



OpenShift Container Platform 4.16

クラスタの更新

OpenShift Container Platform クラスタの更新

OpenShift Container Platform 4.16 クラスターの更新

OpenShift Container Platform クラスターの更新

法律上の通知

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

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 OPENSIFT の更新について | 3 |
| 1.1. OPENSIFT の更新の概要 | 3 |
| 1.2. クラスターの更新の仕組み | 8 |
| 1.3. 更新チャンネルとリリースについて | 17 |
| 1.4. OPENSIFT CONTAINER PLATFORM の更新期間について | 21 |
| 第2章 クラスターの更新の準備 | 27 |
| 2.1. OPENSIFT CONTAINER PLATFORM 4.16 への更新の準備 | 27 |
| 2.2. 手動で維持された認証情報でクラスターを更新する準備 | 33 |
| 2.3. KERNEL MODULE MANAGEMENT (KMM) モジュールのプリフライト検証 | 52 |
| 第3章 クラスターの更新の実行 | 56 |
| 3.1. CLI を使用したクラスターの更新 | 56 |
| 3.2. WEB コンソールを使用してクラスターを更新 | 64 |
| 3.3. EUS から EUS への更新の実行 | 70 |
| 3.4. カナリアロールアウト更新の実行 | 76 |
| 3.5. RHEL コンピュータマシンを含むクラスターの更新 | 84 |
| 3.6. 非接続環境でのクラスターの更新 | 92 |
| 3.7. VSPHERE で稼働するノードでのハードウェアの更新 | 129 |
| 3.8. マルチアーキテクチャーのコンピュータマシンを備えたクラスターへの移行 | 133 |
| 3.9. HOSTED CONTROL PLANE の更新 | 135 |
| 3.10. BOOTUPD を使用して RHCOS ノード上のブートローダーを更新する | 136 |
| 第4章 クラスター更新のトラブルシューティング | 140 |
| 4.1. クラスター更新に関するデータ収集 | 140 |

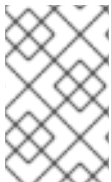
第1章 OPENSIFT の更新について

1.1. OPENSIFT の更新の概要

OpenShift Container Platform 4 では、Web コンソールまたは OpenShift CLI (**oc**) を使用して、OpenShift Container Platform クラスターを1回の操作で更新できます。プラットフォーム管理者は、Web コンソールの **Administration** → **Cluster Settings** に移動するか、**oc adm upgrade** コマンドの出力を確認して、新しい更新オプションを表示できます。

Red Hat はパブリック OpenShift Update Service (OSUS) をホストします。これは、公式レジストリーの OpenShift Container Platform リリースイメージに基づいて更新の可能性を示すグラフを提供します。グラフには、パブリック OCP リリースの更新情報が含まれます。OpenShift Container Platform クラスターはデフォルトで OSUS に接続するように設定されており、OSUS は既知の更新ターゲットに関する情報をクラスターに応答します。

クラスター管理者または自動更新コントローラーのいずれかが、Cluster Version Operator (CVO) のカスタムリソース (CR) を新しいバージョンで編集すると、更新が開始されます。クラスターを新たに指定したバージョンに合わせて調整するために、CVO はイメージレジストリーからターゲットリリースイメージを取得し、クラスターへの変更適用を開始します。



注記

Operator Lifecycle Manager (OLM) を介して以前にインストールされた Operator は、異なる更新プロセスに従います。詳細は、[インストールされている Operator の更新](#) を参照してください。

ターゲットリリースイメージには、特定の OCP バージョンを形成するすべてのクラスターコンポーネントのマニフェストファイルが含まれます。クラスターを新しいバージョンに更新する場合、CVO は Runlevels と呼ばれる別のステージでマニフェストを適用します。すべてではありませんが、ほとんどのマニフェストは、いずれかのクラスター Operator をサポートしています。CVO がクラスター Operator にマニフェストを適用すると、Operator が指定された新しいバージョンに適合させるために更新タスクを実行する可能性があります。

CVO は、適用された各リソースの状態と、すべてのクラスター Operator によって報告される状態を監視します。CVO は、アクティブな Runlevel のすべてのマニフェストおよびクラスター Operator が安定した状態に達した場合にのみ更新を続行します。CVO がこのプロセスを通じてコントロールプレーン全体を更新した後、Machine Config Operator (MCO) がオペレーティングシステムとクラスター内のすべてのノードの設定を更新します。

1.1.1. 更新の可用性に関するよくある質問

OpenShift Container Platform クラスターで更新が利用可能になるかどうか、またいつ利用可能になるかに影響を与える要因がいくつかあります。次のリストは、更新の入手可能性に関する一般的な質問を示しています。

各更新チャネルの違いは何ですか？

- 新しいリリースは、最初に **candidate** チャネルに追加されます。
- 最終テストが成功すると、**candidate** チャネルのリリースが **fast** チャネルに昇格され、正誤表が公開され、リリースは完全にサポートされるようになります。
- 遅延の後、**fast** チャネルでのリリースは最終的に **stable** チャネルに昇格されます。この遅延は、**fast** チャネルと **stable** チャネルの唯一の違いを表します。



注記

最新の z-stream リリースの場合、この遅延は通常 1~2 週間かかる可能性があります。ただし、最新のマイナーバージョンへの最初の更新の遅延にはさらに時間がかかる場合があります、通常は 45~90 日かかります。

- **stable** チャンネルにプロモートされたリリースは、同時に **eus** チャンネルにもプロモートされます。**eus** チャンネルの主な目的は、EUS から EUS への更新を実行するクラスターの利便性を高めることです。

stable チャンネルでのリリースは **fast** チャンネルでのリリースよりも安全ですか、それともよりサポートされていますか？

- **fast** チャンネルのリリースで回帰が特定された場合、その回帰が **stable** チャンネルのリリースで特定された場合と同じ程度に解決され、管理されます。
- **fast** チャンネルと **stable** チャンネルのリリースの唯一の違いは、リリースが **fast** チャンネル上にしばらく存在した後にはのみ **stable** チャンネルに表示されることです。これにより、新しい更新のリスクが発見されるまでの時間が長くなります。
- この遅延の後、**fast** チャンネルで利用可能なリリースは必ず **安定** チャンネルでも利用可能になります。

更新に既知の問題がある場合、それは何を意味しますか？

- Red Hat は、複数のソースからのデータを継続的に評価して、あるバージョンから別のバージョンに更新する場合に、報告された問題があるかどうかを判断します。問題が特定されている場合は通常、バージョンのリリースノートに記載されます。更新パスに既知の問題がある場合でも、更新を行ってもサポートが受けられます。
- Red Hat は、ユーザーが特定のバージョンに更新することをブロックしません。Red Hat は条件付き更新のリスクを宣言する場合がありますが、これは特定のクラスターに適用される場合と適用されない場合があります。
 - 宣言されたリスクにより、クラスター管理者はサポートされている更新に関する詳細なコンテキストが得られます。クラスター管理者はリスクを受け入れて、その特定のターゲットバージョンに更新することができます。

特定のリリースへの更新が推奨されなくなった場合はどうすればよいですか？

- Red Hat がリグレッションのためにサポートされているリリースから更新の推奨事項を削除した場合、リグレッションを修正する将来のバージョンに、代替となる更新の推奨事項が提供されます。問題の修正とテスト、選択したチャンネルへの昇格に、時間がかかる可能性があります。

次の z-stream リリースが高速で安定したチャンネルで利用できるようになるまで、どれくらいかかりますか？

- 具体的な頻度はさまざまな要因によって異なりますが、最新のマイナーバージョンの新しい z-stream リリースは通常、ほぼ毎週公開されます。古いマイナーバージョンは時間の経過とともに安定してきており、新しい z-stream リリースが利用可能になるまでにさらに時間がかかる場合があります。



重要

これらは、z-stream リリースに関する過去のデータに基づく推定にすぎません。Red Hat は、必要に応じてリリース頻度を変更する権利を保持しています。問題がいくらかでも発生すると、このリリースサイクルに不規則性や遅れが生じる可能性があります。

- Z-stream リリースが公開されると、そのマイナーバージョンの **fast** チャンネルにも表示されません。遅延後、z-stream リリースがそのマイナーバージョンの **stable** チャンネルに表示される場合があります。

関連情報

- [更新チャンネルとリリースについて](#)

1.1.2. OpenShift Update Service について

OpenShift Update Service (OSUS) は、Red Hat Enterprise Linux CoreOS (RHCOS) を含む OpenShift Container Platform に更新の推奨項目を提供します。コンポーネント Operator のグラフ、または **頂点** とそれらをつなぐ **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。更新をリクエストすると、CVO は対応するリリースイメージを使用し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 によって推奨される更新パスがない場合は、非互換性や可用性など、更新パスに関連する既知の問題が原因で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、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

新しいバージョンへの更新のみがサポートされています。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** フィールドで指定されたノードの数を制限し、それら

を使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。MCO は、**topology.kubernetes.io/zone** ラベルに基づいて、影響を受けるノードをゾーンごとにアルファベット順に更新します。ゾーンに複数のノードがある場合、最も古いノードが最初に更新されます。ベアメタルデプロイメントなど、ゾーンを使用しないノードの場合、ノードは経過時間ごとに更新され、最も古いノードが最初に更新されます。MCO は、マシン設定プールの **maxUnavailable** フィールドで指定されたノード数を一度に更新します。次に、MCO は新しい設定を適用して、マシンを再起動します。



警告

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

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。

1.1.3. クラスター Operator の状態タイプについて

クラスター Operator のステータスには、Operator の現在の正常性状態を通知する状態タイプが含まれています。以下の定義では、一般的な ClusterOperator の状態タイプをいくつか取り上げています。追加の状態タイプがあり、Operator 固有の言語を使用する Operator は省略されています。

Cluster Version Operator (CVO) は、クラスター管理者が OpenShift Container Platform クラスターのステータス状態をよりよく理解できるように、クラスター Operator からステータス状態を収集します。

- Available: 条件タイプ **Available** は、Operator が機能しており、クラスターで使用可能であることを示します。ステータスが **False** の場合、オペランドの少なくとも1つの部分が機能していないため、管理者が介入する必要があります。
- Progressing: 条件タイプ **Progressing** は、Operator がアクティブに新しいコードをロールアウトしている、設定の変更を伝達している、またはある安定状態から別の安定状態に移行していることを示します。
Operator が以前の既知の状態を調整している場合は、状態タイプ **Progressing** は **True** として報告されません。監視されたクラスターの状態が変化し、Operator がそれに反応している場合は、ある安定状態から別の安定状態に移行しているため、ステータスは **True** として報告されます。
- Degraded: 状態タイプ **Degraded** は、Operator の現在の状態が一定期間にわたって必要な状態に一致しないことを示します。期間はコンポーネントによって異なる場合がありますが、**Degraded** ステータスは、Operator の状態が継続的に監視されていることを表します。そ

のため、Operator の **Degraded** 状態が変動することはありません。

ある状態から別の状態への移行期間が短すぎるために **Degraded** を報告できない場合は、別の状態タイプが報告される可能性があります。Operator は、通常の更新中、**Degraded** を報告しません。Operator は、最終的に管理者の介入を必要とする永続的なインフラストラクチャー障害への対応として、機能 **Degraded** を報告する場合があります。



注記

この状態タイプは、調査と調整が必要な可能性があることを示しているにすぎません。Operator が使用可能である限り、**Degraded** 状態によってユーザーワークロードの障害やアプリケーションのダウンタイムが発生することはありません。

- Upgradeable: 条件タイプ **Upgradeable** は、Operator が、現在のクラスタの状態に基づいて、安全に更新できるかどうかを示します。メッセージフィールドには、クラスタを正常に更新するために管理者が行う必要があることについて、人間が判読できる説明が含まれています。CVO は、この状態が **True**、**Unknown**、または状態がない場合に更新を許可します。**Upgradeable** ステータスが **False** の場合、マイナー更新のみが影響を受け、CVO は、強制されない限り、影響を受ける更新をクラスタが実行できないようにします。

1.1.4. クラスタバージョン条件タイプについて

Cluster Version Operator (CVO) は、クラスタ Operator およびその他のコンポーネントを監視し、クラスタバージョンとその Operator の両方のステータスを収集します。このステータスには、OpenShift Container Platform クラスタの正常性と現在の状態を通知する条件タイプが含まれます。

Available、**Progressing**、**Upgradeable** に加えて、クラスタのバージョンと Operator に影響する条件タイプがあります。

- Failing: クラスタバージョン条件タイプ **Failing** は、クラスタが目的の状態に到達できず、異常であり、管理者の介入が必要であることを示します。
- Invalid: クラスタバージョン条件タイプ **Invalid** は、サーバーがアクションを実行できないエラーがクラスタバージョンにあることを示します。この条件が設定されているかぎり、CVO は現在の状態のみを調整します。
- RetrievedUpdates: クラスタバージョン条件タイプ **RetrievedUpdates** は、利用可能な更新が上流の更新サーバーから取得されたかどうかを示します。取得前の条件は **Unknown** です。更新が最近失敗したか、取得できなかった場合は、**False**、**availableUpdates** フィールドが最新および正確である場合は、**True** です。
- ReleaseAccepted: **True** ステータスのクラスタバージョン条件タイプ **ReleaseAccepted** は、要求されたリリースペイロードが、イメージの検証および前提条件のチェック中、失敗せずに、正常に読み込まれたことを示します。
- ImplicitlyEnabledCapabilities: **True** ステータスのクラスタバージョン条件タイプ **ImplicitlyEnabledCapabilities** は、ユーザーが現在 **spec.capabilities** を介して要求していない有効な機能があることを示します。関連するリソースが以前に CVO によって管理されていた場合、CVO は機能の無効化をサポートしません。

1.1.5. 一般的な用語

コントロールプレーン

コントロールプレーンマシンで設定される **コントロールプレーン** は、OpenShift Container Platform クラスターを管理します。コントロールプレーンマシンは、コンピュータマシン (ワーカーマシンとしても知られる) のワークロードを管理します。

Cluster Version Operator

Cluster Version Operator (CVO) は、クラスターの更新プロセスを開始します。現在のクラスターバージョンに基づいて OSUS を確認し、利用可能または可能な更新パスを含むグラフを取得します。

Machine Config Operator

Machine Config Operator (MCO) は、オペレーティングシステムおよびマシン設定を管理するクラスターレベルの Operator です。プラットフォーム管理者は、MCO を介して、systemd、CRI-O、Kubelet、カーネル、NetworkManager、およびワーカーノード上のその他のシステム機能を設定および更新できます。

OpenShift Update Service

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。

チャンネル

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た更新戦略を宣言します。OSUS は、この設定された戦略を使用して、その戦略と一致する更新エッジを推奨します。

推奨される更新エッジ

推奨される更新エッジ は、OpenShift Container Platform リリース間の推奨される更新です。特定の更新が推奨されるかどうかは、クラスターの設定済みチャンネル、現在のバージョン、既知のバグ、およびその他の情報によって異なります。OSUS は、推奨されるエッジを、すべてのクラスターで実行される CVO に伝達します。

延長更新サポート (EUS)

4.7 以降の偶数番号のマイナーリリースはすべて、**Extended Update Support (EUS)** リリースとしてラベル付けされています。これらのリリースでは、EUS リリース間に検証済みの更新パスが導入され、お客様はワーカーのワーカーノードの更新が合理化され、ワーカーノードの再起動を減らす EUS から EUS への OpenShift Container Platform リリースの更新戦略を策定できます。

詳細は、[Red Hat OpenShift 延長更新サポート \(EUS\) の概要](#) を参照してください。

関連情報

- [マシン設定の概要](#)
- [非接続環境での OpenShift Update Service の使用](#)
- [更新チャンネル](#)

1.1.6. 関連情報

- [クラスターの更新の仕組み](#)。

1.2. クラスターの更新の仕組み

以下のセクションでは、OpenShift Container Platform (OCP) 更新プロセスの各腫瘍点について詳しく説明しています。更新の仕組みの概要は、[OpenShift 更新の概要](#) を参照してください。

1.2.1. Cluster Version Operator

Cluster Version Operator (CVO) は、OpenShift Container Platform の更新プロセスを調整および促進する主要コンポーネントです。インストールや標準的なクラスター操作を実行する間、CVO はマネージドクラスター Operator のマニフェストとクラスター内リソースを常に比較し、これらのリソースの実際の状態が求められる状態と一致するように、不一致を調整します。

1.2.1.1. ClusterVersion オブジェクト

Cluster Version Operator (CVO) が監視するリソースの1つに、**ClusterVersion** リソースがあります。

管理者と OpenShift コンポーネントは、**ClusterVersion** オブジェクトを通じて CVO と通信または対話できます。CVO に求められる状態は **ClusterVersion** オブジェクトを通じて宣言され、現在の CVO 状態はオブジェクトのステータスに反映されます。



注記

ClusterVersion オブジェクトは直接変更しないでください。代わりに、**oc** CLI や Web コンソールなどのインターフェイスを使用して、更新ターゲットを宣言します。

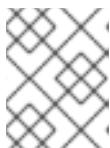
CVO は、**ClusterVersion** リソースの **spec** プロパティで宣言されたターゲットとする状態とクラスターを継続的に調整します。必要なリリースと実際のリリースが異なる場合、その調整によってクラスターが更新されます。

可用性データの更新

ClusterVersion リソースには、クラスターが利用できる更新に関する情報も含まれています。これには、利用可能な更新プログラムも含まれますが、クラスターに適用される既知のリスクのため推奨されません。これらの更新は条件付き更新として知られています。CVO が **ClusterVersion** リソース内の利用可能な更新に関する情報をどのように維持するかについては、「更新の可用性評価」セクションを参照してください。

- 以下のコマンドを使用して、利用可能なすべての更新を確認できます。

```
$ oc adm upgrade --include-not-recommended
```



注記

追加の **--include-not-recommended** パラメーターには、クラスターに適用される既知の問題で利用可能な更新が含まれます。

出力例

```
Cluster version is 4.13.40
```

```
Upstream is unset, so the cluster will use an appropriate default.
```

```
Channel: stable-4.14 (available channels: candidate-4.13, candidate-4.14, eus-4.14, fast-4.13, fast-4.14, stable-4.13, stable-4.14)
```

```
Recommended updates:
```

```
VERSION IMAGE
```

```
4.14.27 quay.io/openshift-release-dev/ocp-release@sha256:4d30b359aa6600a89ed49ce6a9a5fdab54092bcb821a25480dfbc47e66af9ec
```

```
4.14.26 quay.io/openshift-release-dev/ocp-release@sha256:4fe7d4ccf4d967a309f83118f1a380a656a733d7fcee1dbaf4d51752a6372890
```

4.14.25 quay.io/openshift-release-dev/ocp-release@sha256:a0ef946ef8ae75aef726af1d9bbaad278559ad8cab2c1ed1088928a0087990b6

4.14.24 quay.io/openshift-release-dev/ocp-release@sha256:0a34eac4b834e67f1bca94493c237e307be2c0eae7b8956d4d8ef1c0c462c7b0

4.14.23 quay.io/openshift-release-dev/ocp-release@sha256:f8465817382128ec7c0bc676174bad0fb43204c353e49c146ddd83a5b3d58d92

4.13.42 quay.io/openshift-release-dev/ocp-release@sha256:dcf5c3ad7384f8bee3c275da8f886b0bc9aea7611d166d695d0cf0fff40a0b55

4.13.41 quay.io/openshift-release-dev/ocp-release@sha256:dbb8aa0cf53dc5ac663514e259ad2768d8c82fd1fe7181a4cfb484e3ffdbd3ba

Updates with known issues:

Version: 4.14.22

Image: quay.io/openshift-release-dev/ocp-release@sha256:7093fa606debe63820671cc92a1384e14d0b70058d4b4719d666571e1fc62190

Reason: MultipleReasons

Message: Exposure to AzureRegistryImageMigrationUserProvisioned is unknown due to an evaluation failure: client-side throttling: only 18.061µs has elapsed since the last match call completed for this cluster condition backend; this cached cluster condition request has been queued for later execution

In Azure clusters with the user-provisioned registry storage, the in-cluster image registry component may struggle to complete the cluster update.

<https://issues.redhat.com/browse/IR-468>

Incoming HTTP requests to services exposed by Routes may fail while routers reload their configuration, especially when made with Apache HTTPClient versions before 5.0. The problem is more likely to occur in clusters with higher number of Routes and corresponding endpoints. <https://issues.redhat.com/browse/NE-1689>

Version: 4.14.21

Image: quay.io/openshift-release-dev/ocp-release@sha256:6e3fba19a1453e61f8846c6b0ad3abf41436a3550092cbfd364ad4ce194582b7

Reason: MultipleReasons

Message: Exposure to AzureRegistryImageMigrationUserProvisioned is unknown due to an evaluation failure: client-side throttling: only 33.991µs has elapsed since the last match call completed for this cluster condition backend; this cached cluster condition request has been queued for later execution

In Azure clusters with the user-provisioned registry storage, the in-cluster image registry component may struggle to complete the cluster update.

<https://issues.redhat.com/browse/IR-468>

Incoming HTTP requests to services exposed by Routes may fail while routers reload their configuration, especially when made with Apache HTTPClient versions before 5.0. The problem is more likely to occur in clusters with higher number of Routes and corresponding endpoints. <https://issues.redhat.com/browse/NE-1689>

oc adm upgrade コマンドは、利用可能な更新に関する情報を **ClusterVersion** リソースにクエリーし、人間が判読できる形式で表示します。

- CVO が作成した基礎となる可用性データを直接検査する方法の1つに、次のコマンドを使用して **ClusterVersion** リソースをクエリーする方法があります。

```
$ oc get clusterversion version -o json | jq '.status.availableUpdates'
```

出力例

```
[
  {
    "channels": [
      "candidate-4.11",
      "candidate-4.12",
      "fast-4.11",
      "fast-4.12"
    ],
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:400267c7f4e61c6bfa0a59571467e8bd85c9188e442cbd820cc8263809be377
5",
    "url": "https://access.redhat.com/errata/RHBA-2023:3213",
    "version": "4.11.41"
  },
  ...
]
```

- 同様のコマンドを使用して条件付き更新を確認できます。

```
$ oc get clusterversion version -o json | jq '.status.conditionalUpdates'
```

出力例

```
[
  {
    "conditions": [
      {
        "lastTransitionTime": "2023-05-30T16:28:59Z",
        "message": "The 4.11.36 release only resolves an installation issue
https://issues.redhat.com//browse/OCPBUGS-11663 , which does not affect already running
clusters. 4.11.36 does not include fixes delivered in recent 4.11.z releases and therefore
upgrading from these versions would cause fixed bugs to reappear. Red Hat does not
recommend upgrading clusters to 4.11.36 version for this reason.
https://access.redhat.com/solutions/7007136",
        "reason": "PatchesOlderRelease",
        "status": "False",
        "type": "Recommended"
      }
    ],
    "release": {
      "channels": [...],
      "image": "quay.io/openshift-release-dev/ocp-
release@sha256:8c04176b771a62abd801fcd3e952633566c8b5ff177b93592e8e8d2d1f8471d
",
      "url": "https://access.redhat.com/errata/RHBA-2023:1733",
      "version": "4.11.36"
    }
  },
  ...
]
```



```

    "risks": [...],
  },
  ...
]

```

1.2.1.2. 更新の可用性評価

Cluster Version Operator (CVO) は、OpenShift Update Service (OSUS) に対して、更新の可能性に関する最新データを定期的にクエリーします。このデータは、クラスターがサブスクライブしているチャンネルに基づいています。次に、CVO は更新の推奨事項に関する情報を、**ClusterVersion** リソースの **availableUpdates** フィールドまたは **conditionalUpdates** フィールドに保存します。

CVO は、条件付き更新の更新リスクを定期的に確認します。これらのリスクは、OSUS によって提供されるデータを通じて伝えられます。このデータには、そのバージョンに更新されたクラスターに影響を与える可能性がある各バージョンの既知の問題に関する情報が含まれています。ほとんどのリスクは、特定のサイズのクラスターや特定のクラウドプラットフォームにデプロイされたクラスターなど、特定の特性を持つクラスターに限定されます。

CVO は、各条件付き更新の条件付きリスクに関する情報に対して、継続的にクラスターの特性を評価します。CVO は、クラスターが基準に一致することを検出すると、その情報を **ClusterVersion** リソースの **conditionalUpdates** フィールドに保存します。CVO は、クラスターが更新のリスクに一致しないこと、または更新に関連するリスクがないことを検出すると、ターゲットバージョンを **ClusterVersion** リソースの **availableUpdates** フィールドに保存します。

Web コンソールまたは OpenShift CLI (**oc**) のユーザーインターフェイスは、この情報をセクションの見出しで表示します。更新パスに関連付けられた既知の問題にはそれぞれ、リスクに関する詳細なリソースへのリンクが含まれているため、管理者は更新について十分な情報に基づいた決定を下すことができます。

関連情報

- [更新推奨の削除と条件付き更新プログラム](#)

1.2.2. リリースイメージ

リリースイメージは、特定の OpenShift Container Platform (OCP) バージョンのディストリビューションメカニズムです。これには、リリースメタデータ、リリースバージョンに一致する Cluster Version Operator (CVO) バイナリー、個々の OpenShift Cluster Operator のデプロイに必要なすべてのマニフェスト、この OpenShift バージョンを構成するすべてのコンテナイメージへの SHA ダイジェストバージョン参照リストが含まれています。

次のコマンドを実行して、特定のリリースイメージの内容を検査できます。

```
$ oc adm release extract <release image>
```

出力例

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.12.6-x86_64
Extracted release payload from digest
sha256:800d1e39d145664975a3bb7cbc6e674fbf78e3c45b5dde9ff2c5a11a8690c87b created at
2023-03-01T12:46:29Z
```

