
- [制限されたネットワーククラスターの理解](#)
- [CLI プロファイル間の切り替え](#)

1.4. WEB コンソールを使用してクラスターを更新

[Web コンソールを使用したクラスターの更新](#): Web コンソールを使用して OpenShift Container Platform クラスターを更新できます。次の手順は、マイナーバージョン内の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も、同じ手順を使用できます。

- [Web コンソールを使用したクラスターの更新](#)

- [Web コンソールを使用した更新サーバーの変更](#)

1.5. CLI を使用したクラスターの更新

CLI を使用したクラスターの更新: OpenShift CLI (oc) を使用して、マイナーバージョン内で OpenShift Container Platform クラスターを更新できます。次の手順は、マイナーバージョン内の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も、同じ手順を使用できます。

- [CLI を使用したクラスターの更新](#)
- [CLI を使用した更新サーバーの変更](#)

1.6. RHEL コンピュータマシンを含むクラスターの更新

RHEL コンピュータマシンを含むクラスターの更新: クラスターに Red Hat Enterprise Linux (RHEL) マシンが含まれている場合は、追加の手順を実行してそれらのマシンを更新する必要があります。次のセクションを確認できます。

- [Web コンソールを使用したクラスターの更新](#)
- [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](#)
- [クラスター内の RHEL コンピュータマシンの更新](#)

1.7. ネットワークが制限された環境でのクラスターの更新

制限付きネットワーククラスターの更新: 制限付きネットワーク環境とは、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です。OpenShift CLI (oc) を使用して、制限されたネットワーククラスターを更新できます。制限されたネットワーク環境では、ローカルの OpenShift Update Service インスタンスを使用して更新を受け取るように OpenShift Container Platform クラスターを設定できます。ミラ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は、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。詳細は以下のリンクをご覧ください。

- [ミラーホストの準備](#)
- [イメージのミラーリングを可能にする認証情報の設定](#)
- [OpenShift Container Platform イメージリポジトリーのミラーリング](#)
- [イメージ署名設定マップの作成](#)
- [制限付きネットワーククラスターの更新](#)
- [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](#)
- [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](#)

第2章 OPENSIFT 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せん。[OpenShift Update Service のインストールと設定](#) を参照してください。

2.1. OPENSIFT 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更新します。リリースアーティファクトは、コンテナイメージとして Quay でホストされます。

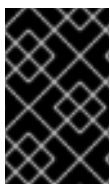
OpenShift Update Service が互換性のある更新のみを提供できるようにするために、リリース検証 Pipeline で自動化を支援します。それぞれのリリースアーティファクトについて、他のコンポーネントパッケージだけでなくサポートされているクラウドプラットフォームおよびシステムアーキテクチャーとの互換性の有無が検証されます。Pipeline がリリースの適合性を確認した後に、OpenShift Update Service は更新が利用可能であることを通知します。



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 manifests を絶えず更新し、その manifests 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。更新が失敗した場合は、Red Hat サポートに連絡してください。

更新プロセスで、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設定されます。

2.2. 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、Cluster 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せん。

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3章 OPENSIFT 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更新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。

3.1. 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。

3.1.1. 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。

3.1.2. グローバルクラスターのプルシークレットの更新

現在のプルシークレットを置き換えるか、新しいプルシークレットを追加することで、クラスターのグローバルプルシークレットを更新できます。

ユーザーがインストール中に使用したレジストリーとは別のレジストリーを使用してイメージを保存する場合は、この手順が必要です。



警告

クラスターリソースは新規のプルシークレットに合わせて調整する必要がありますが、これにより、クラスターのユーザービリティが一時的に制限される可能性があります。



警告

グローバルプルシークレットを更新すると、Machine Config Operator (MCO) が変更を同期している間にノードが再起動します。

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。

手順

1. オプション: 既存のプルシークレットに新しいプルシークレットを追加するには、以下の手順を実行します。
 - a. 以下のコマンドを入力してプルシークレットをダウンロードします。

```
$ oc get secret/pull-secret -n openshift-config --template='{{index .data ".dockerconfigjson" | base64decode}}' ><pull_secret_location> 1
```

- 1 プルシークレットファイルへのパスを指定します。

- b. 以下のコマンドを実行して、新しいプルシークレットを追加します。

```
$ oc registry login --registry="<registry>" \ 1
--auth-basic="<username>:<password>" \ 2
--to=<pull_secret_location> 3
```

