



OpenShift Container Platform 4.9

クラスタの更新

OpenShift Container Platform クラスタの更新

OpenShift Container Platform 4.9 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 © 2023 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 クラスターの更新の概要	4
1.1. OPENSIFT CONTAINER PLATFORM の更新について	4
1.2. アップグレードチャンネルとリリースを理解	4
1.3. クラスター OPERATOR の状態タイプについて	4
1.4. クラスターバージョン条件タイプについて	5
1.5. EUS から EUS への更新を実行するための準備	5
1.6. WEB コンソールを使用してクラスターを更新	5
1.7. コマンドラインインターフェイス (CLI) を使用したマイナーバージョン内のクラスターの更新	6
1.8. カナリアロールアウト更新の実行	6
1.9. RHEL コンピュータマシンを含むクラスターの更新	6
1.10. 非接続環境でのクラスターの更新	7
1.11. VSPHERE で稼働するノードでのハードウェアの更新	7
第2章 OPENSIFT CONTAINER PLATFORM の更新について	8
2.1. OPENSIFT UPDATE SERVICE について	8
2.2. 一般的な用語	9
第3章 アップグレードチャンネルとリリースを理解	11
3.1. チャンネルおよびリリースパスのアップグレード	11
第4章 OPENSIFT CONTAINER PLATFORM 4.9 への更新の準備	15
4.1. KUBERNETES API の削除	15
4.2. 削除された API に関するクラスターの評価	16
4.3. 削除された API インスタンスのインスタンスの移行	18
4.4. 管理者の確認の提供	18
第5章 EUS から EUS への更新を実行するための準備	20
5.1. EUS から EUS への更新	20
第6章 WEB コンソールを使用してクラスターを更新	23
6.1. 前提条件	23
6.2. カナリアロールアウト更新の実行	24
6.3. 手動でメンテナンスされた認証情報を使用したクラスターのアップグレード	25
6.4. WEB コンソールを使用した MACHINEHEALTHCHECK リソースの一時停止	26
6.5. 単一ノードの OPENSIFT CONTAINER PLATFORM の更新	27
6.6. WEB コンソールを使用したクラスターの更新	28
6.7. WEB コンソールを使用した更新サーバーの変更	29
第7章 CLI を使用したクラスターの更新	30
7.1. 前提条件	30
7.2. MACHINEHEALTHCHECK リソースの一時停止	31
7.3. 単一ノードの OPENSIFT CONTAINER PLATFORM の更新	32
7.4. CLI を使用したクラスターの更新	32
7.5. CLI を使用した更新サーバーの変更	36
第8章 カナリアロールアウト更新の実行	37
8.1. カナリアロールアウト更新プロセスおよび MCP について	38
8.2. カナリアロールアウト更新の実行について	38
8.3. カナリアロールアウト更新を実行するためのマシン設定プールの作成	39
8.4. マシン設定プールの一時停止	41
8.5. クラスターの更新の実行	42
8.6. マシン設定プールの一時停止の解除	42
8.7. ノードを元のマシン設定プールに移行	42

第9章 RHEL コンピュータマシンを含むクラスターの更新	44
9.1. 前提条件	44
9.2. WEB コンソールを使用したクラスターの更新	44
9.3. 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	45
9.4. クラスター内の RHEL コンピュータマシンの更新	47
第10章 非接続環境でのクラスターの更新	51
10.1. 非接続環境でのクラスターの更新について	51
10.2. OPENSIFT CONTAINER PLATFORM イメージリポジトリのミラーリング	51
10.3. OPENSIFT UPDATE SERVICE を使用した非接続環境でのクラスターの更新	59
10.4. OPENSIFT UPDATE SERVICE を使用しない非接続環境でのクラスターの更新	69
10.5. クラスターからの OPENSIFT UPDATE SERVICE のアンインストール	77
第11章 VSPHERE で稼働するノードでのハードウェアの更新	81
11.1. VSPHERE での仮想ハードウェアの更新	81
11.2. VSPHERE での仮想ハードウェアの更新のスケジューリング	84