```
$ ls
0000_03_authorization-openshift_01_rolebindingrestriction.crd.yaml
```



```

0000_03_config-operator_01_proxy.crd.yaml
0000_03_marketplace-operator_01_operatorhub.crd.yaml
0000_03_marketplace-operator_02_operatorhub.cr.yaml
0000_03_quota-openshift_01_clusterresourcequota.crd.yaml ❶
...
0000_90_service-ca-operator_02_prometheusrolebinding.yaml ❷
0000_90_service-ca-operator_03_servicemonitor.yaml
0000_99_machine-api-operator_00_tombstones.yaml
image-references ❸
release-metadata

```

- ❶ Runlevel 03 に適用される **ClusterResourceQuota** CRD のマニフェスト
- ❷ ランレベル 90 に適用される、**service-ca-operator** の **PrometheusRoleBinding** リソースのマニフェスト
- ❸ 必要なすべてのイメージへの SHA ダイジェストバージョン参照リスト

1.2.3. プロセスワークフローの更新

以下の手順は、OpenShift Container Platform (OCP) 更新プロセスの詳細なワークフローを示しています。

1. ターゲットバージョンは、**ClusterVersion** リソースの **spec.desiredUpdate.version** フィールドに保存され、Web コンソールまたは CLI から管理されます。
2. Cluster Version Operator (CVO) は、**ClusterVersion** リソースの **desiredUpdate** が現在のクラスタのバージョンとは異なることを検出します。OpenShift Update Service からのグラフデータを使用して、CVO は必要なクラスタバージョンをリリースイメージのプル仕様に解決します。
3. CVO は、リリースイメージの整合性と信頼性を検証します。Red Hat は、イメージ SHA ダイジェストを一意で不変のリリースイメージ識別子として使用し、公開されたリリースイメージに関する暗号署名されたステートメントを事前定義された場所に公開します。CVO はビルトイン公開鍵のリストを使用して、チェックされたリリースイメージに一致するステートメントの存在と署名を検証します。
4. CVO は、**openshift-cluster-version** namespace に **version-\$version-\$hash** という名前のジョブを作成します。このジョブはリリースイメージを実行しているコンテナを使用するため、クラスタはコンテナランタイムを通じてイメージをダウンロードします。次に、ジョブはマニフェストとメタデータをリリースイメージから CVO がアクセス可能な共有ボリュームに展開します。
5. CVO は、展開されたマニフェストとメタデータを検証します。
6. CVO はいくつかの前提条件をチェックして、クラスタ内で問題のある状態が検出されないことを確認します。特定の状態により、更新が続行できない場合があります。これらの状態は、CVO 自体によって決定されるか、Operator が更新に問題ありと判断するクラスタの詳細を検出する個々のクラスタ Operator によって報告されます。
7. CVO は、承認されたリリースを **status.desired** に記録し、新しい更新に関する **status.history** エントリーを作成します。

8. CVO は、リリースイメージからマニフェストの調整を開始します。クラスター Operator は Runlevels と呼ばれる別のステージで更新され、CVO は次のレベルに進む前に Runlevel 内のすべての Operator が更新を完了するようにします。
9. CVO 自体のマニフェストはプロセスの早い段階で適用されます。CVO デプロイメントが適用されると、現在の CVO Pod が停止し、新しいバージョンを使用する CVO Pod が開始されます。新しい CVO は、残りのマニフェストの調整を進めます。
10. 更新は、コントロールプレーン全体が新しいバージョンに更新されるまで続行されます。個々のクラスター Operator は、クラスターのドメインで更新タスクを実行することがあり、その場合は実行中に、**Progressing=True** 状態を通して状態を報告します。
11. Machine Config Operator (MCO) マニフェストはプロセスの最後に適用されます。その後、更新された MCO は、すべてのノードのシステム設定とオペレーティングシステムの更新を開始します。各ノードは、再びワークロードの受け入れを開始する前に、ドレイン、更新、および再起動される可能性があります。

クラスターは、コントロールプレーンの更新が完了した後、通常はすべてのノードが更新される前に更新済みであることを報告します。更新後、CVO はすべてのクラスターリソースを、リリースイメージで提供される状態と一致するように維持します。

1.2.4. 更新時のマニフェストの適用方法について

リリースイメージで提供される一部のマニフェストは、依存関係があるため、特定の順序で適用する必要があります。たとえば、**CustomResourceDefinition** リソースは、一致するカスタムリソースの前に作成する必要があります。さらに、クラスター内の断絶を最小限に抑えるために、個々のクラスター Operator は論理的な順序に従い更新される必要があります。Cluster Version Operator (CVO) は、Runlevels の概念を通じてこの論理的な順序を実装します。

これらの依存関係は、リリースイメージのマニフェストのファイル名でエンコードされます。

```
0000_<runlevel>_<component>_<manifest-name>.yaml
```

以下に例を示します。

```
0000_03_config-operator_01_proxy.crd.yaml
```

CVO は内部でマニフェストの依存関係グラフをビルドします。ここで CVO は次のルールに従います。

- 更新中、より低位の Runlevel のマニフェストは、高位の Runlevel のマニフェストよりも先に適用されます。
- 1つの Runlevel 内で、異なるコンポーネントのマニフェストを並行して適用できます。
- 1つの Runlevel 内で、単一のコンポーネントのマニフェストは辞書式の順序で適用されます。

次に、CVO は生成された依存関係グラフの順にマニフェストを適用します。



注記

一部のリソースタイプでは、CVO はマニフェストの適用後にリソースを監視し、リソースが安定した状態に達して場合に限り正常に更新されたとみなします。この状態に達するまでに時間がかかる場合があります。これは特に **ClusterOperator** リソースに当てはまりますが、CVO はクラスター Operator が自身を更新するのを待ってから、その **ClusterOperator** ステータスを更新します。

CVO は、Runlevel のすべてのクラスター Operator が以下の状態になるまで待機してから、次の Runlevel に進みます。

- クラスター Operator は **Available=True** の状態です。
- クラスター Operator は **Degraded=False** の状態です。
- クラスター Operator は、ClusterOperator リソースで必要なバージョンになったことを宣言します。

一部のアクションは、完了するまでにかなりの時間がかかる場合があります。CVO は、後続の Runlevel で安全に続行できるように、アクションが完了するのを待ちます。新しいリリースのマニフェストを初めて調整する場合、合計で 60 ~ 120 分かかると予想されます。**更新期間に影響を与える要因の詳細については、OpenShift Container Platform の更新期間についてを参照してください。**

✔ Completed
🔄 In progress
🕒 Waiting



341_OpenShift_0623

前のサンプル図では、CVO は Runlevel 20 ですべての作業が完了するまで待機しています。CVO はすべてのマニフェストを Runlevel の Operator に適用しましたが、**kube-apiserver-operator ClusterOperator** は一部のアクションを新しいバージョンがデプロイされた後に実行します。**kube-apiserver-operator ClusterOperator** は、**Progressing=True** の状態であることと、新しいバージョン

を **status.versions** で調整済みとして宣言しないことによって、進捗を宣言します。CVO は、ClusterOperator が許容可能なステータスを報告するまで待機し、その後、Runlevel 25 でマニフェストの調整を開始します。

関連情報

- [OpenShift Container Platform の更新期間について](#)

1.2.5. Machine Config Operator によるノードの更新方法

Machine Config Operator (MCO) は、新しいマシン設定を各コントロールプレーンノードとコンピュータードに適用します。マシン設定の更新時に、コントロールプレーンノードとコンピュータードは、マシンプールが並行して更新される独自のマシン設定プールに編成されます。**.spec.maxUnavailable** パラメーター (デフォルト値は **1**) は、マシン設定プール内の更新プロセスを同時に実行できるノードの数を決定します。



警告

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

マシン設定の更新プロセスが開始されると、MCO はプール内の現在利用できないノードの数を確認します。使用できないノードの数が **.spec.maxUnavailable** の値よりも少ない場合、MCO はプール内の使用可能なノードに対して次の一連のアクションを開始します。

1. ノードを遮断してドレインします。



注記

ノードが遮断されている場合、ワークロードをそのノードにスケジュールすることはできません。

2. ノードのシステム設定およびオペレーティングシステム (OS) を更新します。
3. ノードを再起動します。
4. ノードの遮断を解除します。

このプロセスが実行されているノードは、社団が解除されてワークロードが再度スケジュールされるまで使用できません。MCO は、使用できないノードの数が **.spec.maxUnavailable** の値と等しくなるまでノードの更新を開始します。

ノードが更新を完了して使用可能になると、マシン設定プール内の使用不可ノードの数は再び **.spec.maxUnavailable** より少なくなります。更新する必要があるノードが残っている場合、MCO は **.spec.maxUnavailable** 制限に再度達するまで、ノード上で更新プロセスを開始します。このプロセスは、各コントロールプレーンノードとコンピュータードが更新されるまで繰り返されます。

次のワークフロー例は、5つのノードを持つマシン設定グループでこのプロセスがどのように発生するかを示しています。ここでの `.spec.maxUnavailable` は3で、最初はすべてのノードが使用可能です。

1. MCO はノード 1、2、3 を遮断し、それらのドレインを開始します。
2. ノード 2 は、ドレインを完了して再起動すると再び使用可能になります。MCO はノード 4 を遮断し、そのドレインを開始します。
3. ノード 1 は、ドレインを完了して再起動すると再び使用可能になります。MCO はノード 5 を遮断し、そのドレインを開始します。
4. ノード 3 は、ドレインを完了して再起動すると再び使用可能になります。
5. ノード 5 は、ドレインを完了して再起動すると再び使用可能になります。
6. ノード 4 は、ドレインを完了して再起動すると再び使用可能になります。

各ノードの更新プロセスは他のノードから独立しているため、上記の例におけるノードの一部は、MCO によって遮断された順序とは異なる順序で更新を終了します。

次のコマンドを実行して、マシン設定の更新ステータスを確認できます。

```
$ oc get mcp
```

出力例

| NAME | CONFIG | UPDATED | UPDATING | DEGRADED | | |
|----------------------|--|---------------------|----------|----------|-------|---|
| MACHINECOUNT | READYMACHINECOUNT | UPDATEDMACHINECOUNT | | | | |
| DEGRADEDMACHINECOUNT | AGE | | | | | |
| master | rendered-master-acd1358917e9f98cbdb599aea622d78b | | True | False | False | 3 |
| 3 | 3 | 0 | | 22h | | |
| worker | rendered-worker-1d871ac76e1951d32b2fe92369879826 | | False | True | False | 2 |
| 1 | 1 | 0 | | 22h | | |

関連情報

- [マシン設定の概要](#)

1.3. 更新チャンネルとリリースについて

更新チャンネルは、クラスターを更新する予定の OpenShift Container Platform マイナーバージョンをユーザーが宣言するメカニズムです。また、ユーザーは、更新のタイミングとサポートレベルを、**fast**、**stable**、**candidate**、および **eus** チャンネルオプションから選択することもできます。Cluster Version Operator は、チャンネル宣言に基づく更新グラフを他の条件付き情報と共に使用して、クラスターで利用可能な推奨更新と条件付き更新のリストを提供します。

更新チャンネルは、OpenShift Container Platform のマイナーバージョンに対応します。チャンネルのバージョン番号は、クラスターの現在のマイナーバージョンよりも新しいバージョンであっても、クラスターが最終的に更新されるターゲットマイナーバージョンを表します。

例えば、OpenShift Container Platform 4.10 更新チャンネルは以下の推奨事項を提供します。

- 4.10 内の更新。
- 4.9 内での更新。

- 4.9 から 4.10 への更新。すべての 4.9 クラスターが、z-stream の最小バージョン要件をすぐに満たさなくても、最終的に 4.10 に更新できます。
- **eus-4.10** のみ: 4.8 内で更新。
- **eus-4.10** のみ: 4.8 から 4.9 を経て 4.10 に更新され、すべての 4.8 クラスターが最終的に 4.10 に更新されます。

4.10 更新チャンネルでは、4.11 以降のリリースへの更新は推奨されません。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に更新することを明示的に決定する必要があります。

更新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しません。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、常にそのバージョンをインストールします。

OpenShift Container Platform 4.16 は、以下の更新チャンネルを提供します。

- **stable-4.16**
- **eus-4.y** (EUS バージョンでのみ提供され、EUS バージョン間の更新を容易にするためのもの)
- **fast-4.16**
- **candidate-4.16**

Cluster Version Operator を更新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ン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更新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 によって提案されたバージョンにのみ更新することが推奨されます。マイナーバージョン更新の場合、バージョンは連続している必要があります。Red Hat は、非連続バージョンへの更新をテストせず、以前のバージョンとの互換性を保証できません。

1.3.1. 更新チャンネル

1.3.1.1. fast-4.16 チャンネル

fast-4.16 チャンネルは、Red Hat が OpenShift Container Platform 4.16 の新しいバージョンを一般公開 (GA) リリースとして宣言するとすぐに更新されます。そのため、これらのリリースは完全にサポートされており、実稼働環境での使用を目的としています。

1.3.1.2. stable-4.16 チャンネル

fast-4.16 チャンネルにはエラータが公開されるとすぐにリリースが含まれますが、**stable-4.16** チャンネルの場合は遅延後にリリースが追加されます。この遅延の間に、複数のソースからデータが収集され、製品のリグレッションの兆候がないか分析されます。相当数のデータポイントが収集されると、これらの

リリースは stable チャンネルに追加されます。



注記

かなりの数のデータポイントを取得するのに必要な時間は多くの要因によって異なるため、高速チャンネルと安定チャンネルの間の遅延期間に関して、サービスレベル目標 (SLO) は設定されていません。詳細は、「Choosing the correct channel for your cluster」を参照してください。

新しくインストールされたクラスターは、デフォルトで安定したチャンネルを使用します。

1.3.1.3. 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。stable チャンネルに昇格したリリースは、同時に EUS チャンネルにも昇格されます。EUS チャンネルの主な目的は、EUS から EUS への更新を実行するクラスターの利便性を高めることです。



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。

1.3.1.4. candidate-4.16 チャンネル

candidate-4.16 チャンネルは、リリースがビルドされるとすぐに、そのリリースへのアクセスを提供しますが、サポートはされません。candidate チャンネルのみに存在するリリースには、最終的な GA リリースの完全な機能セットが含まれていないか、GA の前に機能が削除される可能性があります。さらに、これらのリリースは完全な Red Hat 品質保証の対象ではなく、後の GA リリースへの更新パスが提供されない可能性があります。これらの考慮事項を鑑みると、candidate チャンネルは、クラスターの破棄と再作成が許容されるテスト目的にのみ適しています。

1.3.1.5. チャンネルでの更新推奨

OpenShift Container Platform には更新推奨サービスがあり、インストール済みの OpenShift Container Platform バージョンと、次のリリースにアクセスするためにチャンネル内のパスを確認できるようになっています。更新パスも、現在選択されているチャンネルとそのプロモーション特性に関連するバージョンに限定されます。

お使いのチャンネルでは、以下のリリースが確認できます。

- 4.16.0
- 4.16.1
- 4.16.3
- 4.16.4

このサービスは、テスト済みで重大なリグレッションが確認されていない更新のみを推奨します。たとえば、クラスターが 4.16.1 にあり、OpenShift Container Platform が 4.16.4 を提案している場合は、4.16.1 から 4.16.4 に更新することを推奨します。



重要

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。今回の例では、過去から現在に至るまでこのチャンネルで 4.16.2 が使用できなかったことがないため、4.16.2 への更新は推奨されず、サポートもされません。

1.3.1.6. 更新の推奨と条件付き更新

Red Hat は、サポートチャンネルに追加する前後で、新規リリースバージョンおよび、このような新規リリースバージョンに関連する更新パスをモニタリングしています。

Red Hat は、サポート対象リリースから更新の推奨を削除する場合には、今後のバージョンに対して、そのリグレッションを修正する、代わりとなる更新の推奨が提供される予定です。ただし、問題の修正とテスト、選択したチャンネルへの昇格に、時間がかかる可能性があります。

OpenShift Container Platform 4.10 以降、確認された更新リスクは、該当する更新の条件付き更新リスクとして宣言されます。既知の各リスクは、すべてのクラスターに適用される場合もあれば、特定の条件に一致するクラスターのみ適用される場合もあります。たとえば、**Platform** を **None** に、CNI プロバイダーを **OpenShiftSDN** に設定しています。Cluster Version Operator (CVO) は、現在のクラスター状態に対する既知のリスクを継続的に評価します。該当するリスクがない場合は、更新を推奨します。リスクが一致する場合、それらの更新パスは **既知の問題のある更新** としてラベル付けされ、既知の問題への参照リンクが提供されます。参照リンクは、クラスター管理者がリスクを受け入れてクラスターの更新を続行するかどうかを決定するのに役立ちます。

Red Hat が条件付き更新リスクを宣言することを選択した場合、関連するすべてのチャンネルで同時に宣言します。条件付き更新リスクの宣言は、サポートされているチャンネルに更新がプロモートされる前、または後に発生する可能性があります。

1.3.1.7. クラスターに適したチャンネルの選択

適切なチャンネルを選択する際には、2つの点を決定する必要があります。

まず、クラスターの更新に必要なマイナーバージョンを選択します。現在のバージョンに一致するチャンネルを選択すると、z-stream 更新のみが適用され、機能更新は受信されません。現在のバージョンよりも新しいバージョンを含む利用可能なチャンネルを選択すると、更新を1回または複数回行うことで、対象のバージョンに更新されます。クラスターには、現在のバージョン、次のバージョン、または次の EUS バージョンに該当するチャンネルのみが提供されます。



注記

多数のマイナーバージョンをまたいだ更新を計画している場合は複雑になるので、どのチャンネルでも、単一の EUS から EUS を超えた更新の計画に対するサポートは提供していません。

次に、目的とするロールアウト戦略を選択する必要があります。Red Hat がリリース GA を宣言したらすぐに fast チャンネルから選択して更新するか、Red Hat がリリースを stable チャンネルにプロモートするのを待つかを選択できます。**fast-4.16** と **stable-4.16** で提供される更新の推奨はいずれも完全にサポートされており、同じように進行中のデータ分析からの恩恵を受けます。リリースを stable チャンネルに昇格させる前の昇格の遅延は、2つのチャンネルの唯一の違いです。最新の z-stream への更新は通常、1~2 週間以内に stable チャンネルに昇格されますが、最新のマイナーへの更新を最初にロールアウトするまでの時間は、通常 45~90 日と、はるかに長くなります。安定したチャンネルへの昇格を待つとスケジュール計画に影響する可能性があるため、希望のチャンネルを選択する際は昇格の遅延を考慮してください。

また、組織が fast チャンネルに永続的または一時的に移行する要因がいくつかあります。

- 遅滞なく、お使いの環境に影響を与えている既知の問題に対する特定の修正を適用する場合。
- 遅滞なく CVE 修正プログラムを適用する場合。CVE 修正によりリグレッションが発生する可能性があるため、CVE 修正を含む z-stream には引き続き昇格に時間がかかります。
- 内部テストプロセス。組織がリリースの認定に数週間かかる場合は、待たずに昇格プロセスと同時にテストすることを推奨します。こうすることで、Red Hat に対して遠隔測定からのシグナルが送られ、ロールアウトに考慮されるので、お客様に影響を与えている問題をより迅速に修正できます。

1.3.1.8. 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。

1.3.1.9. 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 **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-4.16** チャンネルから **fast-4.16** 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-4.16** チャンネルにいつでも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公開リリースの場合、**candidate-4.16** チャンネルから **fast-4.16** チャンネルに切り換えることができます。
- 常に、**fast-4.16** チャンネルから **stable-4.16** チャンネル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-4.16** にプロモートされるまでに最大で1日の遅延が生じる可能性があります。

関連情報

- [条件付きアップグレードパスに沿った更新](#)
- [クラスターに適したチャンネルの選択](#)

1.4. OPENSIFT CONTAINER PLATFORM の更新期間について

OpenShift Container Platform の更新期間は、デプロイメントのトポロジーによって異なります。このページは、更新期間に影響を与える要因を理解し、ご使用の環境でクラスターの更新にかかる時間を見積もるのに役立ちます。

1.4.1. 更新期間に影響する要因

次の要因は、クラスターの更新期間に影響を与える可能性があります。

- Machine Config Operator (MCO) による新しいマシン設定へのコンピューターノードの再起動
 - マシン設定プールの **MaxUnavailable** の値



警告

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

- Pod 中断バジェット (PDB) に設定されたレプリカの最小数またはパーセンテージ
- クラスター内のノード数
- クラスターノードの可用性

1.4.2. クラスターの更新フェーズ

OpenShift Container Platform では、クラスターの更新は2つのフェーズで行われます。

- Cluster Version Operator (CVO) ターゲット更新ペイロードのデプロイメント
- Machine Config Operator (MCO) ノードの更新

1.4.2.1. Cluster Version Operator ターゲット更新ペイロードのデプロイメント

Cluster Version Operator (CVO) は、ターゲットの更新リリースイメージを取得し、クラスターに適用します。Podとして実行されるすべてのコンポーネントはこのフェーズ中に更新されますが、ホストコンポーネントは Machine Config Operator (MCO) によって更新されます。このプロセスには60~120分かかる場合があります。



注記

更新の CVO フェーズでは、ノードは再起動されません。

1.4.2.2. Machine Config Operator ノードの更新

Machine Config Operator (MCO) は、新しいマシン設定を各コントロールプレーンとコンピューターノードに適用します。このプロセス中に、MCO はクラスターの各ノードで次の一連のアクションを実行します。

1. すべてのノードを遮断してドレインする
2. オペレーティングシステム (OS) を更新する

3. ノードを再起動します。
4. すべてのノードのコードを解除し、ノードでワークロードをスケジュールします



注記

ノードが遮断されている場合、ワークロードをそのノードにスケジュールすることはできません。

このプロセスが完了するまでの時間は、ノードやインフラストラクチャーの設定など、いくつかの要因によって異なります。このプロセスは、ノードごとに完了するまでに5分以上かかる場合があります。

MCOに加えて、次のパラメーターの影響を考慮する必要があります。

- コントロールプレーンノードの更新期間は予測可能であり、多くの場合、コンピュータノードよりも短くなります。これは、コントロールプレーンのワークロードが適切な更新と迅速なドレインに合わせて調整されているためです。
- Machine Config Pool (MCP) で **maxUnavailable** フィールドを 1 より大きい値に設定することで、コンピュータノードを並行して更新できます。MCO は、**maxUnavailable** で指定された数のノードを遮断し、それらを更新不可としてマークします。
- MCP で **maxUnavailable** を増やすと、プールがより迅速に更新されるのに役立ちます。ただし、**maxUnavailable** の設定が高すぎて、複数のノードが同時に遮断されている場合、レプリカを実行するスケジュール可能なノードが見つからないため、Pod 中断バジェット (PDB) で保護されたワークロードのドレインに失敗する可能性があります。MCP の **maxUnavailable** を増やす場合は、PDB で保護されたワークロードを排出できるように、スケジュール可能なノードがまだ十分であることを確認してください。
- 更新を開始する前に、すべてのノードが使用可能であることを確認する必要があります。ノードが利用できないと、**maxUnavailable** および Pod 中断バジェットに影響するため、利用できないノードがあると、更新期間に大きな影響を与える可能性があります。ターミナルからノードのステータスを確認するには、次のコマンドを実行します。