- 1 新しいレジストリーを指定します。同じレジストリー内に複数のリポジトリを含めることができます (例: **--registry="<registry/my-namespace/my-repository>"**)。
- 2 新しいレジストリーの認証情報を指定します。

- 3 プルシークレットファイルへのパスを指定します。

または、プルシークレットファイルを手動で更新することもできます。

2. 以下のコマンドを実行して、クラスターのグローバルプルシークレットを更新します。

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=  
<pull_secret_location> 1
```

- 1 新規プルシークレットファイルへのパスを指定します。

この更新はすべてのノードにロールアウトされます。これには、クラスターのサイズに応じて多少時間がかかる場合があります。この間に、ノードがドレイン (解放) され、Pod は残りのノードで再スケジューリングされます。

3.2. OPENSIFT 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

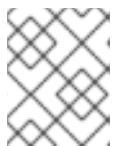
ネットワークが制限された環境 (非接続クラスターとして知られる) にインストールされているクラスターの場合には、デフォルトで Operator Lifecycle Manager はリモートレジストリーでホストされる Red Hat が提供する OperatorHub ソースにアクセスできません。それらのリモートソースには完全なインターネット接続が必要であるためです。詳細は、[ネットワークが制限された環境での Operator Lifecycle Manager の使用](#) を参照してください。

3.2.1. 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ンネルであるため、チャンネル **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。

3.2.2. 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
```

以下に例を示します。

```
$ 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
```

以下に例を示します。

```
$ 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" ❶
  sourceNamespace: "openshift-marketplace"
  name: "cincinnati-operator"

```

- ❶ 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
```

以下に例を示します。

```
$ 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。

3.2.3. OpenShift Update Service グラフデータコンテナイメージの作成

OpenShift Update Service には、OpenShift Update Service がチャン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 コンテナからこの情報を読み込むことができます。init コンテナのロールとして、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 コンテナイメージを OpenShift Update Service がアクセスできるリポジトリにコピーします。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。

3.2.4. OpenShift Container Platform イメージリポジトリのミラーリング

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 からプルシークレット](#) をダウンロードし、ミラーリポジトリへの認証を含める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、更新する必要がある 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の名前を指定します。

- d. 追加のローカルリポジトリ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の名前を指定します。

- e. ミラーリングするリポジトリ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> ❶
```

- ❶ 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ミラ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}/${RELEASE_NAME}:${OC_P_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} ❶
```

- ❶ **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}
```

3.3. OPENSIFT 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。

3.3.1. 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イメージの作成で作成した graph-data コンテナ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 (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。

3.3.2. 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イメージの作成で作成した graph-data コンテナ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。 **チャンネル** 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 4.6 で実行している場合は、 **stable-4.6** 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、63 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。

3.3.3. 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を有効にして、更新サーバーを信頼するように CA を設定するを参照してください。

3.4. OPENSIFT 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。

3.4.1. 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。

3.4.2. 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
```

3.5. OPENSIFT 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。

3.5.1. 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。

3.5.2. 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
```

- 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
```

- 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
```

- 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
```