第1章 クラスターの更新の概要

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

1.1. OPENSIFT CONTAINER PLATFORM の更新について

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

1.2. アップグレードチャンネルとリリースを理解

[アップグレードチャンネルとリリース](#): アップグレードチャンネルを使用すると、アップグレード戦略を選択できます。アップグレードチャンネルは OpenShift Container Platform のマイナーバージョン固有のもので、アップグレードチャンネルはリリースの選択のみを制御し、インストールするクラスターのバージョンには影響しません。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、常にそのマイナーバージョンをインストールします。詳細は、以下を参照してください。

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

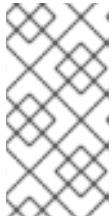
1.3. クラスター OPERATOR の状態タイプについて

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、通常のアップグレード中、**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: 状態タイプが **Upgradeable** の Operator は、Operator が現在のクラスターの状態に基づいて安全にアップグレードできるかどうかを示します。メッセージフィールドには、クラスター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表示されます。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ーが実行できないようにします。

1.4. クラスターバージョン条件タイプについて

Cluster Version Operator (CVO) は、クラスター Operator およびその他のコンポーネントを監視し、クラスターバージョンとその 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 クラスター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、クラスターのバージョンと Operator に影響する条件タイプがあります。

- Failing: クラスターバージョン条件タイプ **Failing** は、クラスター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: クラスター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ー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: クラスター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 recent および accurate である場合は **True** です。

1.5. EUS から EUS への更新を実行するための準備

EUS から EUS への更新を実行するための準備: 基本的な Kubernetes 設計のため、マイナーバージョン間のすべての OpenShift Container Platform 更新をシリアル化する必要があります。OpenShift Container Platform 4.8 から 4.9 に更新してから、4.10 に更新する必要があります。OpenShift Container Platform 4.8 から 4.10 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する場合は、コントロールプレーン以外のホストを1回再起動するだけで更新できます。詳細は、以下を参照してください。

- [EUS から EUS への更新](#)

1.6. WEB コンソールを使用してクラスターを更新

Web コンソールを使用したマイナーバージョン内のクラスターの更新: Web コンソールを使用して OpenShift Container Platform クラスターを更新できます。次の手順は、マイナーバージョン内のクラスターを更新します。マイナーバージョン間でクラスターを更新する場合も、同じ手順を使用できます。

- [カナリアロールアウト更新の実行](#)
- [MachineHealthCheck リソースの一時停止](#)
- [単一ノードクラスターでの OpenShift Container Platform の更新について](#)
- [Web コンソールを使用したクラスターの更新](#)
- [Web コンソールを使用した更新サーバーの変更](#)

1.7. コマンドラインインターフェイス (CLI) を使用したマイナーバージョン内のクラスターの更新

CLI の使用によるマイナーバージョン内でのクラスターの更新: OpenShift CLI (**oc**) を使用して、マイナーバージョン内で OpenShift Container Platform クラスターを更新できます。次の手順は、マイナーバージョン内のクラスターを更新します。マイナーバージョン間でクラスターを更新する場合も、同じ手順を使用できます。

- [MachineHealthCheck リソースの一時停止](#)
- [単一ノードクラスターでの OpenShift Container Platform の更新について](#)
- [CLI を使用したクラスターの更新](#)
- [CLI を使用した更新サーバーの変更](#)

1.8. カナリアロールアウト更新の実行

カナリアロールアウト更新の実行: ワーカーノードへの更新のロールアウトを制御することで、更新プロセスによってアプリケーションに障害が発生した場合でも、ミッションクリティカルなアプリケーションを更新全体を通じて利用できるようにすることができます。組織のニーズによっては、ワーカーノードの小規模なサブセットを更新し、一定期間でクラスターおよびワークロードの正常性を評価し、残りのノードを更新する必要がある場合があります。これは **カナリア更新**と呼ばれます。または、クラスター全体を一度に更新するために大きなメンテナンスウィンドウを使用できない場合は、ホストの再起動が必要になることが多いワーカーノードの更新を、定義済みの小さなメンテナンスウィンドウに収めることもできます。次の手順を実行できます。