```
$ oc get node
```

出力例

```

NAME                                STATUS    ROLES    AGE    VERSION
ip-10-0-137-31.us-east-2.compute.internal Ready,SchedulingDisabled  worker  12d
v1.23.5+3afdacb
ip-10-0-151-208.us-east-2.compute.internal Ready                                master  12d
v1.23.5+3afdacb
ip-10-0-176-138.us-east-2.compute.internal Ready                                master  12d
v1.23.5+3afdacb
ip-10-0-183-194.us-east-2.compute.internal Ready                                worker  12d
v1.23.5+3afdacb
ip-10-0-204-102.us-east-2.compute.internal Ready                                master  12d
v1.23.5+3afdacb
ip-10-0-207-224.us-east-2.compute.internal Ready                                worker  12d
v1.23.5+3afdacb

```

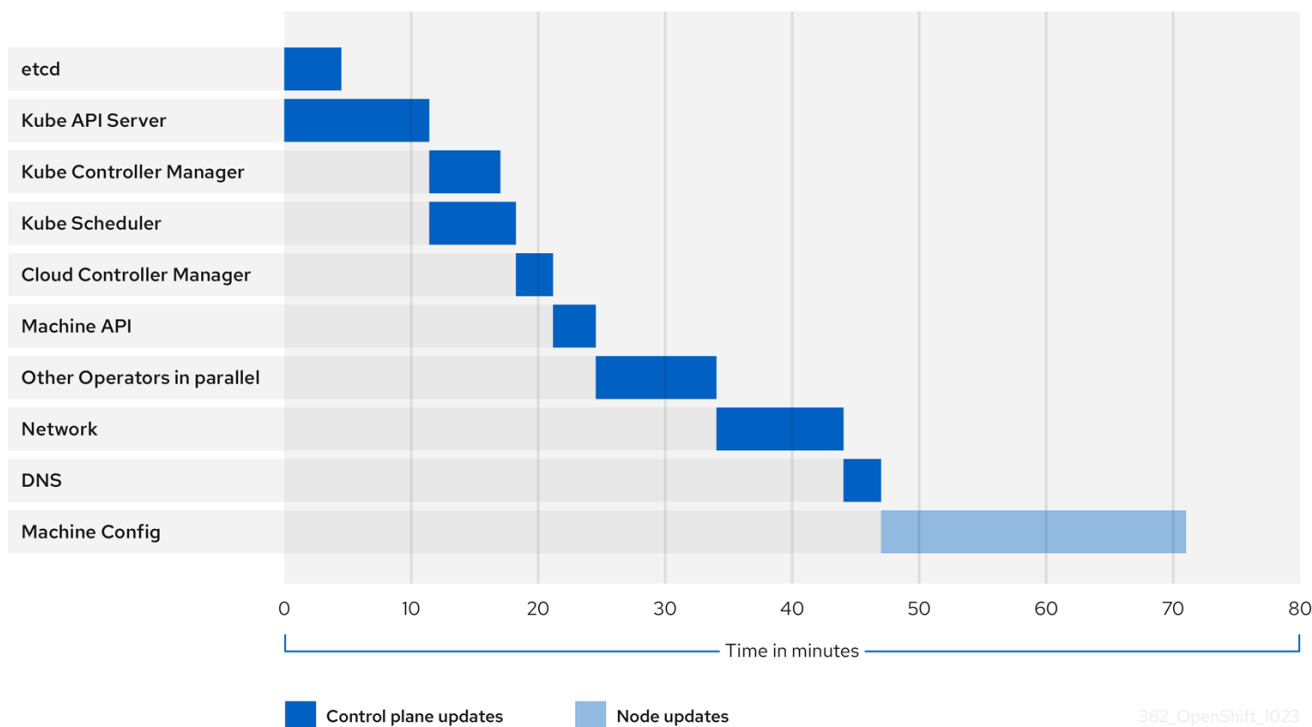
ノードのステータスが **NotReady** または **SchedulingDisabled** の場合、ノードは使用できず、更新期間に影響します。

Compute → Node を展開することで、Web コンソールの Administrator パースペクティブからノードのステータスを確認できます。

関連情報

- [マシン設定の概要](#)
- [Pod の中断バジェット](#)

1.4.2.3. クラスター Operator の更新期間の例



前の図は、クラスター Operator が新しいバージョンに更新するのにかかる時間の例を示しています。この例は、3 ノードの AWS OVN クラスターに基づいています。このクラスターには、正常なコンピュート **MachineConfigPool** があり、ドレインに時間がかかるワークロードがなく、4.13 から 4.14 に更新されます。



注記

- クラスターとその Operator の具体的な更新期間は、ターゲットバージョン、ノードの量、ノードにスケジュールされたワークロードの種類など、クラスターのいくつかの特性に基づいて変化する可能性があります。
- Cluster Version Operator などの一部のオペレーターは、短時間で自身を更新します。これらの Operator は図から省略されているか、「並列のその他の Operator」というラベルが付いたより広範な Operator のグループに含まれています。

各クラスター Operator には、それ自体の更新にかかる時間に影響する特性があります。たとえば、この例の Kube API Server Operator の更新には 11 分以上かかりました。これは、**kube-apiserver** が正常な終了サポートを提供しているためです。つまり、実行中の既存のリクエストは正常に完了できます。これにより、**kube-apiserver** のシャットダウンに時間がかかる可能性があります。この Operator の場合は、更新中のクラスター機能の中断を防止および制限するために、更新速度が犠牲になります。

Operator の更新期間に影響を与えるもう1つの特性は、Operator が DaemonSet を利用するかどうかです。Network Operator と DNS Operator はフルクラスター DaemonSet を利用するため、バージョン変更のデプロイメントに時間がかかる場合があります。これが、これらの Operator の更新に時間がかかる理由の1つです。

一部の Operator の更新期間は、クラスター自体の特性に大きく依存します。たとえば、Machine Config Operator の更新では、クラスター内の各ノードにマシン設定の変更が適用されます。多くのノードを含むクラスターは、ノードが少ないクラスターと比較して、Machine Config Operator の更新にかかる時間が長くなります。



注記

各クラスター Operator には、更新できるステージが割り当てられます。同じステージ内の Operator は同時に更新できますが、特定のステージの Operator は、前のステージがすべて完了するまで更新を開始できません。詳細は、関連情報セクションの更新中に「マニフェストが適用される方法について」を参照してください。

関連情報

- [OpenShift の更新の概要](#)
- [更新時のマニフェストの適用方法について](#)

1.4.3. クラスター更新時間の概算

同様のクラスターの履歴更新期間は、将来のクラスター更新の最適な概算を提供します。ただし、履歴データが利用できない場合は、次の規則を使用してクラスターの更新時間を概算することができます。

$$\text{Cluster update time} = \text{CVO target update payload deployment time} + (\# \text{ node update iterations} \times \text{MCO node update time})$$

ノード更新反復は、並行して更新される1つ以上のノードで設定されます。コントロールプレーンノードは常に、コンピューターノードと並行して更新されます。さらに、**maxUnavailable** 値に基づいて、1つ以上のコンピューターノードを並行して更新できます。



警告

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

例えば、更新時間を概算するには、3つのコントロールプレーンノードと6つのコンピューターノードを持つ OpenShift Container Platform クラスターがあり、各ホストの再起動に約5分かかるとします。



注記

特定のノードの再起動にかかる時間は、大幅に異なります。クラウドインスタンスでは、再起動に約1~2分かかる場合がありますが、物理的なベアメタルホストでは、再起動に15分以上かかる場合があります。

シナリオ 1:

コントロールプレーンとコンピューターノードの Machine Config Pool (MCP) の両方で **maxUnavailable** を **1** に設定すると、6つのコンピューターノードすべてが反復ごとに次々と更新されます。

Cluster update time = $60 + (6 \times 5) = 90$ minutes

シナリオ 2

コンピューターノード MCP の **maxUnavailable** を **2** に設定すると、2つのコンピューターノードが反復ごとに並行して更新されます。したがって、すべてのノードを更新するには合計3回の反復が必要です。

Cluster update time = $60 + (3 \times 5) = 75$ minutes



重要

maxUnavailable のデフォルト設定は、OpenShift Container Platform のすべての MCP で **1** です。コントロールプレーン MCP で **maxUnavailable** を変更しないことを推奨します。

1.4.4. Red Hat Enterprise Linux (RHEL) コンピューターノード

Red Hat Enterprise Linux (RHEL) コンピューターノードでは、ノードのバイナリーコンポーネントを更新するために **openshift-ansible** を追加で使用する必要があります。RHEL コンピューターノードの更新に費やされる実際の時間は、Red Hat Enterprise Linux CoreOS (RHCOS) コンピューターノードと大きく変わらないはずですが。

関連情報

- [RHEL コンピューターマシンの更新](#)

1.4.5. 関連情報

- [OpenShift Container Platform アーキテクチャー](#)
- [OpenShift Container Platform の更新](#)

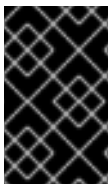
第2章 クラスターの更新の準備

2.1. OPENSIFT CONTAINER PLATFORM 4.16 への更新の準備

更新を正常に初期化するためにクラスター管理者が実行する必要がある管理タスクと、更新を確実に成功させるためのオプションのガイドラインについて詳しく説明します。

2.1.1. RHEL 9.2 マイクロアーキテクチャー要件の変更

OpenShift Container Platform は現在、RHEL 9.2 ホストオペレーティングシステムをベースとしています。マイクロアーキテクチャーの要件が x86_64-v2、Power9、Z14 に増加しました。[RHEL マイクロアーキテクチャー要件ドキュメント](#) を参照してください。この [ナレッジベース](#) に記載されている手順に従って、更新前に互換性を確認できます。



重要

正しいマイクロアーキテクチャー要件がないと、更新プロセスは失敗します。各アーキテクチャーに適切なサブスクリプションを購入してください。詳細は [Get Started with Red Hat Enterprise Linux - additional architectures](#) を参照してください。

2.1.2. Kubernetes API の削除

OpenShift Container Platform 4.16 は Kubernetes 1.29 を使用します。これにより、複数の非推奨 API が削除されました。

クラスター管理者は、クラスターを OpenShift Container Platform 4.15 から 4.16 にアップグレードする前に、手動で確認を行う必要があります。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 4.16 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、削除が予定されている使用中の API に対するクラスターの評価を実施し、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 4.15 クラスターを 4.16 に更新する前に、管理者の確認を提供する必要があります。

2.1.2.1. Kubernetes API の削除

OpenShift Container Platform 4.16 は Kubernetes 1.29 を使用します。これにより、以下の非推奨 API が削除されました。適切な API バージョンを使用するには、マニフェストと 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、詳細は [Kubernetes documentation](#) を参照してください。

表2.1 Kubernetes 1.29 から削除された API

| リソース | 削除された API | 移行先 | 大きな変更 |
|------------|--------------------------------------|--|-------|
| FlowSchema | flowcontrol.apiserver.k8s.io/v1beta2 | flowcontrol.apiserver.k8s.io/v1 または flowcontrol.apiserver.k8s.io/v1beta3 | いいえ |

| リソース | 削除された API | 移行先 | 大きな変更 |
|--------------------------------|--|---|-------|
| PriorityLevelConfigurati on | flowcontrol.apiserver.k8 s.io/v1beta2 | flowcontrol.apiserver.k8 s.io/v1 または flowcontrol.apiserver.k8 s.io/v1beta3 | はい |

2.1.2.2. 削除された API に対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。

2.1.2.2.1. 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。

アラートにはこの情報が含まれないため、**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します。さらに、API によってはこれらのアラートがトリガーされない場合もありますが、**APIRequestCount** がキャプチャーします。アラートは、機密性が低くなるように調整して、実稼働システムでのアラートの疲弊を回避します。

2.1.2.2.2. APIRequestCount を使用して削除された API の使用状況を特定

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。

手順

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

```
$ oc get apirequestcounts
```

出力例

| NAME | REMOVEDINRELEASE | REQUESTSINCURRENTHOURL | REQUESTSINLAST24H |
|--|------------------|------------------------|-------------------|
| ... | | | |
| flowschemas.v1beta2.flowcontrol.apiserver.k8s.io | 1.29 | 0 | 3 |
| ... | | | |
| prioritylevelconfigurations.v1beta2.flowcontrol.apiserver.k8s.io | 1.29 | 0 | |
| 1 | | | |
| ... | | | |

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はありません。

- **system:serviceaccount:kube-system:generic-garbage-collector** および **system:serviceaccount:kube-system:namespace-controller** ユーザーは、削除するリソースの検索時に登録されたすべての API を呼び出すので、結果に表示される可能性があります。
- **system:kube-controller-manager** および **system:cluster-policy-controller** ユーザーは、さまざまなポリシーを適用しながらすべてのリソースをウォークスルーするため、結果に表示される場合があります。

-o jsonpath を使用して結果をフィルタリング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

```
1.29 flowschemas.v1beta2.flowcontrol.apiserver.k8s.io
1.29 prioritylevelconfigurations.v1beta2.flowcontrol.apiserver.k8s.io
```

2.1.2.2.3. APIRequestCount を使用して削除された API を使用しているワークロードを特定

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

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。

手順

- 以下のコマンドを実行して **username** および **userAgent** を確認すると、API を使用しているワークロードの特定に役立ちます。

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

以下に例を示します。

```
$ oc get apirequestcounts flowschemas.v1beta2.flowcontrol.apiserver.k8s.io -o yaml
```

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

```
$ oc get apirequestcounts flowschemas.v1beta2.flowcontrol.apiserver.k8s.io \
  -o jsonpath='{range .status.currentHour..byUser[*]}{.byVerb[*].verb}{","}{.username}{","}
  {.userAgent}{"\n"}{end}' \
  | sort -k 2 -t, -u | column -t -s, -NVERBS,USERNAME,USERAGENT
```

出力例

```
VERBS  USERNAME                USERAGENT
create system:admin        oc/4.13.0 (linux/amd64)
list get system:serviceaccount:myns:default oc/4.16.0 (linux/amd64)
watch  system:serviceaccount:myns:webhook webhook/v1.0.0 (linux/amd64)
```

2.1.2.3. 削除された API 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。

2.1.2.4. 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 4.15 から 4.16 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。

手順

- 以下のコマンドを実行して、評価が完了し、クラスターが OpenShift Container Platform 4.16 で Kubernetes API を削除する準備ができていることを確認します。

```
$ oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.15-kube-1.29-api-removals-in-4.16":"true"}}' --type=merge
```

2.1.3. 条件付き更新のリスク評価

条件付き更新 は、使用可能な更新ターゲットですが、クラスターに適用される既知のリスクのため推奨

されません。Cluster Version Operator (CVO) は、OpenShift Update Service (OSUS) に定期的にクエリーを実行して、更新の推奨事項に関する最新のデータを取得します。ターゲットとなりうる一部の更新には、それに関連するリスクが含まれる可能性があります。

CVO は条件付きリスクを評価します。そのリスクがクラスターに当てはまらない場合、クラスターはそのターゲットバージョンを推奨される更新パスとして使用できます。リスクが当てはまると判断された場合、または何らかの理由で CVO がリスクを評価できない場合、クラスターはその更新ターゲットを条件付き更新として使用できます。

ターゲットバージョンに更新しようとしているときに条件付き更新が発生した場合は、クラスターをそのバージョンに更新するリスクを評価する必要があります。一般的に、そのターゲットバージョンに更新する必要性が特にない場合は、推奨される更新パスが Red Hat から提供されるまで待つのが最善です。

ただし、そのバージョンに更新する明確な理由がある場合 (たとえば重要な CVE を修正する必要がある場合など)、CVE を修正する利点が、更新によってクラスターに問題が発生するリスクを上回る可能性があります。以下のタスクを実行して、Red Hat の更新リスク評価に同意するか判断してください。

- 実稼働環境で問題なく更新を完了できると確信が持てるまで、非実稼働環境で幅広くテストしてください。
- 条件付き更新の説明に記載されているリンクを使用してバグを調査し、使用しているクラスターに問題を引き起こす可能性があるか判断します。リスクを把握するためにサポートが必要な場合は、Red Hat サポートにお問い合わせください。

関連情報

- [更新の可用性評価](#)

2.1.4. クラスター更新前の etcd バックアップ

etcd バックアップには、クラスターとそのすべてのリソースオブジェクトの状態が記録されます。現在機能不全状態にあるクラスターを復元できない障害シナリオでは、バックアップを使用してクラスターの状態の復元を試みることができます。

更新のコンテキストでは、更新によってクラスターの以前のバージョンに戻さないと修正できない壊滅的な状態が発生した場合に、クラスターの etcd 復元を試みることができます。etcd 復元は、実行中のクラスターにとって破壊的で不安定になる可能性があるため、最後の手段としてのみ使用してください。



警告

etcd 復元は重大な影響をもたらすため、ロールバックソリューションとして使用することは意図されていません。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 サポートにお問い合わせください。

etcd 復元の実行可能性に影響を与える要因がいくつかあります。詳細は、「etcd データのバックアップ」および「以前のクラスター状態への復元」を参照してください。

関連情報

- [etcd のバックアップ](#)
- [クラスターの直前の状態への復元](#)

2.1.5. クラスター更新のベストプラクティス

OpenShift Container Platform は、更新中のワークロードの中断を最小限に抑える堅牢な更新エクスペリエンスを提供します。更新要求時にクラスターがアップグレード可能な状態にない限り、更新は開始されません。

この設計では、更新を開始する前にいくつかの重要な条件を強制しますが、クラスターの更新が成功する可能性を高めるために実行できるアクションは多数あります。

2.1.5.1. OpenShift Update Service 推奨バージョンの選択

OpenShift Update Service (OSUS) は、クラスターがサブスクライブしているチャンネルなどをはじめとするクラスターの特性に基づき、更新に関する推奨を提示します。Cluster Version Operator は、これらの推奨事項を推奨される更新または条件付き更新として保存します。OSUS が推奨していないバージョンへの更新を試行できますが、推奨される更新パスに従うことで、ユーザーはクラスター上での既知の問題や予期せぬ結果の発生から保護されます。

更新を確実に成功させるためには、OSUS が推奨する更新ターゲットのみを選択してください。

2.1.5.2. クラスター上ですべての重大アラートに対処する

重大なアラートには、常に可能な限り早く対処する必要がありますが、クラスターの更新を開始する前にこれらのアラートに対処し、問題を解決することが特に重要です。更新を開始する前に重大アラートに対処しなければ、クラスターが問題のある状態に陥る可能性があります。

Web コンソールの **Administrator** パースペクティブで、**Observe** → **Alerting** に移動して重大なアラートを見つけます。

2.1.5.3. クラスターの状態が Upgradeable であることを確認する

1つ以上の Operator が1時間以上 **Upgradeable** 条件を **True** として報告しなかった場合、クラスター内で **ClusterNotUpgradeable** 警告アラートがトリガーされます。このアラートがパッチ更新をブロックすることはほぼありませんが、このアラートを解決し、すべての Operator が **Upgradeable** に対して **True** と報告するまで、マイナーバージョンの更新は実行できません。

Upgradeable 条件の詳細には、追加リソースセクションの「クラスター Operator の条件タイプ」を参照してください。

2.1.5.4. 十分な予備ノードが利用可能であることを確認する

特にクラスターの更新を開始する場合は、予備のノード容量がほとんどない、またはまったくない状態でクラスターを実行するべきではありません。ノードが実行されておらず、使用できない場合、クラスターのワークロードの中断を最小限に抑えつつ更新を実行するという能力が制限される可能性があります。

クラスターの **maxUnavailable** 仕様の設定値によっては、使用できないノードがある場合にクラスターはマシン設定の変更をノードに適用できない可能性があります。さらに、コンピューターノードに十分な予備容量がない場合、最初のノードが更新のためにオフラインになっている間、ワークロードを一時的に別のノードに移行できない可能性があります。

ノード更新が成功する可能性を高めるために、各ワーカープールに十分な使用可能なノードがあること、およびコンピューターノードに十分な予備容量があることを確認してください。



警告

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

2.1.5.5. クラスターの PodDisruptionBudget が適切に設定されていることを確認する

PodDisruptionBudget オブジェクトを使用して、常に使用可能でなければならない Pod レプリカの最小数または割合を定義できます。この設定により、クラスター更新などのメンテナンスタスク中にワークロードが中断されないように保護できます。

ただし、クラスターの更新中にノードがドレインおよび更新されないように、特定のトポロジーに対して **PodDisruptionBudget** を設定できます。

クラスターの更新を計画する際には、次の要因に対する **PodDisruptionBudget** オブジェクトの設定を確認してください。

- 高可用性ワークロードの場合は、**PodDisruptionBudget** で禁止されることなく一時的にオフラインにできるレプリカがあることを確認してください。
- 高可用性以外のワークロードの場合は、**PodDisruptionBudget** で保護されていないこと、または最終的にこれらのワークロードをドレインするための代替メカニズム (定期的な再起動や最終的な終了の保証など) があることを確認してください。

関連情報

- [クラスター Operator の状態タイプについて](#)

2.2. 手動で維持された認証情報でクラスターを更新する準備

手動で維持された認証情報をを含むクラスターの Cloud Credential Operator (CCO) の **upgradable** ステータスはデフォルトで **false** となります。

- 4.12 から 4.13 などのマイナーリリースの場合は、このステータスを使用することで、権限を更新して **CloudCredential** リソースにアノテーションを付けて権限が次のバージョンの要件に合わせて更新されていることを指定するまで、更新できなくなります。このアノテーションは、**Upgradable** ステータスを **True** に変更します。
- 4.13.0 から 4.13.1 などの z-stream リリースの場合には、権限は追加または変更されないため、更新はブロックされません。

手動で維持された認証情報を使用してクラスターを更新する前に、更新後の OpenShift Container Platform バージョンのリリースイメージにおける新規認証情報または変更された認証情報に対応する必要があります。

2.2.1. 手動で維持された認証情報を使用したクラスターの更新要件

手動で維持された認証情報を Cloud Credential Operator (CCO) で使用するクラスターを更新する前に、新しいリリースのクラウドプロバイダーリソースを更新する必要があります。

クラスターのクラウド認証情報管理が CCO ユーティリティ (**ccoctl**) を使用して設定されている場合、**ccoctl** ユーティリティを使用してリソースを更新します。**ccoctl** ユーティリティなしで手動モードを使用するように設定されたクラスターの場合、リソースを手動で更新する必要があります。

クラウドプロバイダーのリソースを更新したら、クラスターの **upgradeable-to** アノテーションを更新して、更新の準備ができていることを示す必要があります。



注記

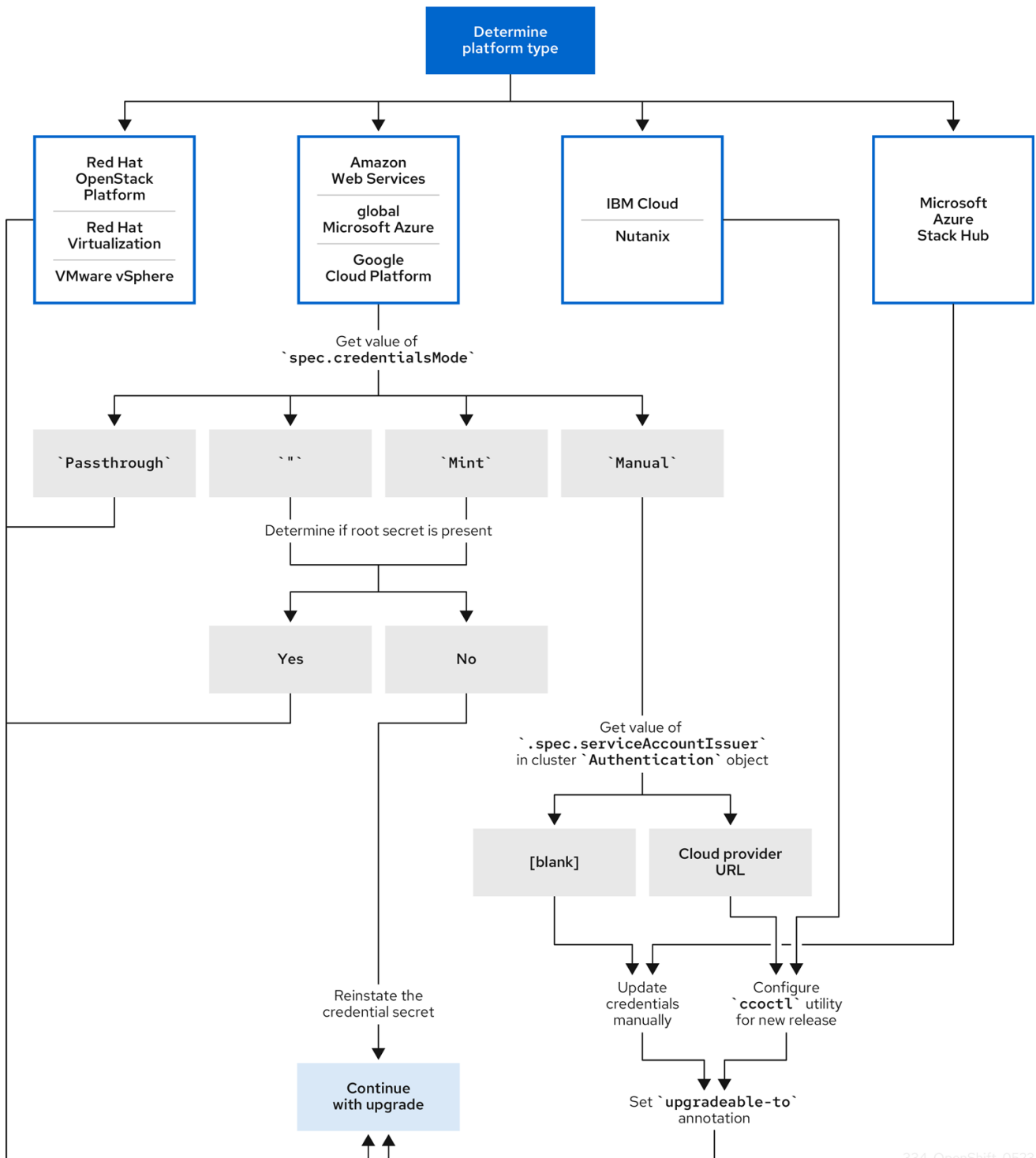
クラウドプロバイダーリソースと **upgradeable-to** アノテーションを更新するプロセスは、コマンドラインツールを使用しなければ完了できません。

2.2.1.1. プラットフォームタイプ別のクラウド認証情報の設定オプションと更新要件

一部のプラットフォームでは、CCO のモードを1つしか使用できません。そのようなプラットフォームにインストールされているクラスターの場合、プラットフォームタイプによって認証情報の更新要件が決まります。

CCO のモードを複数サポートしているプラットフォームの場合、クラスターが使用するように設定されているモードを判別し、その設定に必要なアクションを実行する必要があります。

図2.1 プラットフォームタイプ別の認証情報の更新要件



334_OpenShift_0523

Red Hat OpenStack Platform (RHOSP) と VMware vSphere

これらのプラットフォームは、手動モードでの CCO の使用をサポートしていません。これらのプラットフォーム上のクラスターでは、クラウドプロバイダーのリソース変更が自動的に処理され、**upgradeable-to** アノテーションへの更新は必要ありません。

これらのプラットフォーム上にあるクラスターの管理者は、更新プロセスの手動で維持された認証情報セクションをスキップする必要があります。

IBM Cloud と Nutanix

これらのプラットフォームにインストールされたクラスターは、**ccoctl** ユーティリティーを使用して設定されます。

これらのプラットフォーム上にあるクラスターの管理者は、以下のアクションを実行する必要があります。

1. 新しいリリースの **CredentialsRequest** カスタムリソース (CR) を抽出して準備します。
2. 新しいリリースの **ccoctl** ユーティリティーを設定し、それを使用してクラウドプロバイダーリソースを更新します。
3. **upgradeable-to** アノテーションで、クラスターの更新準備が完了したことを示します。

Microsoft Azure Stack Hub

これらのクラスターは、有効期間の長い認証情報と手動モードを使用し、**ccoctl** ユーティリティーは使用しません。

これらのプラットフォーム上にあるクラスターの管理者は、以下のアクションを実行する必要があります。

1. 新しいリリースの **CredentialsRequest** カスタムリソース (CR) を抽出して準備します。
2. 新しいリリースのクラウドプロバイダーリソースを手動で更新します。
3. **upgradeable-to** アノテーションで、クラスターの更新準備が完了したことを示します。

Amazon Web Services (AWS)、グローバル Microsoft Azure、Google Cloud Platform (GCP)

これらのプラットフォームにインストールされたクラスターは、複数の CCO モードをサポートします。

必要な更新プロセスは、クラスターが使用するように設定されたモードにより異なります。CCO がクラスターで使用するように設定されたモードが不明な場合は、Web コンソールまたは CLI を使用して判別できます。

関連情報

- [Web コンソールを使用した Cloud Credential Operator モードの判別](#)
- [CLI を使用した Cloud Credential Operator モードの判別](#)
- [認証情報要求リソースの抽出と準備](#)
- [Cloud Credential Operator について](#)

2.2.1.2. Web コンソールを使用した Cloud Credential Operator モードの判別

Cloud Credential Operator (CCO) がどのモードを使用するように設定されているかは、Web コンソールを使用して判別できます。



注記

複数の CCO モードをサポートするのは、Amazon Web Services (AWS)、グローバル Microsoft Azure、および Google Cloud Platform (GCP) クラスターのみです。

前提条件

- クラスター管理者パーミッションを持つ OpenShift Container Platform アカウントにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。
2. **Administration** → **Cluster Settings** に移動します。
3. **Cluster Settings** ページで、**Configuration** タブを選択します。
4. **Configuration resource** で **CloudCredential** を選択します。
5. **CloudCredential details** ページで、**YAML** タブを選択します。
6. YAML ブロックで、**spec.credentialsMode** の値を確認します。次の値が可能ですが、すべてのプラットフォームですべてがサポートされているわけではありません。
 - **"**: CCO はデフォルトモードで動作しています。この設定では、CCO は、インストール中に提供されたクレデンシャルに応じて、ミントモードまたはパススルーモードで動作しません。
 - **Mint**: CCO はミントモードで動作しています。
 - **Passthrough**: CCO はパススルーモードで動作しています。
 - **Manual**: CCO は手動モードで動作します。



重要

spec.credentialsMode が **"**、**Mint**、または **Manual** である AWS、GCP、またはグローバル Microsoft Azure クラスターの特定の設定を判断するには、さらに調査する必要があります。

AWS および GCP クラスターは、ルートシークレットが削除されたミントモードの使用をサポートします。クラスターが、mint モードを使用するように設定されている場合や、デフォルトで mint モードを使用するように設定されている場合、更新前に root シークレットがクラスターに存在するか確認する必要があります。

手動モードを使用する AWS、GCP、またはグローバル Microsoft Azure クラスターは、AWS STS、GCP Workload Identity、または Microsoft Entra Workload ID を使用してクラスターの外部からクラウド認証情報を作成および管理するように設定されている場合があります。クラスター **Authentication** オブジェクトを調べることで、クラスターがこの戦略を使用しているかどうかを判断できます。

7. mint モードのみを使用する AWS または GCP クラスター: クラスターがルートシークレットなしで動作しているかどうかを判断するには、**Workloads** → **Secrets** に移動し、クラウドプロバイダーのルートシークレットを探します。



注記

Project ドロップダウンが **All Projects** に設定されていることを確認します。

| プラットフォーム | シークレット名 |
|----------|------------------------|
| AWS | aws-creds |
| GCP | gcp-credentials |

- これらの値のいずれかが表示される場合、クラスターはルートシークレットが存在するミントモードまたはパススルーモードを使用しています。
 - これらの値が表示されない場合、クラスターはルートシークレットが削除されたミントモードで CCO を使用しています。
8. 手動モードのみを使用する AWS、GCP、または global Microsoft Azure クラスター: クラスターがクラスターの外部からクラウド認証情報を作成および管理するように設定されているかどうかを判断するには、クラスター **Authentication** オブジェクトの YAML 値を確認する必要があります。
- Administration** → **Cluster Settings** に移動します。
 - Cluster Settings** ページで、**Configuration** タブを選択します。
 - Configuration resource** で **Authentication** を選択します。
 - Authentication details** ページで、**YAML** タブを選択します。
 - YAML ブロックで、**.spec.serviceAccountIssuer** パラメーターの値を確認します。
 - クラウドプロバイダーに関連付けられた URL が含まれる値は、CCO がコンポーネントの短期認証情報を使用して手動モードを使用していることを示します。このクラスターは、クラスターの外からクラウド認証情報を作成および管理するために **ccoctl** ユーティリティーを使用して設定されています。
 - 空の値 ("") は、クラスターが手動モードで CCO を使用しているが、**ccoctl** ユーティリティーを使用して設定されていないことを示します。

次のステップ

- mint モードまたは passthrough モードで動作する CCO が含まれ、root シークレットが存在するクラスターを更新する場合、クラウドプロバイダーリソースを更新する必要はなく、更新プロセスの次の手順に進むことができます。
- クラスターが、root シークレットが削除された状態で mint モードの CCO を使用している場合、更新プロセスの次の手順に進む前に、管理者レベルの認証情報を使用して認証情報シークレットを元に戻す必要があります。
- クラスターが CCO ユーティリティー (**ccoctl**) を使用して設定されている場合、次のアクションを実行する必要があります。
 - 新しいリリースの **CredentialsRequest** カスタムリソース (CR) を抽出して準備します。
 - 新しいリリースの **ccoctl** ユーティリティーを設定し、それを使用してクラウドプロバイダーリソースを更新します。

- c. **upgradeable-to** アノテーションを更新して、クラスターの更新準備が完了していることを示します。
- クラスターが手動モードで CCO を使用しており、**ccoctl** ユーティリティを使用して設定されていない場合は、以下のアクションを実行する必要があります。
 - a. 新しいリリースの **CredentialsRequest** カスタムリソース (CR) を抽出して準備します。
 - b. 新しいリリースのクラウドプロバイダーリソースを手動で更新します。
 - c. **upgradeable-to** アノテーションを更新して、クラスターの更新準備が完了していることを示します。

関連情報

- [認証情報要求リソースの抽出と準備](#)

2.2.1.3. CLI を使用した Cloud Credential Operator モードの判別

CLI を使用して、Cloud Credential Operator (CCO) が使用するよう設定されているモードを判別できます。



注記

複数の CCO モードをサポートするのは、Amazon Web Services (AWS)、グローバル Microsoft Azure、および Google Cloud Platform (GCP) クラスターのみです。

前提条件

- クラスター管理者パーミッションを持つ OpenShift Container Platform アカウントにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

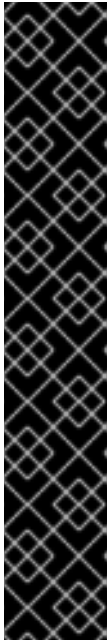
手順

1. **cluster-admin** ロールを持つユーザーとしてクラスターの **oc** にログインします。
2. CCO が使用するよう設定されているモードを確認するには、次のコマンドを入力します。