- 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
```

- 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4章 アップグレードチャンネルとリリースを理解

OpenShift Container Platform 4.1 で、Red Hat はクラスターの更新の適切なリリースバージョンを推奨するためにチャンネルという概念を導入しました。これらの更新チャンネルでは、更新のペースを制御することで、更新戦略を選択できます。アップグレードチャンネルは 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ンネルでは、4.9 への更新と 4.9 以内の更新が推奨されます。また、4.8 内および 4.8 から 4.9 への更新を推奨し、4.8 上のクラスターが最終的に 4.9 に更新できるようにします。4.10 以降のリリースへの更新は推奨していません。この戦略により、管理者は OpenShift Container Platform の次のマイナーバージョンへの更新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ン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 4.6 は以下のアップグレードチャンネルを提供します。

- **candidate-4.6**
- **fast-4.6**
- **stable-4.6**
- **eus-4.y** (4.6 などの偶数番号の 4.y クラスターリリースを実行している場合のみ)

4.1. チャンネルおよびリリースパスのアップグレード

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ンネルを設定できます。

4.1.1. candidate-4.6 チャンネル

candidate-4.6 チャンネルには、z-stream (4.6.z) リリースの候補となるビルドとそれ以前のマイナーバージョンのリリースが含まれます。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、候補チャンネルで利用可能なビルドを指します。候補チャン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は、これは **fast-4.6** または **stable-4.6** チャンネルにプロモートされます。この戦略により、特定のリリースが **candidate-4.6** チャンネルと **fast-4.6** または **stable-4.6** チャン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-4.6** チャンネルには、いずれのチャンネルでも推奨されていないリリースバージョンを含めることができます。

candidate-4.6 チャンネルを使用して、OpenShift Container Platform の直前のマイナーバージョンから更新できます。

4.1.2. fast-4.6 チャンネル

fast-4.6 チャンネルは、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 4.6 の新規およびそれ以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-4.6** チャン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-4.6** チャンネルに表示されてからしばらくすると、**stable-4.6** チャンネルに追加されます。リリースは **fast-4.6** チャンネルに表示される前に、**stable-4.6** チャンネルに表示されることはありません。

fast-4.6 チャンネルを使用して、OpenShift Container Platform の以前のマイナーバージョンからの更新を実行できます。

4.1.3. stable-4.6 チャンネル

fast-4.6 チャン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-4.6** チャン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-4.6 チャンネルを使用して、OpenShift Container Platform の以前のマイナーバージョンからの更新を実行できます。

4.1.4. EUS-4.y チャンネル

stable チャンネルのほかに、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 \(延長更新サポート\)](#) (EUS) を提供します。EUS 4.6 チャンネルは、Premium サブスクリプションのメンテナンスフェーズを 14 カ月に拡張します。

OpenShift Container Platform 4.6 が EUS フェーズに移行するまで **stable-4.6** と **eus-4.6** チャンネル間に相違はありませんが、EUS チャンネルが利用可能になり次第、これに切り換えることができます。OpenShift Container Platform 4.6 がライフサイクルの EUS フェーズに移行すると、**stable-4.6** チャンネルは後続の z-stream 更新を受信しなくなります。次の EUS バージョンは 4.8 で、そのバージョンの更新には 4.6 から 4.7、そして 4.8 への連続した更新が必要です。



注記

標準サブスクライバーと非 EUS サブスクライバーの両方が、すべての EUS リポジトリと必要な RPM (**rhel-*-eus-rpms**) にアクセスして、ドライバーのデバッグやビルドなどの重要な目的をサポートできます。

4.1.5. 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ンネル内のパスの確認を可能にする更新推奨サービスが提供されます。

fast-4.6 チャンネルでは以下を確認できます。

- 4.6.0
- 4.6.1
- 4.6.3
- 4.6.4

このサービスは、テスト済みの重大な問題のない更新のみを推奨します。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。たとえば、クラスターが 4.6.1 にあり、OpenShift Container Platform が 4.6.4 を提案している場合、4.6.1 から 4.6.4 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 4.6.2 はチャン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はありません。

更新の安定性は、チャンネルによって異なります。**candidate-4.6** チャン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-4.6** または **stable-4.6** チャン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。リリースがチャン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ン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-4.6** または **stable-4.6** チャンネルのサポートされるリリースから 4.6.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。

4.1.6. 高速かつ安定したチャンネルの使用およびストラテジー

fast-4.6 および **stable-4.6** チャンネルでは、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更新は **fast-4.6** および **stable-4.6** チャンネルの両方でブロックされ、新たに推奨される更新先の新規バージョンが導入される可能性があります。

fast-4.6 チャンネルで実稼働前のシステムを設定し、**stable-4.6** チャン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-4.6** から **stable-4.6** チャン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。

4.1.7. 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、更新への影響に関するコメントに留意する必要があります。更新時に、インターフェイ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。

4.1.8. CLI プロファイル間の切り替え

チャンネルは、Web コンソールまたは **patch** コマンドを使用して切り替えることができます。