- [カナリアロールアウトの更新を実行するためのマシン設定プールの作成](#)
- [マシン設定プールの一時停止](#)
- [クラスターの更新の実行](#)
- [マシン設定プールの一時停止の解除](#)
- [ノードを元のマシン設定プールに移動](#)

1.9. RHEL コンピュータマシンを含むクラスターの更新

RHEL コンピューティングマシンを含むクラスターの更新: OpenShift Container Platform クラスターを更新できます。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。次の手順を実行できます。

- [Web コンソールを使用したクラスターの更新](#)
- [オプション: RHEL マシンで Ansible タスクを実行するためのフックの追加](#)
- [クラスター内の RHEL コンピュータマシンの更新](#)

1.10. 非接続環境でのクラスターの更新

非接続環境でのクラスターの更新について: ミラーホストがインターネットとクラスターの両方にアクセスできない場合は、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できます。続いて、そのホストまたはリムーバブルメディアをそのギャップを越えて移動させることができます。ローカルコンテナーレジストリーとクラスターが、レジストリーのミラーホストに接続されている場合は、リリースイメージをローカルレジストリーに直接プッシュできます。

- [ミラーホストの準備](#)
- [イメージのミラーリングを可能にする認証情報の設定](#)
- [OpenShift Container Platform イメージリポジトリーのミラーリング](#)
- [切断されたクラスターの更新](#)
- [イメージレジストリーのリポジトリーミラーリングの設定](#)
- [クラスターノードの再起動の頻度を減らすために、ミラーイメージカタログの範囲を拡大](#)
- [OpenShift Update Service のインストール](#)
- [OpenShift Update Service アプリケーションの作成](#)
- [OpenShift Update Service アプリケーションの削除](#)
- [OpenShift Update Service Operator のアンインストール](#)

1.11. VSPHERE で稼働するノードでのハードウェアの更新

vSphere でのハードウェアの更新: vSphere で実行されているノードが OpenShift Container Platform でサポート対象のハードウェアバージョンで実行されていることを確認する必要があります。現時点で、ハードウェアバージョン 13 以降は、クラスター内の vSphere 仮想マシンでサポートされます。詳細については、次の情報を参照してください。

- [vSphere での仮想ハードウェアの更新](#)
- [vSphere での仮想ハードウェアの更新のスケジューリング](#)



重要

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

第2章 OPENSIFT CONTAINER PLATFORM の更新について

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、クラスターの更新が利用可能になると自動的に通知されます。

OpenShift Update Service (OSUS) は、レジストリー内のリリースイメージに基づいて更新の可能性のグラフを作成します。グラフは、推奨され、テストされた特定のバージョンからの更新パスに基づいています。OpenShift Container Platform クラスターは Red Hat Hybrid Cloud サーバーに接続し、バージョン情報とともに、ユーザーが実行しているクラスターを識別します。OSUS は、既知の更新ターゲットに関する情報で応答します。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、更新する新しいバージョンで編集します。CVO がレジストリーから更新イメージを受信した後、CVO は変更を適用します。



注記

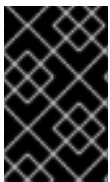
Operator Lifecycle Manager (OLM) を介して以前にインストールされた Operator は、異なる更新プロセスに従います。詳細は、[インストールされている Operator の更新](#) を参照してください。

2.1. OPENSIFT UPDATE SERVICE について

OpenShift Update Service (OSUS) は、Red Hat Enterprise Linux CoreOS (RHCOS) を含む OpenShift Container Platform に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。ユーザーが更新をリクエストすると、CVO はその更新のリリースイメージを使ってクラスターを更新します。リリースアーティファクトは、コンテナイメージとして Quay でホストされます。

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



重要

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

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



重要

サポートされているのは、新規バージョンへのアップグレードのみです。クラスターを以前のバージョンに戻すまたはロールバックすることはサポートされていません。更新が失敗した場合は、Red Hat サポートに連絡してください。

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

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

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

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

2.2. 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。

クラスターバージョン 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、Network Manager、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](#)
- [チャンネルとリリースの更新](#)

第3章 アップグレードチャンネルとリリースを理解

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

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

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

- **candidate-4.9**
- **fast-4.9**
- **stable-4.9**
- **eus-4.y** (4.8 などの偶数番号の 4.y クラスターリリースを実行している場合のみ)