```
$ oc get cloudcredentials cluster \
  -o=jsonpath={.spec.credentialsMode}
```

すべてのプラットフォームですべてがサポートされているわけではありませんが、次の出力値が可能です。

- **":** CCO はデフォルトモードで動作しています。この設定では、CCO は、インストール中に提供されたクレデンシャルに応じて、ミントモードまたはパススルーモードで動作します。
- **Mint:** CCO はミントモードで動作しています。
- **Passthrough:** CCO はパススルーモードで動作しています。
- **Manual:** CCO は手動モードで動作します。



重要

spec.credentialsMode が **"**、**Mint**、または **Manual** である AWS、GCP、またはグローバル Microsoft Azure クラスターの特定の設定を判断するには、さらに調査する必要があります。

AWS および GCP クラスターは、ルートシークレットが削除されたミントモードの使用をサポートします。クラスターが、mint モードを使用するように設定されている場合や、デフォルトで mint モードを使用するように設定されている場合、更新前に root シークレットがクラスターに存在するか確認する必要があります。

手動モードを使用する AWS、GCP、またはグローバル Microsoft Azure クラスターは、AWS STS、GCP Workload Identity、または Microsoft Entra Workload ID を使用してクラスターの外部からクラウド認証情報を作成および管理するように設定されている場合があります。クラスター **Authentication** オブジェクトを調べることで、クラスターがこの戦略を使用しているかどうかを判断できます。

3. mint モードのみを使用する AWS または GCP クラスター: クラスターがルートシークレットなしで動作しているかどうかを判断するには、次のコマンドを実行します。

```
$ oc get secret <secret_name> \
-n=kube-system
```

<secret_name> は、AWS の場合は **aws-creds**、GCP の場合は **gcp-credentials** です。

ルートシークレットが存在する場合、このコマンドの出力はシークレットに関する情報を返します。エラーは、ルートシークレットがクラスターに存在しないことを示します。

4. 手動モードのみを使用する AWS、GCP、またはグローバル Microsoft Azure クラスター: クラスターの外部からクラウド認証情報を作成および管理するようにクラスターが設定されているかどうかを確認するには、次のコマンドを実行します。

```
$ oc get authentication cluster \
-o jsonpath \
--template='{ .spec.serviceAccountIssuer }'
```

このコマンドは、クラスター **Authentication** オブジェクトの **.spec.serviceAccountIssuer** パラメーターの値を表示します。

- クラウドプロバイダーに関連付けられた URL の出力は、CCO がコンポーネントの短期認証情報を使用して手動モードを使用していることを示します。このクラスターは、クラスターの外からクラウド認証情報を作成および管理するために **ccoctl** ユーティリティを使用して設定されています。
- 空の出力は、クラスターが手動モードで CCO を使用しているが、**ccoctl** ユーティリティを使用して設定されていないことを示します。

次のステップ

- mint モードまたは passthrough モードで動作する CCO が含まれ、root シークレットが存在するクラスターを更新する場合、クラウドプロバイダーリソースを更新する必要はなく、更新プロセスの次の手順に進むことができます。

- クラスターが、root シークレットが削除された状態で mint モードの CCO を使用している場合、更新プロセスの次の手順に進む前に、管理者レベルの認証情報を使用して認証情報シークレットを元に戻す必要があります。
- クラスターが CCO ユーティリティー (**ccoctl**) を使用して設定されている場合、次のアクションを実行する必要があります。
 - a. 新しいリリースの **CredentialsRequest** カスタムリソース (CR) を抽出して準備します。
 - b. 新しいリリースの **ccoctl** ユーティリティーを設定し、それを使用してクラウドプロバイダーリソースを更新します。
 - c. **upgradeable-to** アノテーションを更新して、クラスターの更新準備が完了していることを示します。
- クラスターが手動モードで CCO を使用しており、**ccoctl** ユーティリティーを使用して設定されていない場合は、以下のアクションを実行する必要があります。
 - a. 新しいリリースの **CredentialsRequest** カスタムリソース (CR) を抽出して準備します。
 - b. 新しいリリースのクラウドプロバイダーリソースを手動で更新します。
 - c. **upgradeable-to** アノテーションを更新して、クラスターの更新準備が完了していることを示します。

関連情報

- [認証情報要求リソースの抽出と準備](#)

2.2.2. 認証情報要求リソースの抽出と準備

Cloud Credential Operator (CCO) を使用するクラスターを手動モードで更新する前に、新しいリリース用の **CredentialsRequest** カスタムリソース (CR) を抽出して準備する必要があります。

前提条件

- 仕様している更新バージョンのバージョンに一致する OpenShift CLI (**oc**) をインストールしている。
- **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ている。

手順

1. 次のコマンドを実行して、適用する更新のプル仕様を取得します。

```
$ oc adm upgrade
```

このコマンドの出力には、次のような利用可能な更新のプル仕様が含まれます。

部分的な出力例

```
...
Recommended updates:

VERSION IMAGE
4.16.0 quay.io/openshift-release-dev/ocp-
```

```
release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056df6d01803
2
...
```

2. 次のコマンドを実行して、使用するリリースイメージを **\$RELEASE_IMAGE** 変数に設定します。

```
$ RELEASE_IMAGE=<update_pull_spec>
```

<update_pull_spec> は、使用するリリースイメージのプル仕様です。以下に例を示します。

```
quay.io/openshift-release-dev/ocp-
release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056df6d01803
2
```

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) のリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \1
  --to=<path_to_directory_for_credentials_requests> \2
```

1 **--included** パラメーターには、ターゲットリリースに特定のクラスター設定が必要とするマニフェストのみが含まれます。

2 **CredentialsRequest** オブジェクトを保存するディレクトリーへのパスを指定します。指定したディレクトリーが存在しない場合は、このコマンドによって作成されます。

このコマンドにより、それぞれの **CredentialsRequest** オブジェクトに YAML ファイルが作成されます。

4. リリースイメージの各 **CredentialsRequest** について、**spec.secretRef.namespace** フィールドのテキストと一致するネームスペースがクラスターに存在することを確認します。このフィールドには、クレデンシャルの設定を保持する生成されたシークレットが保存されます。

サンプル AWS **CredentialsRequest** オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: cloud-credential-operator-iam-ro
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - effect: Allow
        action:
          - iam:GetUser
          - iam:GetUserPolicy
          - iam:ListAccessKeys
```

```
resource: "*"
secretRef:
  name: cloud-credential-operator-iam-ro-creds
  namespace: openshift-cloud-credential-operator 1
```

- 1** このフィールドは、生成されたシークレットを保持するために存在する必要がある namespace を示します。

他のプラットフォームの **CredentialsRequest** CR も同様の形式ですが、プラットフォーム固有の異なる値があります。

5. クラスターが **spec.secretRef.namespace** で指定された名前の namespace をまだ持っていない **CredentialsRequest** CR については、次のコマンドを実行して namespace を作成します。

```
$ oc create namespace <component_namespace>
```

次のステップ

- クラスターのクラウド認証情報管理が CCO ユーティリティー (**ccoctl**) を使用して設定されている場合は、**ccoctl** ユーティリティーをクラスター更新用に設定し、それを使用してクラウドプロバイダーリソースを更新します。
- クラスターが **ccoctl** ユーティリティーを使用して設定されていない場合は、クラウドプロバイダーのリソースを手動で更新します。

関連情報

- [クラスター更新のための Cloud Credential Operator ユーティリティーの設定](#)
- [クラウドプロバイダーのリソースを手動で更新する](#)

2.2.3. クラスター更新のための Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) を手動モードで使用するクラスターをアップグレードして、クラスターの外からクラウド認証情報を作成および管理する場合は、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。



注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- クラスターは、クラスターの外からクラウド認証情報を作成および管理するために **ccoctl** ユーティリティーを使用して設定されている。
- OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) を抽出し、**spec.secretRef.namespace** フィールドのテキストと一致する namespace がクラスター内に存在している。

手順

1. 次のコマンドを実行して、OpenShift Container Platform リリースイメージの変数を設定します。

```
$ RELEASE_IMAGE=$(oc get clusterversion -o jsonpath={..desired.image})
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'  
$RELEASE_IMAGE -a ~/.pull-secret)
```



注記

\$RELEASE_IMAGE のアーキテクチャが、**ccoctl** ツールを使用する環境のアーキテクチャと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE \  
--file="/usr/bin/ccoctl.<rhel_version>" \1  
-a ~/.pull-secret
```

- 1 **<rhel_version>** には、ホストが使用する Red Hat Enterprise Linux (RHEL) のバージョンに対応する値を指定します。値が指定されていない場合は、デフォルトで **ccoctl.rhel8** が使用されます。次の値が有効です。

- **rhel8**: RHEL 8 を使用するホストの場合はこの値を指定します。
- **rhel9**: RHEL 9 を使用するホストの場合はこの値を指定します。

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl.<rhel_version>
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool
```

```
Usage:  
ccoctl [command]
```

```
Available Commands:
```



```
aws      Manage credentials objects for AWS cloud
azure    Manage credentials objects for Azure
gcp      Manage credentials objects for Google cloud
help     Help about any command
ibmcloud Manage credentials objects for IBM Cloud
nutanix  Manage credentials objects for Nutanix
```

Flags:

```
-h, --help  help for ccoctl
```

Use "ccoctl [command] --help" for more information about a command.

2.2.4. Cloud Credential Operator ユーティリティーを使用したクラウドプロバイダーリソースの更新

CCO ユーティリティー (**ccoctl**) を使用して設定された OpenShift Container Platform クラスターをアップグレードするプロセスは、インストール時にクラウドプロバイダーリソースを作成するプロセスに似ています。



注記

AWS クラスターでは、一部の **ccoctl** コマンドが AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) を抽出し、**spec.secretRef.namespace** フィールドのテキストと一致する namespace がクラスター内に存在している。
- リリースイメージから **ccoctl** バイナリーを抽出して設定している。

手順

1. **ccoctl** ツールを使用して、クラウドプロバイダーのコマンドを実行して、すべての **CredentialsRequest** オブジェクトを処理します。以下のコマンドは **CredentialsRequest** オブジェクトを処理します。

例2.1 Amazon Web Services (AWS)

```
$ ccoctl aws create-all \ ①
--name=<name> \ ②
--region=<aws_region> \ ③
--credentials-requests-dir=<path_to_credentials_requests_directory> \ ④
--output-dir=<path_to_ccoctl_output_dir> \ ⑤
--create-private-s3-bucket ⑥
```

- ① AWS リソースを個別に作成するには、「カスタマイズを使用した AWS へのクラスターのインストール」コンテンツの「AWS リソースの個別の作成」手順を使用します。このオプションは、AWS リソースを変更する前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合、または **ccoctl** ツールが AWS リソースを自動的に

に作成するために使用するプロセスが組織の要件を満たしていない場合に役立つ可能性があります。

- 2 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
- 3 クラウドリソースが作成される AWS リージョンです。
- 4 コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
- 5 オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
- 6 オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

例2.2 Google Cloud Platform (GCP)

```
$ ccoctl gcp create-all \
--name=<name> \ 1
--region=<gcp_region> \ 2
--project=<gcp_project_id> \ 3
--credentials-requests-dir=<path_to_credentials_requests_directory> \ 4
--output-dir=<path_to_ccoctl_output_dir> \ 5
```

- 1 トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名を指定します。
- 2 クラウドリソースを作成する GCP リージョンを指定します。
- 3 クラウドリソースを作成する GCP プロジェクト ID を指定します。
- 4 GCP サービスアカウントを作成するには、**CredentialsRequest** マニフェストのファイルが含まれるディレクトリーを指定します。
- 5 オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。

例2.3 IBM Cloud

```
$ ccoctl ibmcloud create-service-id \
--credentials-requests-dir=<path_to_credential_requests_directory> \ 1
--name=<cluster_name> \ 2
--output-dir=<installation_directory> \ 3
--resource-group-name=<resource_group_name> \ 4
```

- 1 コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
- 2 OpenShift Container Platform クラスターの名前を指定します。
- 3 オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
- 4 オプション: アクセスポリシーの範囲に使用されるリソースグループの名前を指定します。

例2.4 Microsoft Azure

```
$ ccoctl azure create-managed-identities \
  --name <azure_infra_name> \ 1
  --output-dir ./output_dir \
  --region <azure_region> \ 2
  --subscription-id <azure_subscription_id> \ 3
  --credentials-requests-dir <path_to_directory_for_credentials_requests> \
  --issuer-url "${OIDC_ISSUER_URL}" \ 4
  --dnszone-resource-group-name <azure_dns_zone_resourcegroup_name> \ 5
  --installation-resource-group-name "${AZURE_INSTALL_RG}" \ 6
```

- 1 **name** パラメーターの値は、Azure リソースグループを作成するために使用します。新しい Azure リソースグループを作成する代わりに既存の Azure リソースグループを使用するには、**--oidc-resource-group-name** を指定し、その値として既存のグループ名を使用します。
- 2 既存のクラスターのリージョンを指定します。
- 3 既存のクラスターのサブスクリプション ID を指定します。
- 4 既存のクラスターから OIDC 発行者 URL を指定します。この値は、以下のコマンドを実行して取得できます。

```
$ oc get authentication cluster \
  -o jsonpath \
  --template='{ .spec.serviceAccountIssuer }'
```

- 5 DNS ゾーンを含むリソースグループの名前を指定します。
- 6 Azure リソースグループを指定します。この値は、以下のコマンドを実行して取得できます。

```
$ oc get infrastructure cluster \
  -o jsonpath \
  --template '{ .status.platformStatus.azure.resourceGroupName }'
```

例2.5 Nutanix

```
$ ccoctl nutanix create-shared-secrets \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --credentials-source-filepath=<path_to_credentials_file> 3
```

- 1 コンポーネント **CredentialsRequests** オブジェクトのファイルを含むディレクトリへのパスを指定します。
- 2 オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリにオブジェクトを作成します。
- 3 オプション: 認証情報データ YAML ファイルを含むディレクトリを指定します。デフォルトでは、**ccoctl** はこのファイルが **<home_directory>/.**nutanix/credentials**** があると想定します。

OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトで定義されているとおり、**ccoctl** は **CredentialsRequest** オブジェクトごとに必要なプロバイダーリソースと権限ポリシーを作成します。

2. 次のコマンドを実行して、シークレットをクラスターに適用します。

```
$ ls <path_to_ccoctl_output_dir>/manifests/*-credentials.yaml | xargs -l{} oc apply -f {}
```

検証

クラウドプロバイダーにクエリーを実行することで、必要なプロバイダーのリソースと権限ポリシーが作成されていることを確認できます。詳細は、適切なクラウドプロバイダーのドキュメントでロールまたはサービスアカウントの一リストを参照してください。

次のステップ

- **upgradeable-to** アノテーションを更新して、クラスターをアップグレードする準備ができていることを示します。

関連情報

- [クラスターがアップグレードの準備ができていることを示す](#)

2.2.5. クラウドプロバイダーのリソースを手動で更新する

手動でメンテナンスされる認証情報でクラスターをアップグレードする前に、アップグレードするリリースイメージ用に新しい認証情報のシークレットを作成する必要があります。また、既存の認証情報に必要なアクセス許可を確認し、それらのコンポーネントの新しいリリースでの新しいアクセス許可要件に対応する必要があります。

前提条件

- OpenShift Container Platform リリースイメージから **CredentialsRequest** カスタムリソース (CR) を抽出し、**spec.secretRef.namespace** フィールドのテキストと一致する namespace がクラスター内に存在している。

手順

1. 新しいリリースイメージで追加される **CredentialsRequest** カスタムリソースのシークレットを含む YAML ファイルを作成します。シークレットは、それぞれの **CredentialsRequest** オブジェクトについて **spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。

例2.6 サンプル AWS YAML ファイル

シークレットを含むサンプル AWS CredentialsRequest オブジェクト

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - effect: Allow
        action:
          - s3:CreateBucket
          - s3>DeleteBucket
        resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...

```

サンプル AWS Secret オブジェクト

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>

```

例2.7 サンプル Azure YAML ファイル



注記

Global Azure と Azure Stack Hub は、同じ **CredentialsRequest** オブジェクトとシークレット形式を使用します。

シークレットを含むサンプル Azure CredentialsRequest オブジェクト

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
      - role: Contributor
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...

```

サンプル Azure Secret オブジェクト

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>
  azure_client_secret: <base64_encoded_azure_client_secret>
  azure_tenant_id: <base64_encoded_azure_tenant_id>
  azure_resource_prefix: <base64_encoded_azure_resource_prefix>
  azure_resourcegroup: <base64_encoded_azure_resourcegroup>
  azure_region: <base64_encoded_azure_region>

```

例2.8 サンプル GCP YAML ファイル

シークレットを含むサンプル GCP CredentialsRequest オブジェクトの

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
      - roles/iam.securityReviewer
      - roles/iam.roleViewer
    skipServiceCheck: true
  ...

```

```
secretRef:
  name: <component_secret>
  namespace: <component_namespace>
...
```

サンプル GCP Secret オブジェクト

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```

- シークレットに保存される既存の認証情報の **CredentialsRequest** カスタムリソースにパーミッション要件を変更した場合は、必要に応じてパーミッションを更新します。

次のステップ

- **upgradeable-to** アノテーションを更新して、クラスターをアップグレードする準備ができていることを示します。

関連情報

- [AWS の長期認証情報を手動で作成](#)
- [Azure の長期認証情報を手動で作成](#)
- [Azure Stack Hub の長期認証情報を手動で作成する](#)
- [GCP の長期認証情報を手動で作成](#)
- [クラスターがアップグレードの準備ができていることを示す](#)

2.2.6. クラスターがアップグレードの準備ができていることを示す

手動で維持された認証情報をを含むクラスターの Cloud Credential Operator (CCO) の **upgradable** ステータスはデフォルトで **false** となります。

前提条件

- アップグレード先のリリースイメージについて、手動で、または Cloud Credential Operator ユーティリティ (**ccctl**) を使用して、新しい認証情報を処理しました。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **cluster-admin** ロールを持つユーザーとしてクラスターの **oc** にログインします。
2. 次のコマンドを実行して **CloudCredential** リソースを編集し、**metadata** フィールド内に **upgradeable-to** アノテーションを追加します。

```
$ oc edit cloudcredential cluster
```

追加するテキスト

```
...
  metadata:
    annotations:
      cloudcredential.openshift.io/upgradeable-to: <version_number>
  ...
```

<version_number> はアップグレード先のバージョンで、形式は **xyz** です。たとえば、OpenShift Container Platform 4.12.2 には **4.12.2** を使用します。

アノテーションを追加してから、upgradeable のステータスが変更されるまで、数分かかる場合があります。

検証

1. Web コンソールの **Administrator** パースペクティブで、**Administration** → **Cluster Settings** に移動します。
2. CCO ステータスの詳細を表示するには、**Cluster Operators** リストで **cloud-credential** をクリックします。
 - **Conditions** セクションの **Upgradeable** ステータスが **False** の場合に、**upgradeable-to** アノテーションに間違いがないことを確認します。
3. **Conditions** セクションの **Upgradeable** ステータスが **True** の場合、OpenShift Container Platform のアップグレードを開始します。

2.3. KERNEL MODULE MANAGEMENT (KMM) モジュールのプリフライト検証

KMM モジュールが適用されたクラスターでアップグレードを実行する前に、KMM を使用してインストールされたカーネルモジュールが、クラスターのアップグレード、および場合によってはカーネルのアップグレード後に、ノードにインストールできることを確認する必要があります。プリフライトは、クラスターにロードされたすべての **Module** を並行して検証しようとしています。プリフライトは、ある **Module** の検証が完了するのを待たずに、別の **Module** の検証を開始します。

2.3.1. 検証のキックオフ

プリフライト検証は、クラスター内に **PreflightValidationOCP** リソースを作成することによってトリガーされます。この仕様には、次の2つのフィールドが含まれます。

releaseImage

クラスターがアップグレードされる OpenShift Container Platform バージョンのリリースイメージの名前を提供する必須フィールド。

pushBuiltImage

true の場合、ビルドおよび署名の検証中に作成されたイメージがリポジトリにプッシュされます。このフィールドはデフォルトで **false** です。

2.3.2. 検証のライフサイクル

プリフライト検証は、クラスターにロードされたすべてのモジュールの検証を試みます。検証が成功した後、プリフライトは **module** リソースでの検証の実行を停止します。モジュールの検証が失敗した場合は、モジュール定義を変更することができ、Preflight は次のループで再度モジュールの検証を試行します。

追加のカーネルにプリフライト検証を実行する場合は、そのカーネル用に別の **PreflightValidationOCP** リソースを作成する必要があります。すべてのモジュールが検証されたら、**PreflightValidationOCP** リソースを削除することを推奨します。