```
$ oc patch clusterversion version --type json -p [{"op": "add", "path": "/spec/channel", "value": "<channel>"}]
```

Web コンソールは、現在のリリースを含まないチャン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ン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ンネルに戻ることができます。

チャンネルの変更は、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-4.6** チャンネルから **fast-4.6** チャン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-4.6** チャンネルに切り換えることはできますが、このチャン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-4.6** チャンネルから **fast-4.6** チャンネルに切り換えることができます。

- **fast-4.6** チャンネルから **stable-4.6** チャン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-4.6** 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5章 WEB コンソールを使用してクラスターを更新

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。次の手順は、マイナーバージョン内の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も、同じ手順を使用できます。



注記

oc を使用して更新チャンネルを変更するのが容易ではないため、Web コンソールを使用して更新チャンネルを変更します。Web コンソール内で更新プロセスを完了することが推奨されます。4.6 チャンネルに変更した後、[CLI を使用してクラスターを更新](#) 手順に従って、更新を完了することができます。

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。
- 更新が失敗し、[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、最新チャン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、確実に有効な更新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。
- 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

- 更新が完了しなかった場合、Cluster Version Operator (CVO) は、更新の調整を試みている間、ブロック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。更新が完了しない場合は、Red Hat サポートに連絡してください。
- **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、クラスターの更新をブロックする可能性があります。クラスターを更新する前に、この設定を削除する必要があります。

関連情報

- [管理外の Operator のサポートポリシー](#)

5.2. 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。

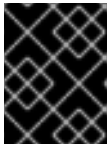
利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータル の [エラータ](#) のセクションを参照してください。

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。

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. 本番クラスターの場合は、**チャンネル** が、**stable-4.6** など、更新するバージョンの正しいチャン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-*** チャン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、クラスターを更新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。
入力チャンネルの **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更新する場合、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ン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、現在のチャン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-*** チャンネルに切り替え、そのチャン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。

5.3. 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. **Save** をクリックします。

第6章 CLI を使用したクラスタの更新

OpenShift CLI (**oc**) を使用して OpenShift Container Platform クラスタ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。同じ手順に従って、マイナーバージョン間でクラスタを更新することもできます。

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。
- 更新が失敗し、[クラスタ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、最新チャン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、確実に有効な更新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。
- クラスタ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のアップグレード](#) を参照してください。



重要

- 更新が完了しなかった場合、Cluster Version Operator (CVO) は、更新の調整を試みている間、ブロック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。更新が完了しない場合は、Red Hat サポートに連絡してください。
- **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、クラスタの更新をブロックする可能性があります。クラスタを更新する前に、この設定を削除する必要があります。

6.2. 関連情報

- [管理外の Operator のサポートポリシー](#)

6.3. CLI を使用したクラスタの更新

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にログインします。

- **jq** パッケージをインストールします。

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

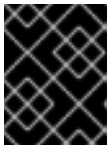
```
NAME      VERSION AVAILABLE PROGRESSING SINCE STATUS
version  4.5.4   True      False      158m Cluster version is 4.5.4
```

2. 現在の更新チャンネル情報を確認し、チャンネルが **stable-4.6** 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-4.6",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff",
  "upstream": "https://api.openshift.com/api/upgrades_info/v1/graph"
}
```



重要

実稼働クラスターの場合、**stable-*** または **fast-*** チャン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、適用する必要が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 4.1.0

Updates:

VERSION IMAGE
4.1.2 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
```

①①<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-4.6",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff",
  "desiredUpdate": {
    "force": false,
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b",

    "version": "4.6.3" ①
  },
  "upstream": "https://api.openshift.com/api/upgrades_info/v1/graph"
}
```

- 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@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b",

    "startedTime": "2020-11-10T20:30:50Z",
    "state": "Partial",
    "verified": true,
    "version": "4.1.2"
  },
  {
    "completionTime": "2020-11-10T20:30:50Z",
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:b8307ac0f3ec4ac86c3f3b52846425205022da52c16f56ec31cbe428501001d6
",
    "startedTime": "2020-11-10T17:38:10Z",
```

```

    "state": "Completed",
    "verified": false,
    "version": "4.1.0"
  }
]

```

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。

- 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  4.6.3    True       False        2m        Cluster version is 4.6.3