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アップグレード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。

3.1. チャンネルおよびリリースパスのアップグレード

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

3.1.1. candidate-4.9 チャンネル

candidate-4.9 チャンネルには、z-stream (4.9.z) リリースの候補となるビルドが含まれます。リリース候補には、製品のすべての機能が含まれますが、それらがサポートされる訳ではありません。リリース候補を使用して機能の受け入れテストを実行し、OpenShift Container Platform の次のバージョンへの対応を支援します。リリース候補は、名前に **-rc** など、**プレリリースバージョン** を含まない、候補チャンネルで利用可能なビルドを指します。候補チャンネルでバージョンが利用可能になると、さらに品質のチェックが行われます。品質基準を満たす場合には、これは **fast-4.9** または **stable-4.9** チャンネルにプロモートされます。この戦略により、特定のリリースが **candidate-4.9** チャンネルと **fast-4.9** ま

または **stable-4.9** チャンネルの両方で利用可能な場合、そのリリースは Red Hat でサポートされるバージョンということになります。**candidate-4.9** チャンネルには、いずれのチャンネルでも推奨されている更新のないリリースバージョンを含めることができます。

candidate-4.9 チャンネルを使用して、OpenShift Container Platform の直前のマイナーバージョンから更新できます。

3.1.2. fast-4.9 チャンネル

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

fast-4.9 チャンネルを使用して、OpenShift Container Platform の以前のマイナーバージョンからの更新を実行できます。

3.1.3. stable-4.9 チャンネル

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

stable-4.9 チャンネルを使用して、OpenShift Container Platform の以前のマイナーバージョンからの更新を実行できます。

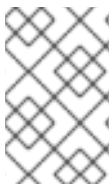
3.1.4. EUS-4.y チャンネル

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

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

次の EUS チャンネルに更新する場合は、次の EUS バージョンに到達するまで、次の EUS チャンネルに切り替えて更新することができます。

この更新プロセスは、**eus-4.6** チャンネルには適用されません。



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。

3.1.5. アップグレードバージョンパス

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

fast-4.9 チャンネルでは以下を確認できます。

- 4.9.0
- 4.9.1
- 4.9.3
- 4.9.4

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

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

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

3.1.6. 高速かつ安定したチャンネルの使用およびストラテジー

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

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

3.1.7. ネットワークが制限された環境のクラスター

OpenShift Container Platform クラスターのコンテナイメージを独自に管理する場合には、製品リリースに関連する Red Hat エラータを確認し、アップグレードへの影響に関するコメントに留意する必要があります。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。

3.1.8. CLI プロファイル間の切り替え

チャンネルは、Web コンソールまたは **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-4.9** チャンネルから **fast-4.9** 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-4.9** チャンネルに切り換えることはできますが、このチャンネルの一部のリリースはサポートされない可能性があります。
- 現在のリリースが一般利用公開リリースの場合、**candidate-4.9** チャンネルから **fast-4.9** チャンネルに切り換えることができます。
- **fast-4.9** チャンネルから **stable-4.9** チャンネルに常に切り換えることができます。現在のリリースが最近プロモートされた場合は、リリースが **stable-4.9** にプロモートされるまでに最長1日分の遅延が生じる可能性があります。

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

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

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

OpenShift Container Platform 4.8 クラスターを 4.9 に更新する前に、管理者の確認を提供する必要があります。

4.1. KUBERNETES API の削除

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

表4.1 v1beta1 API が Kubernetes 1.22 から削除

リソース	API	主な変更
APIService	apiregistration.k8s.io/v1beta1	いいえ
CertificateSigningRequest	certificates.k8s.io/v1beta1	はい
ClusterRole	rbac.authorization.k8s.io/v1beta1	いいえ
clusterRoleBinding	rbac.authorization.k8s.io/v1beta1	いいえ
CSIDriver	storage.k8s.io/v1beta1	いいえ
CSINode	storage.k8s.io/v1beta1	いいえ
CustomResourceDefinition	apiextensions.k8s.io/v1beta1	はい
Ingress	extensions/v1beta1	はい
Ingress	networking.k8s.io/v1beta1	はい
IngressClass	networking.k8s.io/v1beta1	いいえ
Lease	coordination.k8s.io/v1beta1	いいえ
LocalSubjectAccessReview	authorization.k8s.io/v1beta1	はい