2.3.3. 検証のステータス

PreflightValidationOCP リソースは、検証を試行している、または試行したクラスター内の各モジュールのステータスと進行状況を **.status.modules** リストに報告します。そのリストの要素には次のフィールドが含まれます。

lastTransitionTime

Module リソースのステータスが、別のステータスに移行した最後の時刻。これは、基礎となるステータスが変更されたときになります。不明な場合には、API フィールドが変更された時点を使用することも可能です。

name

Module リソースの名前。

namespace

Module リソースの namespace。

statusReason

ステータスに関する動詞の説明。

verificationStage

実行されている検証段階について説明します。

- **image**: イメージの存在確認
- **build**: ビルドプロセスの検証
- **sign**: 署名プロセスの検証

verificationStatus

モジュール検証のステータス:

- **true**: 検証済みです
- **false**: 検証に失敗しました
- **error**: 検証プロセス中にエラーが発生しました
- **unknown**: 検証が開始されていません

2.3.4. モジュールごとのプリフライト検証ステージ

プリフライトは、クラスター内に存在するすべての KMM モジュールに次の検証を実行します。

1. イメージの検証ステージ
2. ビルドの検証ステージ

3. 署名の検証ステージ

2.3.4.1. イメージの検証ステージ

イメージの検証は常に、実行されるプリフライト検証の最初のステージです。イメージの検証が成功した場合、その特定のモジュールで他の検証は実行されません。

イメージの検証は、次の2つのステージで設定されます。

1. イメージの存在とアクセシビリティ。コードは、モジュール内のアップグレードされたカーネル用に定義されたイメージにアクセスし、そのマニフェストを取得しようとします。
2. 今後の **modprobe** の実行のために、**Module** で定義されたカーネルモジュールが正しいパスに存在することを確認します。この検証が成功した場合は、カーネルモジュールが正しい Linux ヘッダーでコンパイルされた可能性が高いです。正しいパスは `<dirname>/lib/modules/<upgraded_kernel>/` です。

2.3.4.2. ビルドの検証ステージ

ビルドの検証は、イメージの検証が失敗し、**Module** にアップグレードされたカーネルに関連する **build** セクションがある場合のみ、実行されます。ビルドの検証は、ビルドジョブを実行し、それが正常に終了したことを検証しようとします。

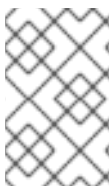


注記

次に示すように、**depmod** を実行する場合は、カーネルバージョンを指定する必要があります。

```
$ RUN depmod -b /opt ${KERNEL_VERSION}
```

PushBuiltImage フラグが **PreflightValidationOCP** カスタムリソース (CR) で定義されている場合は、結果のイメージをリポジトリにプッシュしようとします。結果のイメージ名は、**Module** CR の **containerImage** フィールドの定義から取得されます。



注記

アップグレードされたカーネルに **sign** セクションが定義されている場合、結果のイメージは **Module** CR の **containerImage** フィールドではなく、一時的なイメージ名になります。これは、結果のイメージが Sign フローの製品である必要があるためです。

2.3.4.3. 署名の検証ステージ

署名検証は、イメージ検証が失敗した場合にのみ実行されます。**Module** リソースにはアップグレードカーネルに関連する **Sign** セクションがあり、**Module** にアップグレードされたカーネルに関連する **build** セクションがあった場合は、ビルド検証が正常に終了します。署名検証では、署名ジョブを実行し、それが正常に完了したかどうかを検証します。

PushBuiltImage フラグが **PreflightValidationOCP** CR で定義されている場合、署名の検証は結果のイメージをレジストリにプッシュしようとします。結果のイメージは、常に **Module** の **ContainerImage** フィールドで定義されたイメージです。入力イメージは、Build ステージの出力、または **UnsignedImage** フィールドで定義されたイメージのいずれかです。



注記

build セクションが存在する場合、**sign** セクションの入カイメージは、**build** セクションの出カイメージになります。したがって、入カイメージを **sign** セクションで使用できるようにするには、**PreflightValidationOCP** CR で **PushBuiltImage** フラグを定義する必要があります。

2.3.5. PreflightValidationOCP リソースの例

このセクションでは、YAML 形式の **PreflightValidationOCP** リソースの例を示します。

この例では、現在存在するすべてのモジュールを、OpenShift Container Platform リリース 4.11.18 (以下のリリースイメージが指す) に含まれる今後のカーネルバージョンに対して検証します。

```
quay.io/openshift-release-dev/ocp-  
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
```

.spec.pushBuiltImage が **true** に設定されているため、KMM はビルド/署名の結果のイメージを定義済みのリポジトリにプッシュします。

```
apiVersion: kmm.sigs.x-k8s.io/v1beta2  
kind: PreflightValidationOCP  
metadata:  
  name: preflight  
spec:  
  releaseImage: quay.io/openshift-release-dev/ocp-  
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863  
  pushBuiltImage: true
```

第3章 クラスターの更新の実行

3.1. CLI を使用したクラスターの更新

OpenShift CLI (**oc**) を使用して、OpenShift Container Platform クラスターでマイナーバージョンおよびパッチの更新を実行できます。

3.1.1. 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。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、クラスターを以前の状態に復元する必要がある場合に備えて、最新の [etcd バックアップ](#) を用意している。
- Pod 障害により永続ボリュームを復元する必要がある場合に備えて、最新の [Container Storage Interface \(CSI\) ボリュームのスナップショット](#) を用意している。
- RHEL7 ワーカーは、RHEL8 または RHCOS ワーカーに置き換えられます。Red Hat は、RHEL7 から RHEL8 への RHEL ワーカーのインプレース更新をサポートしていません。このようなホストは、オペレーティングシステムをクリーンインストールして置き換える必要があります。
- Operator Lifecycle Manager (OLM) を通じて以前にインストールされたすべての Operator を、ターゲットリリースと互換性のあるバージョンに更新している。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新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。[インストール済み Operator の更新](#) を参照し、互換性を確認する方法の詳細を確認して、インストールされている Operator を必要に応じて更新してください。
- 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 を一時停止できる。
- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは [手動で維持された認証情報でクラスターを更新する準備](#) 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更新ができるように、すべての **Upgradeable=False** 条件に対応してください。アラートは、アップグレードできない1つ以上のクラスター Operator がある場合に **Cluster Settings** ページの上部に表示されます。引き続き、現在使用しているマイナーリリースについて、次に利用可能なパッチ更新に更新できます。
- Kubernetes 1.28 で削除された API のリストを確認し、影響を受けるすべての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、管理者に承認を提供します。詳細は、[OpenShift Container Platform 4.16 への更新の準備](#) を参照してください。
- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合は、更新プロセス中に中断が発生する可能性があります。**PodDisruptionBudget** で **minAvailable** が1に設定されている場合、**削除** プロセスをブロックする可能性がある保留中のマシン設定を適用するためにノードがドレインされます。複数のノードが再起動された場合に、すべての Pod が1つのノードでのみ実行される可能性があり、**PodDisruptionBudget** フィールドはノードのドレインを防ぐことができます。



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 サポートにお問い合わせください。
- **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](#)

3.1.2. MachineHealthCheck リソースの一時停止

更新プロセスで、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**) がインストールされている。

手順

1. 一時停止する利用可能なすべての **MachineHealthCheck** リソースをリスト表示するには、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=""
```

アノテーション付きの **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-
```

3.1.3. 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、更新に失敗した場合は、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。
 - 更新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 が短時間ですく解決します。

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は、[Machine Config Operator について](#) を参照してください。

3.1.4. CLI を使用したクラスターの更新

OpenShift CLI (**oc**) を使用して、クラスターの更新を確認および要求できます。

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

- 仕様している更新バージョンのバージョンに一致する OpenShift CLI (**oc**) をインストールしている。
- 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ている。
- すべての **MachineHealthCheck** リソースを一時停止している。

手順

- 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 4.13.10
Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.13 (available channels: candidate-4.13, candidate-4.14, fast-4.13, stable-4.13)
Recommended updates:
VERSION  IMAGE
4.13.14  quay.io/openshift-release-dev/ocp-release@sha256:406fcc160c097f61080412afca7fd65284ac8741ac7ad5b480e304aba73674b
4.13.13  quay.io/openshift-release-dev/ocp-release@sha256:d62495768e335c79a215ba56771ff5ae97e3cbb2bf49ed8fb3f6cefabc0f17
4.13.12  quay.io/openshift-release-dev/ocp-release@sha256:73946971c03b43a0dc6f7b0946b26a177c2f3c9d37105441315b4e3359373a55
4.13.11  quay.io/openshift-release-dev/ocp-release@sha256:e1c2377fdae1d063aaddc753b99acf25972b6997ab9a0b7e80cfef627b9ef3dd
```




注記

- 推奨される更新がない場合でも、既知の問題がある更新がまだ利用できる可能性があります。詳細は、[条件付き更新パスに沿った更新](#)を参照してください。
- **EUS-to-EUS** チャンネル更新を実行する方法の詳細と情報は、関連情報セクションにリストされている **EUS-to-EUS アップグレードを実行するための準備** ページを参照してください。

2. 組織の要件に基づいて、適切な更新チャンネルを設定します。たとえば、チャンネルを **stable-4.13** または **fast-4.13** に設定できます。チャンネルの詳細は、追加リソースセクションにリストされている [更新チャンネルとリリースについて](#) を参照してください。

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

たとえば、チャンネルを **stable-4.16** に設定するには、以下を実行します。

```
$ oc adm upgrade channel stable-4.16
```



重要

実稼働クラスターの場合、**stable-***、**eus-*** 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。



注記

次のマイナーバージョンに移行する準備ができたなら、そのマイナーバージョンに対応するチャンネルを選択します。更新チャンネルの宣言が早ければ早いほど、クラスターはターゲットバージョンへの更新パスをより効果的に推奨できます。クラスターは、利用可能なすべての可能な更新プログラムを評価し、最適な更新プログラムの推奨事項を選択するために、しばらく時間がかかる場合があります。更新の推奨事項は、その時点で利用可能な更新オプションに基づいているため、時間の経過とともに変化する可能性があります。

ターゲットマイナーバージョンへの更新パスが表示されない場合は、次のマイナーバージョンがパスで利用可能になるまで、現在のバージョンの最新のパッチリリースにクラスターを更新し続けます。

3. 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、**oc adm upgrade** コマンドの出力から取得した更新バージョンです。



重要

oc adm upgrade --help を使用する場合、**--force** のオプションがリストされています。**--force** オプションを使用すると、リリースの検証や前提条件のチェックをはじめとするクラスター側のガードがバイパスされるため、これは **可能な限り使用しないでください**。**--force** を使用しても、更新が成功することは保証されません。ガードをバイパスすると、クラスターが危険にさらされます。

4. クラスターバージョン Operator を確認します。

```
$ oc adm upgrade
```

5. 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 adm upgrade
```

出力例

```
Cluster version is <version>
```

```
Upstream is unset, so the cluster will use an appropriate default.
```

```
Channel: stable-<version> (available channels: candidate-<version>, eus-<version>, fast-<version>, stable-<version>)
```

```
No updates available. You may force an update to a specific release image, but doing so might not be supported and might result in downtime or data loss.
```

6. クラスターを次のマイナーバージョン (バージョン Xy から X.(y+1) など) に更新する場合は、新しい機能に依存するワークロードをデプロイする前に、ノードが更新されていることを確認することが推奨されます。

```
$ oc get nodes
```

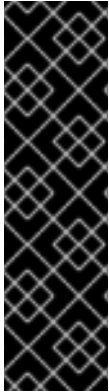
出力例

| NAME | STATUS | ROLES | AGE | VERSION |
|------------------------------|--------|--------|-----|---------|
| ip-10-0-168-251.ec2.internal | Ready | master | 82m | v1.29.4 |
| ip-10-0-170-223.ec2.internal | Ready | master | 82m | v1.29.4 |
| ip-10-0-179-95.ec2.internal | Ready | worker | 70m | v1.29.4 |
| ip-10-0-182-134.ec2.internal | Ready | worker | 70m | v1.29.4 |
| ip-10-0-211-16.ec2.internal | Ready | master | 82m | v1.29.4 |
| ip-10-0-250-100.ec2.internal | Ready | worker | 69m | v1.29.4 |

3.1.5. oc adm upgrade status を使用してクラスターの更新に関する情報を取得する (テクノロジープレビュー)

クラスターを更新するときは、更新の進行状況を把握しておく役立ちます。**oc adm upgrade** コマンドは更新のステータスに関する限定的な情報を返しますが、このリリースでは **oc adm upgrade status** コマンドがテクノロジープレビュー機能として導入されています。このコマンドは、**oc adm upgrade** コマンドからステータス情報を分離し、コントロールプレーンのステータスやワーカーノードの更新など、クラスターの更新に関する特定の情報を提供します。

oc adm upgrade status コマンドは読み取り専用であり、クラスターの状態を変更することはありません。

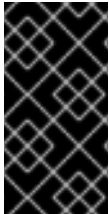


重要

oc adm upgrade status コマンドはテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

oc adm upgrade status コマンドは、バージョン 4.12 から最新のサポートされているリリースまでのクラスターに使用できます。



重要

クラスターはテクノロジープレビュー対応のクラスターである必要はありませんが、**OC_ENABLE_CMD_UPGRADE_STATUS** テクノロジープレビュー環境変数を有効にする必要があります。有効にしないと、OpenShift CLI (**oc**) がコマンドを認識せず、機能を使用できなくなります。

手順

1. 次のコマンドを実行して、**OC_ENABLE_CMD_UPGRADE_STATUS** 環境変数を **true** に設定します。

```
$ export OC_ENABLE_CMD_UPGRADE_STATUS=true
```

2. **oc adm upgrade status** コマンドを実行します。

```
$ oc adm upgrade status
```

例3.1 更新が正常に進行した場合の出力例

```
= Control Plane =
Assessment:   Progressing
Target Version: 4.14.1 (from 4.14.0)
Completion:   97%
Duration:     54m
Operator Status: 32 Healthy, 1 Unavailable

Control Plane Nodes
NAME                                ASSESSMENT  PHASE    VERSION  EST
MESSAGE
ip-10-0-53-40.us-east-2.compute.internal  Progressing  Draining  4.14.0  +10m
ip-10-0-30-217.us-east-2.compute.internal  Outdated    Pending   4.14.0  ?
ip-10-0-92-180.us-east-2.compute.internal  Outdated    Pending   4.14.0  ?

= Worker Upgrade =
```

```

= Worker Pool =
Worker Pool: worker
Assessment: Progressing
Completion: 0%
Worker Status: 3 Total, 2 Available, 1 Progressing, 3 Outdated, 1 Draining, 0 Excluded, 0
Degraded

Worker Pool Nodes
NAME                ASSESSMENT  PHASE    VERSION  EST
MESSAGE
ip-10-0-4-159.us-east-2.compute.internal Progressing  Draining  4.14.0  +10m
ip-10-0-20-162.us-east-2.compute.internal Outdated    Pending   4.14.0  ?
ip-10-0-99-40.us-east-2.compute.internal  Outdated    Pending   4.14.0  ?

= Worker Pool =
Worker Pool: infra
Assessment: Progressing
Completion: 0%
Worker Status: 1 Total, 0 Available, 1 Progressing, 1 Outdated, 1 Draining, 0 Excluded, 0
Degraded

Worker Pool Node
NAME                ASSESSMENT  PHASE    VERSION  EST
MESSAGE
ip-10-0-4-159-infra.us-east-2.compute.internal Progressing  Draining  4.14.0  +10m

= Update Health =
SINCE LEVEL IMPACT MESSAGE
14m4s Info  None  Update is proceeding well

```

この情報を利用して、更新をどのように進めるかについて十分な情報に基づいた決定を下すことができます。

関連情報

- [EUS から EUS への更新の実行](#)
- [条件付き更新パスに沿った更新](#)
- [更新チャンネルとリリースについて](#)

3.1.6. 条件付き更新パスに沿った更新

Web コンソールまたは OpenShift CLI (**oc**) を使用して、推奨される条件付き更新パスに沿って更新できます。クラスターで条件付き更新が推奨されない場合は、OpenShift CLI (**oc**) 4.10 以降を使用して条件付き更新パスに沿って更新できます。

手順

1. リスクが適用される可能性があるため推奨されない場合に更新の説明を表示するには、次のコマンドを実行します。

```
$ oc adm upgrade --include-not-recommended
```

2. クラスター管理者が潜在的な既知のリスクを評価し、それが現在のクラスターに受け入れられると判断した場合、管理者は次のコマンドを実行して安全ガードを放棄し、更新を続行できません。

```
$ oc adm upgrade --allow-not-recommended --to <version> <.>
```

<.> <version> は、前のコマンドの出力から取得した更新バージョンです。サポートされていますが、既知の問題やリスクもあります。

関連情報

- [更新チャンネルとリリースについて](#)

3.1.7. 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
```

3.2. 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 クラスターでマイナーバージョンおよびパッチの更新を実行できます。



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ンネルを変更します。4.16 チャンネルに変更した後、[CLI を使用したクラスターの更新](#) の手順に従って更新を完了できます。

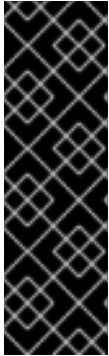
3.2.1. OpenShift Container Platform クラスターを更新する前に

更新する前に、以下を考慮してください。

- 最近 etcd をバックアップしました。
- **PodDisruptionBudget** で **minAvailable** が 1 に設定されている場合、エビクシオンプロセスをブロックする可能性がある保留中のマシン設定を適用するためにノードがドレインされます。複数のノードが再起動された場合に、すべての Pod が 1 つのノードでのみ実行される可能性があります。

あり、**PodDisruptionBudget** フィールドはノードのドレインを防ぐことができます。

- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新する必要がある可能性があります。
- 管理者確認の要求をチェックして推奨されるアクションを実行し、準備ができたなら確認する必要があります。
- 更新にかかる時間を考慮し、ワーカーノードまたはカスタムプールノードを更新して部分的な更新を実行できます。各プールのプログレスバー内で一時停止および再開できます。



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 サポートにお問い合わせください。
- **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。

3.2.2. 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。

前提条件

- **cluster-admin** 権限でクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。

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>' ❶
...
```

- ❶ **<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。

関連情報

- [更新チャンネルとリリースについて](#)

3.2.3. Web コンソールを使用した MachineHealthCheck リソースの一時停止


更新プロセスで、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

- **cluster-admin** 権限でクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。

手順

1. OpenShift Container Platform Web コンソールにログインします。
2. **Compute** → **MachineHealthChecks** に移動します。
3. マシンヘルスチェックを一時停止するには、**cluster.x-k8s.io/paused=""** アノテーションを各 **MachineHealthCheck** リソースに追加します。たとえば、アノテーションを **machine-api-termination-handler** リソースに追加するには、以下の手順を実行します。

- a. **machine-api-termination-handler** の横にあるオプションメニュー  をクリックし、**Edit annotations** をクリックします。
- b. アノテーションの**編集** ダイアログで、**更に追加** をクリックします。
- c. キー および 値 フィールドにそれぞれ **cluster.x-k8s.io/paused** と "" の値を追加し、**保存** をクリックします。

3.2.4. 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

- **cluster-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。
- OpenShift Container Platform Web コンソールにアクセスできる。
- すべての **MachineHealthCheck** リソースを一時停止している。
- Operator Lifecycle Manager (OLM) を通じて以前にインストールされたすべての Operator を、ターゲットリリースと互換性のあるバージョンに更新している。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新時に現行のマイナーバージョン

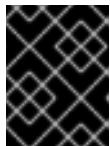
から次のマイナーバージョンに切り替わる際、確実に有効な更新パスがあるようにします。

「関連情報」セクションの「インストール済み Operator の更新」を参照し、互換性を確認する方法の詳細を確認して、インストールされている Operator を必要に応じて更新してください。

- マシン設定プール (MCP) は実行中であり、一時停止されていない。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 を一時停止できる。
- RHEL7 ワーカーは、RHEL8 または RHCOS ワーカーに置き換えられます。Red Hat は、RHEL7 から RHEL8 への RHEL ワーカーのインプレース更新をサポートしていません。このようなホストは、オペレーティングシステムをクリーンインストールして置き換える必要があります。

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は、**チャンネル** が、**stable-4.16** など、更新先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てください。



重要

実稼働クラスターの場合は、**stable-***、**eus-*** 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。



注記

次のマイナーバージョンに移行する準備ができたなら、そのマイナーバージョンに対応するチャンネルを選択します。更新チャンネルの宣言が早ければ早いほど、クラスターはターゲットバージョンへの更新パスをより効果的に推奨できます。クラスターは、利用可能なすべての可能な更新プログラムを評価し、最適な更新プログラムの推奨事項を選択するために、しばらく時間がかかる場合があります。更新の推奨事項は、その時点で利用可能な更新オプションに基づいているため、時間の経過とともに変化する可能性があります。

ターゲットマイナーバージョンへの更新パスが表示されない場合は、次のマイナーバージョンがパスで利用可能になるまで、現在のバージョンの最新のパッチリリースにクラスターを更新し続けます。

- **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、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

クラスターを次のマイナーバージョン (バージョン 4.10 から 4.11 など) に更新する場合は、新しい機能を利用するワークロードをデプロイする前に、ノードが更新されていることを確認してください。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、**チャンネル** を次のマイナーバージョンの **stable-***、**eus-*** または **fast-*** チャンネルに変更し、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。

関連情報

- [インストール済み Operator の更新](#)

3.2.5. Web コンソールで条件付き更新を表示する

条件付き更新を使用して、特定の更新に関連するリスクを表示および評価できます。

前提条件

- **cluster-admin** 権限でクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- すべての **MachineHealthCheck** リソースを一時停止している。
- Operator Lifecycle Manager (OLM) を通じて以前にインストールされたすべての Operator を、ターゲットリリースと互換性のあるバージョンに更新している。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新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。「関連情報」セクションの「インストール済み Operator の更新」を参照し、互換性を確認する方法の詳細を確認して、インストールされている Operator を必要に応じて更新してください。
- マシン設定プール (MCP) は実行中であり、一時停止されていない。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、EUS 更新、コントロールプレーン更新などの高度な更新ストラテジーを実行している場合は、MCP を一時停止できます。

手順

1. Web コンソールから、**Administration** → **Cluster settings** ページをクリックし、**Details** タブの内容を確認します。
2. **Update cluster** モーダルの **Select new version** ドロップダウンで **Include versions with known issues** 機能を有効にして、ドロップダウンリストに条件付き更新を入力できます。



注記

既知の問題のあるバージョンを選択した場合は、そのバージョンに関連する潜在的なリスクに関する詳細情報が提供されます。

3. 更新の潜在的なリスクについて詳しく説明した通知を確認してください。

関連情報

- [インストール済み Operator の更新](#)
- [更新の推奨と条件付き更新](#)

3.2.6. カナリアロールアウト更新の実行

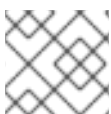
特定のユースケースでは、特定ノードを残りのクラスターと同時に更新しない、制御された更新プロセスが必要になる場合があります。これらのユースケースには、以下のようなものがありますが、これに限定されません。

- 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。

ローリング更新のプロセスは、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。組織がローリング更新を使用し、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更新プロセスでは、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 の一時停止は、慎重に考慮した上で短時間に限定して実行する必要があります。

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。

3.2.7. 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、更新に失敗した場合は、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。
 - 更新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 が短時間ですぐに解決します。



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

- [Machine Config Operator について](#)

3.3. EUS から EUS への更新の実行

基本的な Kubernetes の設計により、マイナーバージョン間のすべての OpenShift Container Platform の更新をシリアライズする必要があります。OpenShift Container Platform <4.y> から <4.y+1> に更新してから、<4.y+2> に更新する必要があります。OpenShift Container Platform <4.y> から <4.y+2> に直接更新することはできません。ただし、2 つの Extended Update Support (EUS) バージョン間で更新したい管理者は、非コントロールプレーンホストを 1 回再起動するだけで更新できます。



重要

EUS から EUS への更新は、OpenShift Container Platform の **偶数番号のマイナーバージョン** 間でのみ実行可能です。

EUS から EUS への更新を試みる際に考慮すべきいくつかの注意事項があります。

- EUS から EUS への更新は、関連するすべてのバージョン間の更新が **stable** チャンネルで利用可能になった後にのみ提供されます。
- 奇数のマイナーバージョンへの更新中またはアップグレード後 (ただし、次の偶数のバージョンに更新する前) に問題が発生した場合、これらの問題を修正するには、コントロールプレーン以外のホストが先に進む前に奇数のバージョンへの更新を完了する必要があります。
- ワーカーまたはカスタムプールノードを更新して、メンテナンスにかかる時間に対応することにより、部分的な更新を行うことができます。
- マシン設定プールの一時停止が解除され、更新が完了するまで、OpenShift Container Platform の <4.y+1> および <4.y+2> の一部の機能およびバグ修正は利用できません。
- すべてのクラスターは、プールを一時停止せずに従来の更新に EUS チャンネルを使用して更新できますが、プールを一時停止して EUS から EUS への更新を実行できるのは、コントロールプレーン以外の **MachineConfigPools** オブジェクトを持つクラスターのみです。

3.3.1. EUS から EUS への更新

以下の手順では、マスター以外のすべてのマシン設定プールを一時停止し、OpenShift Container Platform <4.y> から <4.y+1>、さらに <4.y+2> への更新を実行してから、以前に一時停止したマシン設定プールの一時停止を解除します。この手順に従うと、合計更新期間とワーカーノードが再起動される回数が減ります。

前提条件

- OpenShift Container Platform <4.y+1> および <4.y+2> のリリースノートを確認してください
- 階層化された製品および Operator Lifecycle Manager (OLM) Operator のリリースノートおよび製品ライフサイクルを確認する。EUS から EUS への更新前または更新中に更新が必要になる場合があります。
- OpenShift Container Platform <4.y+1> から <4.y+2> に更新する前に必要な、非推奨の API の削除など、バージョン固有の前提条件をよく理解していることを確認してください。

3.3.1.1. Web コンソールを使用した EUS から EUS への更新

前提条件

- マシン設定プールの一時停止が解除されている。
- **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。