```

- 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  v1.19.0
ip-10-0-170-223.ec2.internal  Ready  master  82m  v1.19.0
ip-10-0-179-95.ec2.internal   Ready  worker  70m  v1.19.0
ip-10-0-182-134.ec2.internal  Ready  worker  70m  v1.19.0
ip-10-0-211-16.ec2.internal   Ready  master  82m  v1.19.0
ip-10-0-250-100.ec2.internal  Ready  worker  69m  v1.19.0

```

6.4. 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 RHEL コンピュータ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。

7.1. 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。
- 更新が失敗し、[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。

関連情報

- [管理外の Operator のサポートポリシー](#)

7.2. 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。

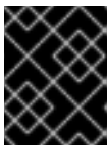
利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータル [の エラータ](#) のセクションを参照してください。

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。

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. 本番クラスターの場合は、**チャンネル** が、**stable-4.6** など、更新するバージョンの正しいチャン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-*** チャン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、クラスターを更新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。

入力チャンネルの **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。

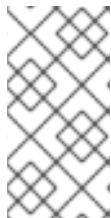


注記

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を更新する場合、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ン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、現在のチャン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-*** チャンネルに切り替え、そのチャンネルに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。クラスタの更新の終了において各 RHEL マシンの状態が **NotReady** になる際に、更新 Playbook を各 RHEL マシンに対して実行する必要があります。

7.3. 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 コンピュータマシンで Ansible タスクを実行できます。

7.3.1. 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 クラスタで RHEL コンピュータ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イスがありません。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。

7.3.2. 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) コンピュータマシン (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 コンピュータ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設定される必要があり、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。

7.3.3. RHEL コンピュータマシンで利用できるフック

Red Hat Enterprise Linux (RHEL) コンピュータマシンを 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
------	----

フック名	説明
openshift_node_pre_cordon_hook	<ul style="list-style-type: none"> ● 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	<ul style="list-style-type: none"> ● 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	<ul style="list-style-type: none"> ● 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。
openshift_node_post_upgrade_hook	<ul style="list-style-type: none"> ● 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。

7.4. クラスター内の RHEL コンピュータマシンの更新

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



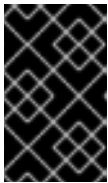
重要

ワーカー (コンピュート) マシン用にサポートされているのは Red Hat Enterprise Linux (RHEL) バージョン 7.9 以降であるため、RHEL ワーカーマシンをバージョン 8 にアップグレードすることはできません。

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。

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 4.6 で必要なりポジトリを有効にします。
 - a. Ansible Playbook を実行するマシンで、必要なりポジトリを更新します。

```
# subscription-manager repos --disable=rhel-7-server-ose-4.5-rpms \
--enable=rhel-7-server-ansible-2.9-rpms \
--enable=rhel-7-server-ose-4.6-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 コンピュータノードで、必要ならリポジトリを更新します。

```
# subscription-manager repos --disable=rhel-7-server-ose-4.5-rpms \
--enable=rhel-7-server-ose-4.6-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	v1.19.0
mycluster-control-plane-1	Ready	master	145m	v1.19.0
mycluster-control-plane-2	Ready	master	145m	v1.19.0
mycluster-rhel7-0 v1.14.6+97c81d00e	NotReady,SchedulingDisabled	worker	98m	
mycluster-rhel7-1	Ready	worker	98m	v1.14.6+97c81d00e
mycluster-rhel7-2	Ready	worker	98m	v1.14.6+97c81d00e
mycluster-rhel7-3	Ready	worker	98m	v1.14.6+97c81d00e

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。

- 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。
- 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	v1.19.0
mycluster-control-plane-1	Ready	master	145m	v1.19.0
mycluster-control-plane-2	Ready	master	145m	v1.19.0
mycluster-rhel7-0	NotReady,SchedulingDisabled	worker	98m	v1.19.0
mycluster-rhel7-1	Ready	worker	98m	v1.19.0
mycluster-rhel7-2	Ready	worker	98m	v1.19.0
mycluster-rhel7-3	Ready	worker	98m	v1.19.0

- 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。4.6 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

4.6 のインストール時に使用したのと同じ RPM リポジトリ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8章 ネットワークが制限された環境でのクラスタの更新

oc コマンドラインインターフェイス (CLI) を使用してネットワークが制限された OpenShift Container Platform クラスタを更新できます。

ネットワークが制限された環境とは、クラスタ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。ローカルコンテナレジストリーとクラスタ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を更新します。

8.1. 前提条件

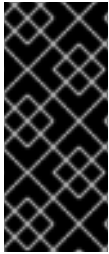
- 必要なコンテナイメージを取得するためのインターネットへのアクセスがあること。
- イメージをプッシュおよびプルするために、ネットワークが制限された環境でコンテナレジストリーへの書き込みアクセスがあること。コンテナレジストリーは Docker レジストリー API v2 と互換性がある必要があります。
- **oc** コマンドツールインターフェイス (CLI) ツールがインストールされていること。
- **admin** 権限を持つユーザーとしてクラスタにアクセスできること。[RBAC の使用によるパーミッションの定義および適用](#) を参照してください。
- 更新が失敗し、[クラスタを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。
- クラスタ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のアップグレード](#) を参照してください。

8.2. 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。

8.2.1. 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イスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.6 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。

8.2.1.1. Linux への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルでの [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 v4.6 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>
```