手順

1. Web コンソールの管理者パースペクティブを使用して、任意の Operator Lifecycle Manager (OLM) Operator を、目的の更新バージョンと互換性のあるバージョンに更新します。このアクションを実行する方法は、インストール済み Operator の更新を参照してください。
2. すべてのマシン設定プールが **Up to date** のステータスを表示し、マシン設定プールが **UPDATING** のステータスを表示していないことを確認します。
すべてのマシン設定プールのステータスを表示するには、**Compute** → **MachineConfigPools** をクリックし、**Update status** 列の内容を確認します。



注記

マシン設定プールのステータスが **Updating** の場合は、このステータスが **Up to date** に変わるまでお待ちください。このプロセスには数分かかる場合があります。

3. チャンネルを **eus-<4.y+2>** に設定します。
チャンネルを設定するには、**Administration** → **Cluster Settings** → **Channel** をクリックします。現在のハイパーリンクチャンネルをクリックすると、チャンネルを編集できます。
4. マスタープール以外のすべてのワーカーマシンプールを一時停止します。このアクションは、**Compute** ページの **MachineConfigPools** タブで実行できます。一時停止するマシン設定プールの横にある縦リーダーを選択し、**Pause updates** をクリックします。
5. バージョン **<4.y+1>** に更新し、**Save** ステップまで完了します。これらのアクションを実行する方法は、関連情報の「Web コンソールを使用したクラスターの更新」を参照してください。
6. クラスターの **最後に完了したバージョン** を表示して、**<4.y+1>** の更新が完了していることを確認します。この情報は、**Cluster Settings** ページの **Details** タブにあります。
7. 必要に応じて、Web コンソールの管理者パースペクティブを使用して OLM オペレーターをアップグレードします。これらのアクションを実行する方法は、インストール済み Operator の更新を参照してください。
8. バージョン **<4.y+2>** に更新し、**Save** ステップまで完了します。これらのアクションを実行する方法は、関連情報の「Web コンソールを使用したクラスターの更新」を参照してください。
9. クラスターの **最後に完了したバージョン** を表示して、**<4.y+2>** の更新が完了していることを確認します。この情報は、**Cluster Settings** ページの **Details** タブにあります。
10. 以前一時停止したすべてのマシン設定プールの一時停止を解除します。このアクションは、**Compute** ページの **MachineConfigPools** タブで実行できます。一時停止を解除するマシン設定プールの横にある縦リーダーを選択し、**Unpause updates** をクリックします。

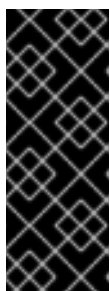


重要

プールが一時停止になった場合、クラスターは将来のマイナーバージョンへのアップグレードが許可されず、一部のメンテナンスタスクが禁止されます。これにより、クラスターは将来の劣化のリスクにさらされます。

11. 以前に一時停止したプールが更新され、クラスターがバージョン **<4.y+2>** への更新を完了したことを確認します。

Compute ページの **MachineConfigPools** タブで、**Update status** の値が **Up to date** になっていることを確認して、プールが更新されたことを確認できます。



重要

Red Hat Enterprise Linux (RHEL) コンピュータマシンを含むクラスターを更新すると、更新プロセス中はそれらのマシンが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 マシンのステータスが **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。詳細は、関連情報セクションの「RHEL コンピュータマシンを含むクラスターの更新」を参照してください。

クラスターの **Last completed version** を表示することで、クラスターが更新を完了したことを確認できます。この情報は、**Cluster Settings** ページの **Details** タブにあります。

関連情報

- [インストール済み Operator の更新](#)
- [Web コンソールを使用したクラスターの更新](#)
- [RHEL コンピュータマシンを含むクラスターの更新](#)

3.3.1.2. CLI を使用した EUS から EUS への更新

前提条件

- マシン設定プールの一時停止が解除されている。
- 各更新の前に OpenShift CLI (**oc**) をターゲットバージョンに更新する。



重要

この前提条件をスキップすることは推奨されていません。更新前に OpenShift CLI (**oc**) がターゲットバージョンに更新されていない場合、予期しない問題が発生する可能性があります。

手順

1. Web コンソールの管理者パースペクティブを使用して、任意の Operator Lifecycle Manager (OLM) Operator を、目的の更新バージョンと互換性のあるバージョンに更新します。このアクションを実行する方法は、インストール済み Operator の更新を参照してください。
2. すべてのマシン設定プールが **UPDATED** のステータスを表示し、マシン設定プールが **UPDATING** のステータスを表示していないことを確認します。すべてのマシン設定プールのステータスを表示するには、以下のコマンドを実行します。

```
$ oc get mcp
```

出力例

```
NAME          CONFIG                                UPDATED  UPDATING
master        rendered-master-ecbb9582781c1091e1c9f19d50cf836c  True    False
worker        rendered-worker-00a3f0c68ae94e747193156b491553d5   True    False
```

3. 現在のバージョンは $\langle 4.y \rangle$ で、更新する予定のバージョンは $\langle 4.y+2 \rangle$ です。次のコマンドを実行して、**eus- $\langle 4.y+2 \rangle$** チャンネルに変更します。

```
$ oc adm upgrade channel eus- $\langle 4.y+2 \rangle$ 
```



注記

eus-<4.y+2> が利用可能なチャンネルの1つでないことを示すエラーメッセージが表示された場合、これは、Red Hat が EUS バージョンの更新をまだロールアウトしていることを示しています。通常、このロールアウトプロセスには GA 日から 45 ~ 90 日かかります。

- 以下のコマンドを実行して、マスタープール以外のすべてのワーカーマシンプールを一時停止します。

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



注記

マスタープールを一時停止することはできません。

- 次のコマンドを実行して、最新バージョンに更新します。

```
$ oc adm upgrade --to-latest
```

出力例

```
Updating to latest version <4.y+1.z>
```

- クラスターのバージョンを確認し、以下のコマンドを実行して更新が完了したことを確認します。

```
$ oc adm upgrade
```

出力例

```
Cluster version is <4.y+1.z>
...
```

- 次のコマンドを実行して、バージョン <4.y+2> に更新します。

```
$ oc adm upgrade --to-latest
```

- 次のコマンドを実行して、クラスターのバージョンを取得し、<4.y+2> の更新が完了していることを確認します。

```
$ oc adm upgrade
```

出力例

```
Cluster version is <4.y+2.z>
...
```

- ワーカーノードを <4.y+2> に更新するには、次のコマンドを実行して、以前に一時停止したすべてのマシン設定プールの一時停止を解除します。

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

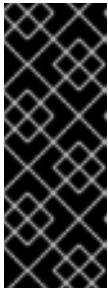


重要

プールの一時停止が解除されていない場合、クラスターは将来のマイナーバージョンへの更新が許可されず、一部のメンテナンスタスクが禁止されます。これにより、クラスターは将来の劣化のリスクにさらされます。

- 次のコマンドを実行して、以前に一時停止したプールが更新され、バージョン <4.y+2> への更新が完了したことを確認します。

```
$ oc get mcp
```



重要

Red Hat Enterprise Linux (RHEL) コンピュータマシンを含むクラスターを更新すると、更新プロセス中はそれらのマシンが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 マシンの状態が **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。詳細は、関連情報セクションの「RHEL コンピュータマシンを含むクラスターの更新」を参照してください。

出力例

| NAME | CONFIG | UPDATED | UPDATING |
|--------|--|---------|----------|
| master | rendered-master-52da4d2760807cb2b96a3402179a9a4c | True | False |
| worker | rendered-worker-4756f60eccae96fb9dcb4c392c69d497 | True | False |

関連情報

- [インストール済み Operator の更新](#)
- [RHEL コンピュータマシンを含むクラスターの更新](#)

3.3.1.3. Operator Lifecycle Manager でインストールされたレイヤード製品および Operator の EUS から EUS への更新

以下におけるクラスターの EUS から EUS への更新を実行する場合、Web コンソールおよび CLI に記載されている EUS から EUS への更新手順に加え、考慮すべき追加の手順があります。

- レイヤード製品
- Operator Lifecycle Manager (OLM) でインストールされた Operator

レイヤード製品とは

レイヤード製品は、併用することが意図され、個別のサブスクリプションに分割できない複数の基礎となる製品で構成される製品を指します。OpenShift Container Platform レイヤード製品の例は、[OpenShift のレイヤード製品](#) を参照してください。

レイヤード製品のクラスターや OLM でインストールされた Operator の EUS から EUS への更新を実行する場合、以下を完了する必要があります。

1. Operator Lifecycle Manager (OLM) を通じて以前にインストールされたすべての Operator を、ターゲットリリースと互換性のあるバージョンに更新している。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新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。「関連情報」セクションの「インストール済み Operator の更新」を参照し、互換性を確認する方法の詳細を確認して、インストールされている Operator を必要に応じて更新してください。
2. 現在の Operator バージョンと更新後の Operator バージョン間のクラスターバージョン互換性を確認します。[Red Hat OpenShift Container Platform Operator Update Information Checker](#) を使用して、OLM Operator と互換性があるバージョンを確認できます。

たとえば以下は、OpenShift Data Foundation (ODF) の <4.y> から <4.y+2> に、EUS から EUS への更新を実行する手順です。これは、CLI または Web コンソールから実行できます。目的のインターフェイスでクラスターを更新する方法については、関連情報の [Web コンソールを使用した EUS から EUS への更新](#) および「CLI を使用した EUS から EUS への更新」を参照してください。

ワークフローの例

1. ワーカーマシンプールを一時停止します。
2. OpenShift <4.y> → OpenShift <4.y+1> を更新します。
3. ODF <4.y> → ODF <4.y+1> を更新します。
4. OpenShift <4.y+1> → OpenShift <4.y+2> を更新します。
5. ODF <4.y+2> に更新します。
6. ワーカーマシンプールの一時停止を解除します。



注記

ODF <4.y+2> への更新は、ワーカーマシンプールの一時停止が解除される前または後に実行できます。

関連情報

- [インストール済み Operator の更新](#)
- [Web コンソールを使用した EUS から EUS への更新](#)
- [CLI を使用した EUS から EUS への更新](#)

3.4. カナリアロールアウト更新の実行

カナリア更新 は、すべてのワーカーノードを同時に更新するのではなく、ワーカーノードの更新を個別の段階に分けて順次実行する更新ストラテジーです。このストラテジーは、次の場合に役立ちます。

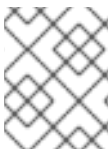
- ワーカーノード更新のロールアウトをより細かく制御して、更新プロセスによってアプリケーションが失敗した場合でも、更新全体を通じてミッションクリティカルなアプリケーションが利用可能な状態を維持する必要がある。
- 少数のワーカーノードを更新し、一定期間にわたりクラスターとワークロードの健全性を評価してから、残りのノードを更新する必要がある。

- クラスター全体を一度に更新するために長いメンテナンス期間を取ることができない場合に、通常はホストの再起動が必要となるワーカーノードの更新を、より短い一定のメンテナンス期間内に収める必要がある。

これらのシナリオでは、複数のカスタムマシン設定プール (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。

3.4.1. カナリア更新ストラテジーの例

次の例では、10%の余剰容量を備えた100ノードのクラスターのカナリア更新ストラテジーについて説明します。メンテナンス期間は4時間未満にする必要があり、ワーカーのドレインと再起動は8分未満で完了することがわかっています。



注記

前述の値は単なる例です。ノードのドレインにかかる時間は、ワークロードなどの要因によって異なる場合があります。

カスタムマシン設定プールの定義

ワーカーノードの更新を個別の段階に分けて編成するために、まず次の MCP を定義します。

- 10個のノードを含む `workerpool-canary`
- 30個のノードを含む `workerpool-A`
- 30個のノードを含む `workerpool-B`
- 30個のノードを含む `workerpool-C`

カナリアワーカープールの更新

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、OpenShift Container Platform 上で実行されるコンポーネントと、一時停止されていない `workerpool-canary` MCP に含まれる10個の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。

残りのワーカープールの更新に進むかどうかの決定

何らかの理由で、クラスターまたはワークロードの健全性が `workerpool-canary` の更新によって悪影響を受けたと判断した場合は、問題を診断して解決するまで十分な容量を維持しながら、そのプール内のすべてのノードを遮断してドレインします。すべてが期待どおりに動作している場合は、クラスターとワークロードの健全性を評価してから、一時停止を解除し、別の各メンテナンス期間中に、`workerpool-A`、`workerpool-B`、および `workerpool-C` を順次更新します。

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と、柔軟性が得られます。一方で、複数のコマンドを実行する必要があるため、時間のかかるプロセスになる可能性があります。この複雑さにより、クラスター全体に影響を及ぼす可能性のあるエラーが発生する可能性があります。開始する前に、組織のニーズを慎重に検討し、プロセスの実装を慎重に計画することを推奨します。



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。

kube-apiserver-to-kubelet-signer CA 証明書の有効期限が切れたときに MCP が一時停止され、MCO が証明書を自動的に更新しようとする、MCO は新しくローテーションされた証明書をそれらのノードにプッシュできません。これにより、**oc debug**、**oc logs**、**oc exec**、**oc attach** などの複数の **oc** コマンドでエラーが発生します。証明書がローテーションされたときに MCP が一時停止された場合、OpenShift Container Platform コンソールのアラート UI でアラートを受け取ります。

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。



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。

3.4.2. 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、ノードは個別に考慮されません。代わりに、ノードはマシン設定プール (MCP) にグループ化されます。デフォルトでは、OpenShift Container Platform クラスター内のノードは 2 つの MCP にグループ化されます。1 つはコントロールプレーンノード用、もう 1 つはワーカーノード用です。OpenShift Container Platform の更新は、すべての MCP に同時に影響します。

更新中、Machine Config Operator (MCO) は、最大数が指定されている場合、指定された **maxUnavailable** ノード数まで MCP 内のすべてのノードをドレインし、遮断します。デフォルトでは、**maxUnavailable** は 1 に設定されます。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。

カスタムマシン設定プールの使用

特定のノードが更新されないようにするには、カスタム MCP を作成します。一時停止された MCP 内のノードは、MCO によって更新されません。そのため、クラスターの更新を開始する前に、更新する必要がないノードを含む MCP を一時停止できます。

1 つ以上のカスタム MCP を使用すると、ワーカーノードを更新する順序をより詳細に制御できます。たとえば、最初の MCP のノードを更新した後、アプリケーションの互換性を確認してから、残りのノードを新しいバージョンに段階的に更新できます。

**警告**

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

**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。

カスタムマシン設定プールを使用する場合の考慮事項

ワークロードのデプロイメントポロジータに基づいて、作成する MCP の数と各 MCP 内のノードの数を慎重に検討してください。たとえば、更新を特定のメンテナンス期間に合わせる必要がある場合は、OpenShift Container Platform が一定期間内に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、それぞれのクラスターとワークロードの特性によって異なります。

カスタム MCP の数と各 MCP 内のノードの数を決定するには、クラスター内で利用可能な余剰容量についても考慮する必要があります。新しく更新されたノードでアプリケーションが期待どおりに動作しない場合は、プール内のそのノードを遮断してドレインし、アプリケーション Pod を他のノードに移動できます。ただし、他の MCP 内の使用可能なノードがアプリケーションに十分なサービス品質 (QoS) を提供できるかどうかを確認する必要があります。

**注記**

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。

3.4.3. カナリアロールアウト更新の実行について

次の手順は、カナリアロールアウト更新プロセスのワークフローの概要を示しています。

1. ワーカープールに基づいてカスタムマシン設定プール (MCP) を作成します。

**注記**

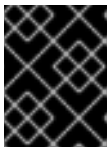
MCP の **maxUnavailable** 設定を変更して、任意の時点で更新できるパーセンテージまたはマシン数を指定できます。デフォルトは **1** です。



警告

OpenShift Container Platform のすべてのマシン設定プールにおける **maxUnavailable** のデフォルト設定は **1** です。この値を変更せず、一度に1つのコントロールプレーンノードを更新することを推奨します。コントロールプレーンプールのこの値を **3** に変更しないでください。

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。



重要

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。クラスター内でノードを適切に機能させるには、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。
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、一時停止されていない MCP が更新されます。
5. 更新されたノードでアプリケーションをテストし、期待どおりに動作することを確認します。
6. 残りの MCP の1つを一時停止解除し、そのプール内のノードの更新が完了するのを待ち、それらのノードでアプリケーションをテストします。すべてのワーカーノードが更新されるまで、このプロセスを繰り返します。
7. オプション: 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。

3.4.4. 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

カナリアロールアウト更新を実行するには、まず1つ以上のカスタムマシン設定プール (MCP) を作成する必要があります。

手順

1. 次のコマンドを実行して、クラスター内のワーカーノードをリスト表示します。

```
$ 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-lldb
```

2. 更新を遅らせるノードごとに、次のコマンドを実行して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>=
```

以下に例を示します。

```
$ 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
```

3. 新規 MCP を作成します。

a. MCP の YAML ファイルを作成します。

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

- ❶ MCP の名前を指定します。
- ❷ **worker** およびカスタム MCP 名を指定します。
- ❸ このプールに必要なノードに追加したカスタムラベルを指定します。

b. 次のコマンドを実行して、**MachineConfigPool** オブジェクトを作成します。

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

出力例

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

4. 次のコマンドを実行して、クラスター内の MCP のリストとそ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。

3.4.5. マシン設定プールの一時停止

カスタムマシン設定プール (MCP) を作成したら、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。

手順

1. 次のコマンドを実行して、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
```

以下に例を示します。

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

出力例

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

3.4.6. クラスターの更新の実行

マシン設定プール (MCP) が準備完了状態になったら、クラスターの更新を実行できます。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。

- [Web コンソールを使用してクラスターを更新](#)
- [CLI を使用したクラスターの更新](#)

クラスターの更新が完了したら、MCP の一時停止を1つずつ解除し始めることができます。

3.4.7. マシン設定プールの一時停止の解除

OpenShift Container Platform の更新が完了したら、カスタムマシン設定プール (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。

手順

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
```

以下に例を示します。

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

出力例

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

2. オプション: 次のいずれかの方法を使用して、更新の進行状況を確認します。
 - a. Web コンソールから **Administration** → **Cluster settings** をクリックして進行状況を確認します。
 - b. 次のコマンドを実行して進行状況を確認します。

```
$ oc get machineconfigpools
```

3. 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。
4. 他の一時停止中の MCP についても、1つずつこのプロセスを繰り返します。

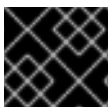


注記

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。

3.4.8. ノードを元のマシン設定プールに移動する

カスタムマシン設定プール (MCP) 内のノード上のアプリケーションを更新して検証したら、ノードに追加したカスタムラベルを削除して、ノードを元の MCP に戻します。



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。

手順

1. カスタム MCP 内のノードごとに、次のコマンドを実行して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>-
```

以下に例を示します。

```
$ 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
```

Machine Config Operator がノードを元の MCP に戻し、ノードを MCP 設定に合わせて調整します。

2. ノードがカスタム MCP から削除されたことを確認するには、次のコマンドを実行して、クラスター内の MCP のリストとその現在の状態を表示します。

```
$ oc get mcp
```

出力例

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

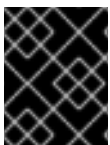
ノードをカスタム MCP から削除し、元の MCP に戻ると、マシン数の更新に数分かかることがあります。この例では、削除した **workerpool-canary** MCP から1つのノードを **worker** MCP に移動しました。

3. オプション: 次のコマンドを実行してカスタム MCP を削除します。

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

3.5. RHEL コンピュータマシンを含むクラスターの更新

OpenShift Container Platform クラスターでマイナーバージョンおよびパッチの更新を実行できます。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。



重要

OpenShift Container Platform クラスターでの RHEL コンピュータマシンの使用は非推奨となり、今後のリリースでは削除される予定です。

3.5.1. 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。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、クラスターを以前の状態に復元する必要がある場合に備えて、最新の [etcd バックアップ](#) を用意している。
- RHEL7 ワーカーは、RHEL8 または RHCOS ワーカーに置き換えられます。Red Hat は、RHEL7 から RHEL8 への RHEL ワーカーのインプレース更新をサポートしていません。このようなホストは、オペレーティングシステムをクリーンインストールして置き換える必要があります。
- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは [手動で維持された認証情報でクラスターを更新する準備](#) を参照してください。
- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合は、更新プロセス中に中断が発生する可能性があります。**PodDisruptionBudget** で **minAvailable** が1に設定されている場合、**削除** プロセスをブロックする可能性がある保留中のマシン設定を適用するためにノードがドレインされます。複数のノードが再起動された場合に、すべての Pod が1つのノードでのみ実行される可能性があり、**PodDisruptionBudget** フィールドはノードのドレインを防ぐことができます。

関連情報

- [管理外の Operator のサポートポリシー](#)

3.5.2. 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

- **cluster-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。
- OpenShift Container Platform Web コンソールにアクセスできる。
- すべての **MachineHealthCheck** リソースを一時停止している。
- Operator Lifecycle Manager (OLM) を通じて以前にインストールされたすべての Operator を、ターゲットリリースと互換性のあるバージョンに更新している。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新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。「[関連情報](#)」セクションの「[インストール済み Operator の更新](#)」を参照し、互換性を確認する方法の詳細を確認して、インストールされている Operator を必要に応じて更新してください。
- マシン設定プール (MCP) は実行中であり、一時停止されていない。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 を一時停止できる。
- RHEL7 ワーカーは、RHEL8 または RHCOS ワーカーに置き換えられます。Red Hat は、RHEL7 から RHEL8 への RHEL ワーカーのインプレース更新をサポートしていません。このようなホストは、オペレーティングシステムをクリーンインストールして置き換える必要があります。

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は、**チャンネル** が、**stable-4.16** など、更新先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てください。



重要

実稼働クラスターの場合は、**stable-***、**eus-*** 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。



注記

次のマイナーバージョンに移行する準備ができたなら、そのマイナーバージョンに対応するチャンネルを選択します。更新チャンネルの宣言が早ければ早いほど、クラスターはターゲットバージョンへの更新パスをより効果的に推奨できます。クラスターは、利用可能なすべての可能な更新プログラムを評価し、最適な更新プログラムの推奨事項を選択するために、しばらく時間がかかる場合があります。更新の推奨事項は、その時点で利用可能な更新オプションに基づいているため、時間の経過とともに変化する可能性があります。

ターゲットマイナーバージョンへの更新パスが表示されない場合は、次のマイナーバージョンがパスで利用可能になるまで、現在のバージョンの最新のパッチリリースにクラスターを更新し続けます。

- **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、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

クラスターを次のマイナーバージョン (バージョン 4.10 から 4.11 など) に更新する場合は、新しい機能を利用するワークロードをデプロイする前に、ノードが更新されていることを確認してください。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、**チャンネル** を次のマイナーバージョンの **stable-***、**eus-*** または **fast-*** チャンネルに変更し、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。



重要

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。

関連情報

- [インストール済み Operator の更新](#)

3.5.3. 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。

3.5.3.1. 更新用の Ansible フックについて

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、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、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更新を再開する必要があります。

3.5.3.2. 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設定される必要があります、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。

3.5.3.3. 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 |
|--|--|
| 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。 |

| フック名 | 説明 |
|---|---|
| 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。 |

3.5.4. 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

RHEL コンピューティングマシンでは、Red Hat Enterprise Linux (RHEL) バージョン 8.6 以降がサポートされています。

RHEL をオペレーティングシステムとして使用する場合は、コンピュータマシンを別の 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。

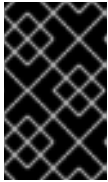


重要

RHEL 7 コンピューティングマシンを RHEL 8 に更新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 コンピュータ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` サービスが有効になります。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 4.16 で必要なリポジトリを有効にします。
 - a. Ansible Playbook を実行するマシンで、必要なリポジトリを更新します。

```
# subscription-manager repos --disable=rhocp-4.15-for-rhel-8-x86_64-rpms \
--enable=rhocp-4.16-for-rhel-8-x86_64-rpms
```



重要

OpenShift Container Platform 4.11 の時点で、Ansible Playbook は RHEL 8 に対してのみ提供されています。RHEL 7 システムが OpenShift Container Platform 4.10 Ansible Playbook のホストとして使用された場合、Ansible ホストを RHEL 8 に更新するか、RHEL 8 システムに新しい Ansible ホストを作成し、古い Ansible ホストからインベントリーをコピーする必要があります。

- b. Ansible Playbook を実行するマシンで、Ansible パッケージを更新します。

```
# yum swap ansible ansible-core
```

- c. 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
```

- d. 各 RHEL コンピュータノードで、必要なリポジトリを更新します。

```
# subscription-manager repos --disable=rhocp-4.15-for-rhel-8-x86_64-rpms \
--enable=rhocp-4.16-for-rhel-8-x86_64-rpms
```

3. RHEL ワーカーマシンを更新します。

- a. 次の例に示すように、/**<path>/inventory/hosts** にある Ansible インベントリファイルを確認し、その内容を更新して、RHEL 8 マシンが **[workers]** セクションにリストされるようにします。

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

openshift_kubeconfig_path=~/.kube/config"

[workers]
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com
mycluster-rhel8-2.example.com
mycluster-rhel8-3.example.com
```

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

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

- c. **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 パッケージのみを更新します。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. 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   v1.29.4
mycluster-control-plane-1 Ready    master   145m   v1.29.4
mycluster-control-plane-2 Ready    master   145m   v1.29.4
mycluster-rhel8-0    Ready    worker   98m    v1.29.4
```

| | | | | |
|-------------------|-------|--------|-----|---------|
| mycluster-rhel8-1 | Ready | worker | 98m | v1.29.4 |
| mycluster-rhel8-2 | Ready | worker | 98m | v1.29.4 |
| mycluster-rhel8-3 | Ready | worker | 98m | v1.29.4 |

- 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。4.16 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

4.16 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。

3.6. 非接続環境でのクラスターの更新

3.6.1. 非接続環境でのクラスターの更新について

非接続環境とは、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。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。

切断されたネットワーク内の複数のクラスターのミラーリングされたイメージをホストするには、1つのコンテナイメージレジストリーで十分です。

3.6.1.1. OpenShift Container Platform イメージのミラーリング

非接続環境でクラスターを更新するには、クラスター環境がターゲット更新に必要なイメージおよびリソースを持つミラーレジストリーにアクセスできる必要があります。以下のページでは、非接続クラスターのリポジトリにイメージをミラーリングする手順を説明します。

- [OpenShift Container Platform イメージのミラーリング](#)

3.6.1.2. 非接続環境でのクラスターの更新の実行

以下のいずれかの手順を使用して、切断された OpenShift Container Platform クラスターを更新できます。

- [OpenShift Update Service を使用した非接続環境でのクラスターの更新](#)
- [OpenShift Update Service を使用しない非接続環境でのクラスターの更新](#)

3.6.1.3. クラスターからの OpenShift Update Service のアンインストール

以下の手順を使用して、OpenShift Update Service (OSUS) のローカルコピーをクラスターからアンインストールできます。

- [クラスターからの OpenShift Update Service のアンインストール](#)

3.6.2. OpenShift Container Platform イメージのミラーリング

非接続環境でクラスターを更新する前に、コンテナイメージをミラーレジストリーにミラーリングする必要があります。接続された環境でこの手順を使用して、外部コンテンツに関する組織の制限を満たしている承認済みコンテナイメージのみをクラスターで実行するようにすることもできます。



注記

ミラーレジストリーは、クラスターの実行中に常に実行されている必要があります。

以下に示す手順は、ミラーレジストリーにイメージをミラーリングする大まかなワークフローです。

1. リリースイメージの取得およびプッシュに使用されるすべてのデバイスに OpenShift CLI (**oc**) をインストールします。
2. レジストリープルシークレットをダウンロードし、クラスターに追加します。
3. **oc-mirror OpenShift CLI (oc) プラグイン** を使用する場合:
 - a. リリースイメージの取得およびプッシュに使用されるすべてのデバイスに **oc-mirror プラグイン** をインストールします。
 - b. ミラーリングするリリースイメージを決定する際に、使用するプラグイン用のイメージセット設定ファイルを作成します。この設定ファイルは後で編集して、プラグインがミラーリングするリリースイメージを変更できます。
 - c. ターゲットのリリースイメージをミラーレジストリーに直接ミラーリングするかリムーバブルメディアにミラーリングしてからミラーレジストリーにミラーリングします。
 - d. **oc-mirror プラグイン** が生成したリソースを使用するようにクラスターを設定します。
 - e. 必要に応じてこれらの手順を繰り返し、ミラーレジストリーを更新します。
4. **oc adm release mirror コマンド** を使用する場合:
 - a. 使用している環境とミラーリングするリリースイメージに対応する環境変数を設定します。
 - b. ターゲットのリリースイメージをミラーレジストリーに直接ミラーリングするかリムーバブルメディアにミラーリングしてからミラーレジストリーにミラーリングします。
 - c. 必要に応じてこれらの手順を繰り返し、ミラーレジストリーを更新します。

oc adm release mirror コマンドを使用する場合と比較して、**oc-mirror プラグイン** には次の利点があります。

- コンテナイメージ以外のコンテンツをミラーリングできます。
- 初めてイメージをミラーリングした後は、レジストリー内のイメージを簡単に更新できます。
- **oc-mirror プラグイン** は、Quay からリリースペイロードをミラーリングする自動化された方法を提供し、非接続環境で実行されている OpenShift Update Service 用の最新のグラフデータイメージを構築します。

3.6.2.1. oc-mirror プラグインを使用したリソースのミラーリング

oc-mirror OpenShift CLI (**oc**) プラグインを使用して、完全なまたは部分的な非接続環境でイメージをミラーレジストリーにミラーリングできます。公式の Red Hat レジストリーから必要なイメージをダウンロードするには、インターネット接続のあるシステムから oc-mirror を実行する必要があります。

詳細は、[oc-mirror プラグインを使用した切断されたインストールのイメージのミラーリング](#) を参照してください。

3.6.2.2. oc adm release mirror コマンドを使用したイメージのミラーリング

oc adm release mirror コマンドを使用して、イメージをミラーレジストリーにミラーリングできます。

3.6.2.2.1. 前提条件

- Red Hat Quay など、OpenShift Container Platform クラスターをホストする場所に [Docker v2-2](#) をサポートするコンテナイメージレジストリーを持っている。



注記

Red Hat Quay を使用する場合は、oc-mirror プラグインでバージョン 3.6 以降を使用する必要があります。Red Hat Quay のライセンスをお持ちの場合は、[概念実証のため](#)に、または [Quay Operator](#) を使用して Red Hat Quay をデプロイする方法を記載したドキュメントを参照してください。レジストリーの選択とインストールについてさらにサポートが必要な場合は、営業担当者または Red Hat サポートにお問い合わせください。

コンテナイメージレジストリーの既存のソリューションがない場合は、OpenShift Container Platform サブスクリプションに含まれる [Red Hat OpenShift のミラーレジストリー](#) を使用できます。**Red Hat OpenShift のミラーレジストリー** は、非接続インストールおよび更新で OpenShift Container Platform コンテナイメージをミラーリングするために使用できる小規模なコンテナレジストリーです。

3.6.2.2.2. ミラーホストの準備

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

3.6.2.2.2.1. 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 4.16 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。非接続環境でのクラスターを更新する場合は、更新する予定の **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での [OpenShift Container Platform ダウンロードページ](#) に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.16 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

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

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

```
$ echo $PATH
```

検証

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルでの [OpenShift Container Platform ダウンロードページ](#) に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 Windows Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. ZIP プログラムでアーカイブを展開します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。**PATH** を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

検証

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

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

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. バージョン ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.16 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.16 macOS arm64 Client** エントリーを選択します。

4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATHを確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

検証

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

```
$ oc <command>
```

関連情報

- [CLI プラグインのインストールおよび使用](#)

3.6.2.2.2.2. 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 からミラーにイメージをミラーリングできるコンテナイメージレジストリー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

- 非接続環境で使用するミラーレジストリーを設定しました。
- 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. 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ます。
2. 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 | jq . > <path>/<pull_secret_file_in_json> ❶
```

- ❶ 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"
    }
  }
}
```

```

    }
  }
}

```

3. オプション : oc-mirror プラグインを使用している場合は、ファイルを `~/.docker/config.json` または `$XDG_RUNTIME_DIR/containers/auth.json` として保存します。
 - a. `.docker` または `$XDG_RUNTIME_DIR/containers` ディレクトリが存在しない場合は、以下のコマンドを入力して作成します。

```
$ mkdir -p <directory_name>
```

ここで、`<directory_name>` は、`~/ .docker` または `$XDG_RUNTIME_DIR/containers` のいずれかになります。

- b. 次のコマンドを入力して、プルシークレットを適切なディレクトリにコピーします。

```
$ cp <path>/<pull_secret_file_in_json> <directory_name>/<auth_file>
```

ここで、`<directory_name>` は `~/.docker` または `$XDG_RUNTIME_DIR/containers` で、`<auth_file>` は `config.json` または `auth.json` のいずれかになります。

4. 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 ❶
BGVtbYk3ZHAAtqXs=
```

- ❶ `<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。

5. JSON 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 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レジストリーのドメイン名およびポート (オプション) を指定します。例: `registry.example.com` または `registry.example.com:8443`
- ❷ 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"
  }
}
}
}

```

3.6.2.2.3. ミラーレジストリーへのイメージのミラーリング



重要

OpenShift Update Service アプリケーションによる過度のメモリー使用を回避するには、以下の手順で説明するように、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必要があります。

前提条件

- 非接続環境で使用するミラーレジストリーを設定し、設定した証明書と認証情報にアクセスできるようにになりました。
- [Red Hat OpenShift Cluster Manager からプルシークレット](#) をダウンロードし、ミラーリポジトリーへの認証を組み込むように変更している。
- 自己署名証明書を使用する場合は、証明書にサブジェクトの別名を指定しています。

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. OpenShift Update Service を使用している場合は、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含めます。

```
$ 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。



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。

- 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. クラスターのアーキテクチャーのタイプをエクスポートします。

```
$ ARCHITECTURE=<cluster_architecture> 1
```


- 1 **x86_64**、**aarch64**、**s390x**、または **ppc64le** など、クラスターのアーキテクチャーを指定します。

- 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. 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}
```



注記

このコマンドは、ミラーリングされたリリースイメージ署名 config map も、リムーバブルメディアに保存します。

- iii. メディアを非接続環境に移動し、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。

- iv. **oc** コマンドラインインターフェイス (CLI) を使用して、更新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`)。

- vi. OpenShift Update Service を使用している場合は、リリースイメージを別のリポジトリ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_RELEASE}-
${ARCHITECTURE}
```

- ローカルコンテナ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は、次の操作を行います。
 - 次のコマンドを使用して、リリースイメージをローカルレジストリーに直接プッシュし、`config map` 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。

- OpenShift Update Service を使用している場合は、リリースイメージを別のリポジトリ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_RELEASE}-
${ARCHITECTURE}
```

3.6.3. OpenShift Update Service を使用した非接続環境でのクラスターの更新

接続されたクラスターと同じように更新するには、次の手順を使用して、非接続環境で OpenShift Update Service (OSUS) をインストールおよび設定できます。

以下の手順は、OSUS を使用して非接続環境でクラスターを更新する大まかな方法を示しています。

- セキュアなレジストリーへのアクセスを設定します。
- グローバルクラスタープルシークレットを更新して、ミラーレジストリーにアクセスします。
- OSUS Operator をインストールします。

4. OpenShift Update Service のグラフデータコンテナイメージを作成します。
5. OSUS アプリケーションをインストールし、環境内で OpenShift Update Service を使用するようクラスターを設定します。
6. 接続されたクラスターの場合と同様に、ドキュメントに記載されているサポートされている更新手順を実行します。

3.6.3.1. 非接続環境での OpenShift Update Service の使用

OpenShift Update Service (OSUS) は、OpenShift Container Platform クラスターに更新の推奨事項を提供します。Red Hat は OpenShift Update Service をパブリックにホストし、接続された環境内のクラスターは、パブリック API を介してサービスに接続して更新の推奨事項を取得できます。

ただし、非接続環境のクラスターは、これらのパブリック API にアクセスして更新情報を取得することはできません。非接続環境で同じように更新を行うには、OpenShift Update Service をインストールして設定し、非接続環境で使用できるようにします。

単一の OSUS インスタンスは、数千のクラスターに推奨事項を提供できます。レプリカ値を変更することで、OSUS を水平方向に拡張して、より多くのクラスターに対応できます。したがって、ほとんどの接続されていないユースケースでは、1つの OSUS インスタンスで十分です。たとえば、Red Hat は、接続されたクラスター全体に対して1つの OSUS インスタンスだけをホストします。

更新の推奨事項を異なる環境で個別に保持したい場合は、環境ごとに1つの OSUS インスタンスを実行できます。たとえば、テスト環境とステージ環境が別々にある場合、バージョン A がテスト環境でまだテストされていない場合、ステージ環境のクラスターがバージョン A への更新推奨を受け取らないようにすることができます。

次のセクションでは、OSUS インスタンスをインストールし、更新の推奨事項をクラスターに提供するように設定する方法について説明します。

関連情報

- [OpenShift Update Service について](#)
- [更新チャンネルとリリースについて](#)

3.6.3.2. 前提条件

- **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。
- [OpenShift Container Platform イメージのミラーリング](#) で説明されているように、更新用のコンテナイメージを使用して、環境内にコンテナイメージレジストリーをプロビジョニングする必要があります。

3.6.3.3. 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、HTTPS X.509 証明書がカスタム認証局により署名されたレジストリーに含まれている場合は、[イメージレジストリーアクセスのトラストストアの追加設定](#) の手順を完了し、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: | ❶
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com.:5000: | ❷
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----

```

- ❶ 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。
- ❷ 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。

3.6.3.4. 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。

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> ❶
```

- ❶ 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>" \ ❶
--auth-basic="<username>:<password>" \ ❷
--to=<pull_secret_location> ❸
```

- ❶ 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。



注記

OpenShift Container Platform 4.7.4 の時点で、グローバルプルシークレットへの変更によってノードドレインまたは再起動がトリガーされなくなりました。

3.6.3.5. 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

非接続環境 (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。

3.6.3.5.1. 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 のリストから **OpenShift Update Service** を選択し、**Install** をクリックします。
 - a. **Update channel** を選択します。

- b. **Version** を選択します。
 - c. **A specific namespace on the cluster**が **Installation Mode** で選択します。
 - d. **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。
 - e. **Update 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。
 - f. **Install** をクリックします。
3. **Operators** → **Installed Operators** に移動し、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. **OpenShift Update Service** が選択した namespace に、**Status** が **Succeeded** でリストされていることを確認します。

3.6.3.5.2. 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" ❶
```

- ❶ **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。
- 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
```

- 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
```

- 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環境にインストールされている場合、Operator Lifecycle Manager (OLM) を設定したときに作成された **CatalogSource** オブジェクトの名前を指定します。

- 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。

- 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。バージョン番号は表示されているものと異なる場合があります。

関連情報

- Operator を namespace にインストールする

3.6.3.6. 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、グラフデータは更新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。



注記

oc-mirror OpenShift CLI (**oc**) プラグインは、ミラーリングするリリースイメージに加えて、このグラフデータコンテナイメージを作成します。oc-mirror プラグインを使用してリリースイメージをミラーリングした場合は、この手順を省略できます。

手順

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

```
FROM registry.access.redhat.com/ubi9/ubi:latest

RUN curl -L -o cincinnati-graph-data.tar.gz
https://api.openshift.com/api/upgrades_info/graph-data

RUN mkdir -p /var/lib/cincinnati-graph-data && tar xvzf cincinnati-graph-data.tar.gz -C
/var/lib/cincinnati-graph-data/ --no-overwrite-dir --no-same-owner

CMD ["/bin/bash", "-c", "exec cp -rp /var/lib/cincinnati-graph-data/* /var/lib/cincinnati/graph-
data"]
```

- 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グラフデータ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

非接続環境でグラフデータイメージをレジストリーにプッシュするには、前の手順で作成したグラフデータコンテナイメージを、OpenShift Update Service からアクセス可能な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。

3.6.3.7. 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。

3.6.3.7.1. 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 のグラフデータ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いる。
- 現在のリリースと更新ターゲットリリースは、非接続環境のレジストリーにミラーリングされています。

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 のリストから **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** リストから、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。

3.6.3.7.2. 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 のグラフデータ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いる。
- 現在のリリースと更新ターゲットリリースは、非接続環境のレジストリーにミラーリングされています。

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定ミラーリング (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レジストリーおよびリポジトリを設定します。

```
$ 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グラフデータ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 4.16 で実行している場合は、**stable-4.16** 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。

3.6.3.8. 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、Cluster Version Operator (CVO) を更新して、環境にインストールされている OpenShift Update Service からグラフデータを取得できるようになります。

前提条件

- OpenShift Update Service Operator がインストールされている。
- OpenShift Update Service のグラフデータ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いる。
- 現在のリリースと更新ターゲットリリースは、非接続環境のレジストリーにミラーリングされています。

- 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、環境で 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の設定](#) を参照して、更新サーバーを信頼するように CA を設定してください。

3.6.3.9. 次のステップ

クラスターを更新する前に、次の条件が満たされていることを確認してください。

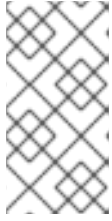
- Cluster Version Operator (CVO) が、インストールされた OpenShift Update Service アプリケーションを使用するように設定されている。
- 新しいリリースのリリースイメージ署名 config map がクラスターに適用されている。



注記

Cluster Version Operator (CVO) は、リリースイメージ署名が期待される結果と一致することを検証し、リリースイメージが変更されていないかを確認するためにリリースイメージ署名を使用します。

- 現在のリリースと更新ターゲットリリースのイメージは、非接続環境のレジストリーにミラーリングされている。
- 最近のグラフデータコンテナイメージがレジストリーにミラーリングされている。
- 最新バージョンの OpenShift Update Service Operator がインストールされている。



注記

OpenShift Update Service Operator を最近インストールまたは更新していない場合は、さらに新しいバージョンが利用できる可能性があります。非接続環境で OLM カタログを更新する方法の詳細は、[制限されたネットワーク上で Operator Lifecycle Manager を使用する](#) を参照してください。

インストールされた OpenShift Update Service とローカルミラーレジストリーを使用するようにクラスターを設定したら、次のいずれかの更新方法を使用できます。

- [Web コンソールを使用してクラスターを更新](#)
- [CLI を使用したクラスターの更新](#)
- [EUS から EUS への更新の実行](#)
- [カナリアロールアウト更新の実行](#)
- [RHEL コンピュータマシンを含むクラスターの更新](#)

3.6.4. OpenShift Update Service を使用しない非接続環境でのクラスターの更新

以下の手順を使用して、OpenShift Update Service にアクセスせずに非接続環境でクラスターを更新します。

3.6.4.1. 前提条件

- **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。
- [OpenShift Container Platform イメージのミラーリング](#) で説明されているように、更新用のコンテナイメージを使用してローカルのコンテナイメージレジストリーをプロビジョニングしている。
- **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。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、[クラスターを以前の状態に復元する](#) 必要がある場合に備えて、最新の **etcd バックアップ** を用意している。
- Operator Lifecycle Manager (OLM) を通じて以前にインストールされたすべての Operator を、ターゲットリリースと互換性のあるバージョンに更新している。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新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。[インストール済み Operator の更新](#) を参照し、互換性を確認する方法の詳細を確認して、インストールされている Operator を必要に応じて更新してください。
- 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する。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 を一時停止できる。
- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは [手動で維持された認証情報でクラスターを更新する準備](#) を参照してください。
- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設

定している場合は、更新プロセス中に中断が発生する可能性があります。 **PodDisruptionBudget** で **minAvailable** が1に設定されている場合、**削除** プロセスをブロックする可能性がある保留中のマシン設定を適用するためにノードがドレインされます。複数のノードが再起動された場合に、すべての Pod が1つのノードでのみ実行される可能性があり、 **PodDisruptionBudget** フィールドはノードのドレインを防ぐことができます。



注記

Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合は、更新プロセス中に中断が発生する可能性があります。 **PodDisruptionBudget** で **minAvailable** が1に設定されている場合、**削除** プロセスをブロックする可能性がある保留中のマシン設定を適用するためにノードがドレインされます。複数のノードが再起動された場合に、すべての Pod が1つのノードでのみ実行される可能性があり、 **PodDisruptionBudget** フィールドはノードのドレインを防ぐことができます。

3.6.4.2. MachineHealthCheck リソースの一時停止

更新プロセスで、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**) がインストールされている。

手順

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

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

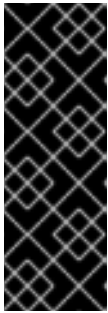
2. マシンヘルスチェックを一時停止するには、 **cluster.x-k8s.io/paused=""** アノテーションを **MachineHealthCheck** リソースに追加します。以下のコマンドを実行します。

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

アノテーション付きの **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-
```

3.6.4.3. リリースイメージダイジェストの取得

`--to-image` オプションを指定して `oc adm upgrade` コマンドを使用することで非接続環境でクラスターを更新する場合、ターゲットリリースイメージに対応する sha256 ダイジェストを参照する必要があります。

手順

1. インターネットに接続されているデバイスで、以下のコマンドを実行します。

```
$ oc adm release info -o 'jsonpath={.digest}' quay.io/openshift-release-dev/ocp-release:${OCP_RELEASE_VERSION}-${ARCHITECTURE}
```

`{OCP_RELEASE_VERSION}` では、更新する OpenShift Container Platform のバージョン (例: `4.10.16`) を指定します。

`{ARCHITECTURE}` では、クラスターアーキテクチャー (例: `x86_64`、`aarch64`、`s390x`、`ppc64le`) を指定します。

出力例

```
sha256:a8bfba3b6ddd1a2fbbbead7dac65fe4fb8335089e4e7cae327f3bad334add31d
```

2. クラスターの更新時に使用する sha256 ダイジェストをコピーします。

3.6.4.4. 切断されたクラスターの更新

切断されたクラスターを、リリースイメージをダウンロードした OpenShift Container Platform バージョンに更新します。

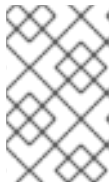


注記

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。



注記

リリースイメージ署名 config map を使用すると、Cluster Version Operator (CVO) は、実際のイメージ署名が想定された署名と一致するか検証し、リリースイメージの整合性を確保できます。

- ターゲットリリースイメージの sha256 ダイジェストを取得している。
- OpenShift CLI (**oc**) がインストールされている。
- すべての **MachineHealthCheck** リソースを一時停止している。

手順

- クラスターを更新します。

```
$ oc adm upgrade --allow-explicit-upgrade --to-image
<defined_registry>/<defined_repository>@<digest>
```

ここでは、以下のようになります。

<defined_registry>

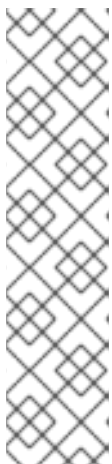
イメージのミラーリング先であるミラーレジストリーの名前を指定します。

<defined_repository>

ミラーレジストリーで使用するイメージリポジトリーの名前を指定します。

<digest>

ターゲットリリースイメージの sha256 ダイジェストを指定します (例:
sha256:81154f5c03294534e1eaf0319bef7a601134f891689ccede5d705ef659aa8c92)。



注記

- ミラーレジストリーとリポジトリー名の定義を確認するには、「OpenShift Container Platform イメージのミラーリング」を参照してください。
- **ImageContentSourcePolicy** または **ImageDigestMirrorSet** を使用した場合は、定義した名前の代わりに標準的なレジストリー名とリポジトリー名を使用できます。標準的なレジストリー名は **quay.io**、標準的なリポジトリー名は **openshift-release-dev/ocp-release** です。
- **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。

関連情報

- [OpenShift Container Platform イメージのミラーリング](#)

3.6.4.5. イメージレジストリーリポジトリーのミラーリングについて

コンテナレジストリーリポジトリーのミラーリングを設定すると、次のタスクを実行できます。

- 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。

OpenShift Container Platform のリポジトリーミラーリングには、以下の属性が含まれます。

- 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。
- 非接続環境のクラスターは、quay.io などの重要な場所からイメージをプルし、会社のファイアウォールの背後にあるレジストリーに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 に必要な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ことで、切断された環境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 のインストール中にミラーリングを設定しなかった場合は、以下のカスタムリソース (CR) オブジェクトのいずれかを使用して、インストール後に設定できます。
 - **ImageDigestMirrorSet (IDMS)**。このオブジェクトを使用すると、ダイジェスト仕様を使用して、ミラーリングされたレジストリーからイメージを取得できます。IDMS CR を使用すると、イメージのプルが失敗した場合に、ソースレジストリーからのプルの継続的な試行を許可または停止するフォールバックポリシーを設定できます。
 - **ImageTagMirrorSet (ITMS)**。このオブジェクトを使用すると、イメージタグを使用して、ミラーリングされたレジストリーからイメージをプルできます。ITMS CR を使用すると、イメージのプルが失敗した場合に、ソースレジストリーからのプルの継続的な試行を許可または停止するフォールバックポリシーを設定できます。

- **ImageContentSourcePolicy** (ICSP)。このオブジェクトを使用すると、ダイジェスト仕様を使用して、ミラーリングされたレジストリーからイメージを取得できます。ミラーが機能しない場合、ICSP CR は必ずソースレジストリーにフォールバックします。



重要

ImageContentSourcePolicy (ICSP) オブジェクトを使用してリポジトリミラーリングを設定することは、非推奨の機能です。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。**ImageContentSourcePolicy** オブジェクトの作成に使用した既存の YAML ファイルがある場合は、**oc adm migrate icsp** コマンドを使用して、それらのファイルを **ImageDigestMirrorSet** YAML ファイルに変換できます。詳細は、次のセクションの「イメージレジストリーリポジトリミラーリング用の **ImageContentSourcePolicy** (ICSP) ファイルの変換」を参照してください。

これらのカスタムリソースオブジェクトはそれぞれ、次の情報を識別します。

- 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。

新しいクラスターの場合は、必要に応じて IDMS、ITMS、および ICSP CR オブジェクトを使用できます。ただし、IDMS と ITMS の使用を推奨します。

クラスターをアップグレードした場合、既存の ICSP オブジェクトは安定を維持し、IDMS オブジェクトと ICSP オブジェクトの両方がサポートされるようになります。ICSP オブジェクトを使用するワークロードは、引き続き期待どおりに機能します。一方、IDMS CR で導入されたフォールバックポリシーを利用する場合は、**oc adm merge icsp** コマンドを使用して、現在のワークロードを IDMS オブジェクトに移行できます。これについては、後述の **イメージレジストリーリポジトリミラーリング用の ImageContentSourcePolicy (ICSP) ファイルの変換** セクションで説明しています。IDMS オブジェクトへの移行に、クラスターの再起動は必要ありません。



注記

クラスターで **ImageDigestMirrorSet**、**ImageTagMirrorSet**、または **ImageContentSourcePolicy** オブジェクトを使用してリポジトリ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は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。

3.6.4.5.1. イメージレジストリーのリポジトリミラーリングの設定

インストール後のミラー設定カスタムリソース (CR) を作成して、ソースイメージレジストリーからミラーリングされたイメージレジストリーにイメージプル要求をリダイレクトできます。

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。

手順

1. ミラーリングされたリ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

ングされたリ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、ソースリポジトリからミラーリングされたリ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/ubi9/ubi-minimal:latest@sha256:5cf... \
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があり、そこに **registry.access.redhat.com** から **ubi9/ubi-minimal** イメージをコピーします。ミラーリングされたレジストリーを作成した後、ソースリポジトリに対する要求をミラーリングされたリポジトリにリダイレクトするように OpenShift Container Platform クラスターを設定できます。

2. OpenShift Container Platform クラスターにログインします。

3. 次の例のいずれかを使用して、インストール後のミラー設定 CR を作成します。

- 必要に応じて **ImageDigestMirrorSet** または **ImageTagMirrorSet** CR を作成し、ソースとミラーを独自のレジストリーとリポジトリのペアとイメージに置き換えます。

```
apiVersion: config.openshift.io/v1 1
kind: ImageDigestMirrorSet 2
metadata:
  name: ubi9repo
spec:
  imageDigestMirrors: 3
  - mirrors:
    - example.io/example/ubi-minimal 4
    - example.com/example/ubi-minimal 5
    source: registry.access.redhat.com/ubi9/ubi-minimal 6
    mirrorSourcePolicy: AllowContactingSource 7
  - mirrors:
    - mirror.example.com/redhat
    source: registry.example.com/redhat 8
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.com
    source: registry.example.com 9
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net/image
    source: registry.example.com/example/myimage 10
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net
    source: registry.example.com/example 11
    mirrorSourcePolicy: AllowContactingSource
```