8.2.1.2. Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルでの [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 v4.6 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>
```

8.2.1.3. macOS への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルでの [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 v4.6 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>
```

8.3. 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の場所を特定している。
- イメージのイメージリポジトリ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. registry.redhat.io プルシークレットを [Red Hat OpenShift Cluster Manager](#) からダウンロードし、`.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
BGVtbYk3ZHAtdXs=
```

- 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>": { ❶
    "auth": "<credentials>", ❷
    "email": "you@example.com"
  },

```

- ❶ **<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:8443**
- ❷ **<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": "BGVtbYk3ZHAAtqXs=",
      "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"
    }
  }
}
}

```

8.4. OPENSIFT CONTAINER PLATFORM イメージリポジトリのミラーリング

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更新する前に、必要なコンテナ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更新を計画します。OpenShift Upgrade Graph はチャンネルのグラフと、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、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: **4.5.4**)。

- 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の名前を指定します。

- d. ミラーリングするリポジトリ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> ❶
```

- ❶ 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ミラ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} ❶
```

- 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。

- iv. **oc** コマンドラインインターフェイス (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。
- v. 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 ${REMOVABLE_MEDIA_PATH}/mirror/config/<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: **signature-sha256-81154f5c03294534.yaml**)。

- 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。

8.5. 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。

8.5.1. 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 更新パス](#) 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> ❶
```

- ❶ <release_version> について、クラスタを更新する 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> ❶
```

- ❶ **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に適用し、更新します。

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

8.6. ネットワークが制限された環境のクラスタのアップグレード

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**)、バージョン 4.4.8 以降をインストールします。

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** を使用する場合は、**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。

8.7. 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り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ことができます。
- 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、クラスタはソースリポジトリ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イメージリポジトリのソース
- ソースリポジトリ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クラスタにアクセスできる。

手順

1. ミラーリングされたりポジトリを設定します。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りポジトリを設定します。Red Hat Quay を使用すると、あるリポジトリから別のリポジトリにイメージをコピーでき、これらのリポジトリ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りポジトリ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@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eba
a6 \
docker://example.io/example/ubi-minimal
```

この例では、**example.io** という名前のコンテナ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 ❶
    source: registry.access.redhat.com/ubi8/ubi-minimal ❷
  - mirrors:
    - example.com/example/ubi-minimal
    source: registry.access.redhat.com/ubi8/ubi-minimal
  - mirrors:
    - mirror.example.com/redhat
    source: registry.redhat.io/openshift4 ❸
```

- ❶ 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。
- ❷ 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。
- ❸ 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は、**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   v1.19.0
ip-10-0-138-148.ec2.internal        Ready              master   11m  v1.19.0
ip-10-0-139-122.ec2.internal        Ready              master   11m  v1.19.0
ip-10-0-147-35.ec2.internal        Ready,SchedulingDisabled worker   7m   v1.19.0
ip-10-0-153-12.ec2.internal        Ready              worker   7m   v1.19.0
ip-10-0-154-10.ec2.internal        Ready              master   11m  v1.19.0
```

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@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eba
a6
```

リポジトリのミラーリングのトラブルシューティング

リポジトリ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。

8.8. 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が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**) をインストールしている。
- **cluster-admin** 権限を持つユーザーとしてログインしている。
- 非接続クラスターで使用するようミラーリングされたイメージカタログを設定する。

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、以下ようになります。

```
<local_registry>
```

非接続クラスタ (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:v4.6**)。

<pull_secret_file>

.json ファイル形式の **registry.redhat.io** プルシークレットです。プルシークレットは、[Red Hat Open Shift Cluster Manager からダウンロード](#)できます。

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:
      kubectrl.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。

8.9. 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](#)