```
- mirrors:
  - mirror.example.net/registry-example-com
  source: registry.example.com 12
  mirrorSourcePolicy: AllowContactingSource
```

- 1 この CR で使用する API を示します。これは **config.openshift.io/v1** である必要があります。
 - 2 プルタイプに応じてオブジェクトの種類を示します。
 - **ImageDigestMirrorSet**: ダイジェスト参照イメージをプルします。
 - **ImageTagMirrorSet**: タグ参照イメージをプルします。
 - 3 次のいずれかのイメージプルメソッドのタイプを示します。
 - **imageDigestMirrors**: **ImageDigestMirrorSet** CR に使用します。
 - **imageTagMirrors**: **ImageTagMirrorSet** CR に使用します。
 - 4 ミラーリングされたイメージのレジストリーとリポジトリーの名前を示します。
 - 5 オプション: 各ターゲットリポジトリーのセカンダリーミラーリポジトリーを示します。1つのミラーがダウンした場合、ターゲットリポジトリーは別のミラーを使用できません。
 - 6 イメージプル仕様で参照されるリポジトリーである、レジストリーおよびリポジトリーソースを示します。
 - 7 オプション: イメージのプルが失敗した場合のフォールバックポリシーを示します。
 - **AllowContactingSource**: ソースリポジトリーからのイメージのプルの継続的な試行を許可します。これはデフォルトになります。
 - **NeverContactSource**: ソースリポジトリーからのイメージのプルの継続的な試行を防ぎます。
 - 8 オプション: レジストリー内の namespace を示します。これにより、その namespace で任意のイメージを使用できます。レジストリードメインをソースとして使用する場合、オブジェクト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。
 - 9 オプション: レジストリーを示し、そのレジストリー内の任意のイメージを使用できるようにします。レジストリー名を指定すると、ソースレジストリーからミラーレジストリーまでのすべてのリポジトリーにオブジェクトが適用されます。
 - 10 イメージ **registry.example.com/example/myimage@sha256:...** をミラー **mirror.example.net/image@sha256:...** からプルします。
 - 11 ミラー **mirror.example.net/image@sha256:...** からソースレジストリー namespace のイメージ **registry.example.com/example/image@sha256:...** をプルします。
 - 12 ミラーレジストリー **example.net/registry-example-com/myimage@sha256:...** からイメージ **registry.example.com/myimage@sha256** をプルします。
- **ImageContentSourcePolicy** カスタムリソースを作成し、ソースとミラーを独自のレジストリーとリポジトリーのペアとイメージに置き換えます。

```

apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mirror-ocp
spec:
  repositoryDigestMirrors:
  - mirrors:
    - mirror.registry.com:443/ocp/release ❶
    source: quay.io/openshift-release-dev/ocp-release ❷
  - mirrors:
    - mirror.registry.com:443/ocp/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

```

- ❶ ミラーイメージレジストリーおよびリポジトリーの名前を指定します。
- ❷ ミラーリングされるコンテンツが含まれるオンラインレジストリーおよびリポジトリーを指定します。

4. 新規オブジェクトを作成します。

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

オブジェクトの作成後、Machine Config Operator (MCO) は **ImageTagMirrorSet** オブジェクトのみのノードをドレインします。MCO は、**ImageDigestMirrorSet** オブジェクトと **ImageContentSourcePolicy** オブジェクトのノードをドレインしません。

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 | v1.29.4 |
| ip-10-0-138-148.ec2.internal | Ready | master | 11m | v1.29.4 |
| ip-10-0-139-122.ec2.internal | Ready | master | 11m | v1.29.4 |
| ip-10-0-147-35.ec2.internal | Ready | worker | 7m | v1.29.4 |
| ip-10-0-153-12.ec2.internal | Ready | worker | 7m | v1.29.4 |
| ip-10-0-154-10.ec2.internal | Ready | master | 11m | v1.29.4 |

- 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. ルートディレクトリーを **/host** に変更します。

```
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
```

次の出力は、インストール後のミラー設定 CR が適用された **registries.conf** ファイルを表しています。最後の2つのエントリーは、それぞれ **digest-only** および **tag-only** とマークされています。

出力例

```
unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
short-name-mode = ""
```

```
[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi9/ubi-minimal" 1
```

```
[[registry.mirror]]
  location = "example.io/example/ubi-minimal" 2
  pull-from-mirror = "digest-only" 3
```

```
[[registry.mirror]]
  location = "example.com/example/ubi-minimal"
  pull-from-mirror = "digest-only"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com"
```

```
[[registry.mirror]]
  location = "mirror.example.net/registry-example-com"
  pull-from-mirror = "digest-only"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com/example"
```

```
[[registry.mirror]]
  location = "mirror.example.net"
  pull-from-mirror = "digest-only"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com/example/myimage"
```

```
[[registry.mirror]]
  location = "mirror.example.net/image"
  pull-from-mirror = "digest-only"
```

```
[[registry]]
```

```

prefix = ""
location = "registry.example.com"

[[registry.mirror]]
location = "mirror.example.com"
pull-from-mirror = "digest-only"

[[registry]]
prefix = ""
location = "registry.example.com/redhat"

[[registry.mirror]]
location = "mirror.example.com/redhat"
pull-from-mirror = "digest-only"
[[registry]]
prefix = ""
location = "registry.access.redhat.com/ubi9/ubi-minimal"
blocked = true ❹

[[registry.mirror]]
location = "example.io/example/ubi-minimal-tag"
pull-from-mirror = "tag-only" ❺

```

- ❶ プルスペックで参照されるリポジトリを示します。
- ❷ そのリポジトリのミラーを示します。
- ❸ ミラーからプルされたイメージがダイジェスト参照イメージであることを示します。
- ❹ このリポジトリに **NeverContactSource** パラメーターが設定されていることを示します。
- ❺ ミラーからプルされたイメージがタグ参照イメージであることを示します。

- e. ソースからノードにイメージをプルし、ミラーによって解決されるかどうかを確認します。

```

sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi9/ubi-minimal@sha256:5cf...

```

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。

3.6.4.5.2. イメージレジストリーリポジトリーミラーリング用の ImageContentSourcePolicy (ICSP) ファイルの変換

ImageContentSourcePolicy (ICSP) オブジェクトを使用してリポジトリーミラーリングを設定することは、非推奨の機能です。この機能は引き続き OpenShift Container Platform に含まれており、引き続きサポートされます。ただし、この製品の将来のリリースでは削除される予定であり、新しいデプロイメントには推奨されません。

ICSP オブジェクトは、リポジトリーミラーリングを設定するために **ImageDigestMirrorSet** および **ImageTagMirrorSet** オブジェクトに置き換えられています。 **ImageContentSourcePolicy** オブジェクトの作成に使用した既存の YAML ファイルがある場合は、 **oc adm migrate icsp** コマンドを使用して、それらのファイルを **ImageDigestMirrorSet** YAML ファイルに変換できます。このコマンドは、API を現在のバージョンに更新し、 **kind** 値を **ImageDigestMirrorSet** に変更し、 **spec.repositoryDigestMirrors** を **spec.imageDigestMirrors** に変更します。ファイルの残りの部分は変更されません。

移行によって **registries.conf** ファイルは変更されないため、クラスターを再起動する必要はありません。

ImageDigestMirrorSet または **ImageTagMirrorSet** オブジェクトの詳細については、前のセクションの「イメージレジストリーリポジトリーミラーリングの設定」を参照してください。

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。
- クラスターに **ImageContentSourcePolicy** オブジェクトがあることを確認します。

手順

1. 次のコマンドを使用して、1つ以上の **ImageContentSourcePolicy** YAML ファイルを **ImageDigestMirrorSet** YAML ファイルに変換します。

```
$ oc adm migrate icsp <file_name>.yaml <file_name>.yaml <file_name>.yaml --dest-dir <path_to_the_directory>
```

ここでは、以下ようになります。

<file_name>

ソース **ImageContentSourcePolicy** YAML の名前を指定します。複数のファイル名をリストできます。

--dest-dir

オプション: 出力 **ImageDigestMirrorSet** YAML のディレクトリーを指定します。設定されていない場合、ファイルは現在のディレクトリーに書き込まれます。

たとえば、次のコマンドは **icsp.yaml** および **icsp-2.yaml** ファイルを変換し、新しい YAML ファイルを **idms-files** ディレクトリーに保存します。

```
$ oc adm migrate icsp icsp.yaml icsp-2.yaml --dest-dir idms-files
```

出力例

```
wrote ImageDigestMirrorSet to idms-
files/imagetagmirrorset_ubi8repo.5911620242173376087.yaml
```



```
wrote ImageDigestMirrorSet to idms-
files/imagetdigestmirrorset_ubi9repo.6456931852378115011.yaml
```

2. 次のコマンドを実行して CR オブジェクトを作成します。

```
$ oc create -f <path_to_the_directory>/<file-name>.yaml
```

ここでは、以下のようになります。

<path_to_the_directory>

--dest-dir フラグを使用した場合は、ディレクトリーへのパスを指定します。

<file_name>

ImageDigestMirrorSet YAML の名前を指定します。

3. IDMS オブジェクトがロールアウトされた後、ICSP オブジェクトを削除します。

3.6.4.6. 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**) がインストールされている。
- **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:v4.16**)。

<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 OpenShift 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:
      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。

3.6.4.7. 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](#)

3.6.5. クラスターからの OpenShift Update Service のアンインストール

OpenShift Update Service (OSUS) のローカルコピーをクラスターから削除するには、最初に OSUS アプリケーションを削除してから、OSUS Operator をアンインストールする必要があります。

3.6.5.1. 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。

3.6.5.1.1. 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 のリストから **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リストから、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。

3.6.5.1.2. 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
```

3.6.5.2. OpenShift Update Service Operator のアンインストール

OpenShift Container Platform Web コンソールまたは CLI を使用して、OpenShift Update Service Operator をアンインストールできます。

3.6.5.2.1. 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 コンソールを使用して 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 のリストから **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。

3.6.5.2.2. 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
```

- 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
```

3.7. 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 15 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

- OpenShift Container Platform のバージョン 4.16 には、VMware 仮想ハードウェアバージョン 15 以降が必要です。
- OpenShift 4.12 を OpenShift 4.13 にアップグレードする前に、vSphere を **v7.0.2 以降** に更新する必要があります。それ以外の場合、OpenShift 4.12 クラスターは **un-upgradeable** とマークされます。

3.7.1. 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。



重要

OpenShift Container Platform 4.13 の時点で、VMware 仮想ハードウェアバージョン 13 はサポート対象外になりました。機能をサポートするには、VMware バージョン 15 以降に更新する必要があります。

3.7.1.1. 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更新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 7.0U2 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リスト表示します。

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

出力例

```
NAME                STATUS  ROLES  AGE  VERSION
control-plane-node-0 Ready   master 75m  v1.29.4
control-plane-node-1 Ready   master 75m  v1.29.4
control-plane-node-2 Ready   master 75m  v1.29.4
```

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 VM を更新します。詳細については、VMware ドキュメントの [仮想マシンの互換性を手動でアップグレードする](#) に従ってください。
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。

3.7.1.2. 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更新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更新できます。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 7.0U2 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リスト表示します。

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

出力例

```
NAME           STATUS  ROLES  AGE  VERSION
compute-node-0 Ready   worker 30m  v1.29.4
compute-node-1 Ready   worker 30m  v1.29.4
compute-node-2 Ready   worker 30m  v1.29.4
```

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 VM を更新します。詳細については、VMware ドキュメントの [仮想マシンの互換性を手動でアップグレードする](#)に従ってください。
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。

3.7.1.3. vSphere 上のテンプレートの仮想ハードウェアの更新

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 7.0U2 以降を使用している。

手順

1. RHCOS テンプレートが vSphere テンプレートとして設定されている場合は、次のステップの前に、VMware ドキュメントの [テンプレートを仮想マシンに変換する](#)に従ってください。



注記

テンプレートから変換したら、仮想マシンをパワーオンしないでください。

2. vSphere クライアントで VM を更新します。詳細については、VMware ドキュメントの [仮想マシンの互換性を手動でアップグレードする](#)に従ってください。
3. vSphere クライアントの VM を VM からテンプレートに変換します。詳細については、VMware ドキュメントの [vSphere Client で仮想マシンをテンプレートに変換する](#)に従ってください。

関連情報

- [ノード上の Pod を退避させる方法](#)

3.7.2. 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 の更新実行前に、更新をスケジュールする場合には、OpenShift Container Platform の更新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。

3.8. マルチアーキテクチャーのコンピュータマシンを備えたクラスターへの移行

マルチアーキテクチャーのマニフェストがリストされたペイロードに更新することで、シングルアーキテクチャーのコンピュータマシンを持つクラスターに、現在のクラスターを移行できます。これにより、混合アーキテクチャーのコンピュータノードをクラスターに追加できます。

マルチアーキテクチャーコンピュータマシンの設定については、「OpenShift Container Platform クラスターでのマルチアーキテクチャーコンピュータマシンの設定」を参照してください。

単一アーキテクチャークラスターをマルチアーキテクチャーコンピュータマシンを備えたクラスターに移行する前に、Multiarch Tuning Operator をインストールし、**ClusterPodPlacementConfig** カスタムリソースをデプロイすることを推奨します。詳細は、「Multiarch Tuning Operator を使用したマルチアーキテクチャークラスターでのワークロードの管理」を参照してください。



重要

マルチアーキテクチャーペイロードからシングルアーキテクチャーペイロードへの移行はサポートされていません。クラスターが移行されてマルチアーキテクチャーペイロードを使用するようになると、シングルアーキテクチャー更新ペイロードを受け入れなくなります。

3.8.1. マルチアーキテクチャーコンピュータマシンが含まれるクラスターへの CLI を使用した移行

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。
- OpenShift Container Platform のバージョンが 4.13.0 以上である。
クラスターバージョンの更新方法について、詳しくは [Web コンソールを使用したクラスター更新](#) または [CLI を使用したクラスター更新](#) を参照してください。
- 現在のクラスターバージョンに一致する OpenShift CLI (**oc**) がインストールされている。
- **oc** クライアントは version 4.13.0 以降に更新されている。
- OpenShift Container Platform クラスターが、AWS、Azure、GCP、ベアメタル、または IBM P/Z プラットフォームにインストールされている。
クラスターインストールでサポートされるプラットフォームの選択について、詳しくは [クラスターインストールタイプの選択](#) を参照してください。

手順

1. 以下のコマンドを実行して、Cluster Version Operator (CVO) で **RetrievedUpdates** 条件が **True** になっていることを確認します。

```
$ oc get clusterversion/version -o=jsonpath="{.status.conditions[?
.type=='RetrievedUpdates']}.status}"
```

RetrievedUpdates 条件が **False** の場合、以下のコマンドを使用して障害に関する補足情報を見つけることができます。

```
$ oc adm upgrade
```

クラスターバージョンの条件タイプに関する詳細は、[クラスターバージョンの条件タイプについて](#) を参照してください。

2. **RetrievedUpdates** 条件が **False** の場合は、次のコマンドを実行してチャンネルを **stable-<4.y>** または **fast-<4.y>** に変更します。

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

チャンネルを設定したら、**RetrievedUpdates** が **True** になっているか確認します。

チャンネルの詳細は、[更新チャンネルとリリースについて](#) を参照してください。

3. 以下のコマンドを実行して、マルチアーキテクチャーペイロードに移行します。

```
$ oc adm upgrade --to-multi-arch
```

検証

- 以下のコマンドを実行して移行をモニタリングできます。

```
$ oc adm upgrade
```



重要

クラスターが新たな状態になると、マシンの起動が失敗する可能性があります。マシンの起動失敗を認識し、回復させるために、マシンヘルスチェックをデプロイすることが推奨されます。マシンヘルスチェックとそのデプロイ方法の詳細は、[マシンヘルスチェックについて](#) を参照してください。

クラスターに異なるアーキテクチャーを持つコンピュータマシンセットを追加するのは、移行が完了し、すべてのクラスター Operator が安定した後でなければなりません。

関連情報

- [OpenShift Container Platform クラスターでのマルチアーキテクチャーコンピュータマシンの設定](#)
- [Multiarch Tuning Operator を使用したマルチアーキテクチャークラスター上のワークロードの管理](#)
- [Web コンソールを使用してクラスターを更新](#)
- [CLI を使用したクラスターの更新](#)
- [クラスターバージョン条件タイプについて](#)

- [更新チャンネルとリリースについて](#)
- [クラスターのインストールタイプの選択](#)
- [マシンのヘルスチェック](#)

3.9. HOSTED CONTROL PLANE の更新

OpenShift Container Platform のホストされたコントロールプレーンでは、更新はコントロールプレーンとノード間で切り離されます。クラスターコントロールプレーンをホストするユーザーであるサービスクラスタープロバイダーは、必要に応じて更新を管理できます。ホストされたクラスターはコントロールプレーンの更新を処理し、ノードプールはノードの更新を処理します。

3.9.1. ホストされたクラスターの更新

spec.release 値は、コントロールプレーンのバージョンを決定します。**HostedCluster** オブジェクトは、意図した **spec.release** 値を **HostedControlPlane.spec.release** 値に送信し、適切なコントロールプレーン Operator バージョンを実行します。

Hosted Control Plane は、新しいバージョンの Cluster Version Operator (CVO) により、新しいバージョンのコントロールプレーンコンポーネントと OpenShift Container Platform コンポーネントのロールアウトを管理します。

3.9.2. ノードプールの更新

ノードプールを使用すると、**spec.release** および **spec.config** の値を公開することで、ノードで実行されているソフトウェアを設定できます。次の方法でノードプールのローリング更新を開始できます。

- **spec.release** または **spec.config** の値を変更します。
- AWS インスタンスタイプなどのプラットフォーム固有のフィールドを変更します。結果は、新しいタイプの新規インスタンスのセットになります。
- クラスター設定を変更します (変更がノードに伝播される場合)。

ノードプールは、置換更新とインプレース更新をサポートします。**nodepool.spec.release** 値は、特定のノードプールのバージョンを決定します。**NodePool** オブジェクトは、**.spec.management.upgradeType** 値に従って、置換またはインプレースローリング更新を完了します。

ノードプールを作成した後は、更新タイプは変更できません。更新タイプを変更する場合は、ノードプールを作成し、他のノードプールを削除する必要があります。

3.9.2.1. ノードプールの置き換え更新

置き換え更新では、以前のバージョンから古いインスタンスが削除され、新しいバージョンでインスタンスが作成されます。この更新タイプは、このレベルの不変性がコスト効率に優れているクラウド環境で効果的です。

置き換え更新では、ノードが完全に再プロビジョニングされるため、手動による変更は一切保持されません。

3.9.2.2. ノードプールのインプレース更新

インプレース更新では、インスタンスのオペレーティングシステムが直接更新されます。このタイプは、ベアメタルなど、インフラストラクチャーの制約が高い環境に適しています。

インプレース更新では手動による変更を保存できますが、kubelet 証明書など、クラスターが直接管理するファイルシステムまたはオペレーティングシステムの設定に手動で変更を加えると、エラーが報告されます。

3.9.3. ホストされたコントロールプレーンのノードプールの設定

ホストされたコントロールプレーンでは、管理クラスターの config map 内に **MachineConfig** オブジェクトを作成することでノードプールを設定できます。

手順

1. 管理クラスターの config map 内に **MachineConfig** オブジェクトを作成するには、次の情報を入力します。

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: <configmap-name>
  namespace: clusters
data:
  config: |
    apiVersion: machineconfiguration.openshift.io/v1
    kind: MachineConfig
    metadata:
      labels:
        machineconfiguration.openshift.io/role: worker
      name: <machineconfig-name>
    spec:
      config:
        ignition:
          version: 3.2.0
        storage:
          files:
            - contents:
                source: data:...
              mode: 420
              overwrite: true
              path: ${PATH} ①

```

- ① **MachineConfig** オブジェクトが保存されているノード上のパスを設定します。

2. オブジェクトを config map に追加した後、次のように config map をノードプールに適用できます。

```

spec:
  config:
    - name: ${CONFIGMAP_NAME}

```

3.10. BOOTUPD を使用して RHCOS ノード上のブートローダーを更新する

bootupd を使用して RHCOS ノード上のブートローダーを更新するには、RHCOS マシン上で **bootupctl update** コマンドを手動で実行するか、**systemd** ユニットを使用してマシン設定を指定する必要があります。

grubby またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。カーネル引数を設定するには、ノードにカーネル引数を追加するを参照してください。



注記

bootupd を使用してブートローダーを更新すると、BootHole 脆弱性から保護できません。

3.10.1. ブートローダーを手動で更新する

bootupctl コマンドラインツールを使用して、システムのステータスを手動で検査し、ブートローダーを更新できます。

1. システムのステータスを検査します。

```
# bootupctl status
```

x86_64 の出力例

```
Component EFI
Installed: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
Update: At latest version
```

aarch64 の出力例

```
Component EFI
Installed: grub2-efi-aa64-1:2.02-99.el8_4.1.aarch64,shim-aa64-15.4-2.el8_1.aarch64
Update: At latest version
```

2. 最初にバージョン 4.4 以前にインストールされた OpenShift Container Platform クラスターには、明示的な導入フェーズが必要です。システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
```

3.10.2. マシン設定を通してブートローダーを自動更新する

bootupd を使用してブートローダーを自動更新するもう1つの方法は、必要に応じて起動するたびにブートローダーを更新する **systemd** サービスユニットを作成することです。このユニットは、ブートプロセス中に **bootupctl update** コマンドを実行し、マシン設定を通してノードにインストールされます。



注記

更新操作が予期せず中断されるとノードが起動不能になる可能性があるため、この設定はデフォルトでは有効になっていません。この設定を有効にする場合は、ブートローダーを更新する間、ブートプロセス中にノードが中断されないように注意してください。通常、ブートローダーの更新操作はすぐに完了するため、リスクは低くなります。

1. **bootupctl-update.service** systemd ユニットの内容を含む Butane 設定ファイル **99-worker-bootupctl-update.bu** を作成します。



注記

Butane の詳細は、「Butane を使用したマシン設定の作成」を参照してください。

出力例

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony ①
  labels:
    machineconfiguration.openshift.io/role: worker ②
systemd:
  units:
  - name: bootupctl-update.service
    enabled: true
    contents: |
      [Unit]
      Description=Bootupd automatic update

      [Service]
      ExecStart=/usr/bin/bootupctl update
      RemainAfterExit=yes

      [Install]
      WantedBy=multi-user.target
```

- ① ② コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-bootupctl-update.yaml**) を生成します。

```
$ butane 99-worker-bootupctl-update.bu -o 99-worker-bootupctl-update.yaml
```

3. 以下の2つの方法のいずれかで設定を適用します。

- クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを `<installation_directory>/openshift` ディレクトリーに追加してから、クラスターの作成を続行します。
- クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-bootupctl-update.yaml
```

第4章 クラスター更新のトラブルシューティング

4.1. クラスター更新に関するデータ収集

更新に関する問題について Red Hat サポートに問い合わせる場合は、失敗したクラスター更新のトラブルシューティングに使用するデータをサポートチームに提供してください。

4.1.1. サポートケース用のログデータ収集

クラスターからログデータなどのデータを収集するには、**oc adm must-gather** コマンドを使用します。クラスターに関するデータの収集を参照してください。

4.1.2. ClusterVersion 履歴の収集

Cluster Version Operator (CVO) は、クラスターに行われた更新を記録します。これは、ClusterVersion 履歴と呼ばれています。そのエントリにより、クラスター動作の変化と潜在的なトリガーとの相関関係が明らかになりますが、相関関係は因果関係を示すものではありません。



注記

初期、マイナー、および z-stream バージョンの更新は、ClusterVersion 履歴として保存されます。ただし、ClusterVersion 履歴にはサイズ制限があります。制限に達すると、制限に対応するために、以前のマイナーバージョンの最も古い z-stream 更新が削除されます。

ClusterVersion 履歴は、OpenShift Container Platform Web コンソールまたは OpenShift CLI (**oc**) を使用して表示できます。

4.1.2.1. OpenShift Container Platform Web コンソールで ClusterVersion 履歴を収集する

OpenShift Container Platform Web コンソールで ClusterVersion 履歴を表示できます。

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。
- OpenShift Container Platform Web コンソールにアクセスできる。

手順

- 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。

4.1.2.2. OpenShift CLI (oc) を使用して ClusterVersion 履歴を収集します。

OpenShift CLI (**oc**) を使用して、ClusterVersion 履歴を表示できます。

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

手順

1. 以下のコマンドを入力して、クラスターの更新履歴を表示します。

```
$ oc describe clusterversions/version
```

出力例

```
Desired:
Channels:
  candidate-4.13
  candidate-4.14
  fast-4.13
  fast-4.14
  stable-4.13
Image: quay.io/openshift-release-dev/ocp-
release@sha256:a148b19231e4634196717c3597001b7d0af91bf3a887c03c444f59d9582864f4

URL: https://access.redhat.com/errata/RHSA-2023:6130
Version: 4.13.19
History:
Completion Time: 2023-11-07T20:26:04Z
Image: quay.io/openshift-release-dev/ocp-
release@sha256:a148b19231e4634196717c3597001b7d0af91bf3a887c03c444f59d9582864f4

Started Time: 2023-11-07T19:11:36Z
State: Completed
Verified: true
Version: 4.13.19
Completion Time: 2023-10-04T18:53:29Z
Image: quay.io/openshift-release-dev/ocp-
release@sha256:eac141144d2ecd6cf27d24efe9209358ba516da22becc5f0abc199d25a9ccec

Started Time: 2023-10-04T17:26:31Z
State: Completed
Verified: true
Version: 4.13.13
Completion Time: 2023-09-26T14:21:43Z
Image: quay.io/openshift-release-dev/ocp-
release@sha256:371328736411972e9640a9b24a07be0af16880863e1c1ab8b013f9984b4ef72
7
Started Time: 2023-09-26T14:02:33Z
State: Completed
Verified: false
Version: 4.13.12
Observed Generation: 4
Version Hash: CMLI3sLq-EA=
Events: <none>
```

関連情報

- [クラスターに関するデータの収集](#)