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

4.2. 削除された API に関するクラスターの評価

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

4.2.1. 削除された API の使用を特定するためのアラートの確認

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

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

これらのアラートのいずれかがクラスターで実行している場合は、アラートを確認し、マニフェストおよび API クライアントを移行して新規 API バージョンを使用することによりアラートをクリアします。

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。

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

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできるようにする必要があります。

手順

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

```
$ oc get apirequestcounts
```

出力例

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

重要

結果に表示される以下のエントリ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ーすることもできます。

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

出力例

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

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

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできるようにする必要があります。

手順

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

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

以下に例を示します。

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

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

```
$ oc get apirequestcounts ingresses.v1beta1.networking.k8s.io \
  -o jsonpath='{range .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
watch bob                oc/v4.8.11
watch system:kube-controller-manager cluster-policy-controller/v0.0.0
```

4.3. 削除された API インスタンスのインスタンスの移行

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

4.4. 管理者の確認の提供

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



警告

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできるようにする必要があります。

手順

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

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

第5章 EUS から EUS への更新を実行するための準備

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 の更新をシリアル化する必要があります。OpenShift Container Platform 4.8 から 4.9 に更新してから、4.10 に更新する必要があります。OpenShift Container Platform 4.8 から 4.10 に直接更新することはできません。ただし、OpenShift Container Platform 4.8 から 4.9、そして 4.10 への更新以降、2つの Extended Update Support (EUS) バージョン間で更新を希望する管理者は、コントロールプレーン以外のホストを1回再起動するだけで更新できます。

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アップグレード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アップグレードする前)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。
- 中間ステップで一時停止することにより、複数のメンテナンスウィンドウ中に更新プロセスを完了することができます。ただし、更新全体を 60 日以内に完了するように計画してください。これは、証明書のローテーションに関連するプロセスを含め、通常のクラスター自動化プロセスを確実に完了するために重要です。
- EUS から EUS への更新手順を開始する前に、少なくとも OpenShift Container Platform 4.8.14 を実行している必要があります。この最小要件を満たしていない場合は、EUS から EUS への更新を試みる前に、以降の 4.8.z に更新してください。
- RHEL7 ワーカーのサポートは OpenShift Container Platform 4.10 で削除され、RHEL8 ワーカーに置き換えられたため、RHEL7 ワーカーを使用するクラスターでは EUS から EUS への更新は利用できません。
- ノードコンポーネントは OpenShift Container Platform 4.9 に更新されません。OpenShift Container Platform 4.10 への更新を完了し、すべての Machine Config Pools の更新を有効にするまで、OpenShift Container Platform 4.9 で修正されたすべての機能とバグが利用可能になることを期待しないでください。
- 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。

5.1. EUS から EUS への更新

次の手順では、マスター以外のすべての MachineConfigPools を一時停止し、OpenShift Container Platform 4.8 から 4.9、そして 4.10 への更新を実行してから、以前に一時停止した MachineConfigPools 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 4.9 および 4.10 のリリースノートを確認します
- 階層化された製品および OLM オペレーターのリリースノートおよび製品ライフサイクルを確認します。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 4.8 から 4.9 にアップグレードする前に必要な [管理者の確認](#) など、バージョン固有の前提条件に精通していることを確認してください。
- クラスターが OpenShift Container Platform バージョン 4.8.14 以降を実行していることを確認します。クラスターが OpenShift Container Platform 4.8.14 より前のバージョンを実行している場合は、4.9 に更新する前に、より新しい 4.8.z バージョンに更新する必要があります。MachineConfigPools を一時停止せずに実行する必要がある最小バージョン要件を満たすには、4.8.14 以降への更新が必要です。
- MachineConfigPools が一時停止されていないことを確認します。

手順

1. OLM オペレーターを、更新する両方のバージョンと互換性のあるバージョンにアップグレードします。
2. すべての MachineConfigPools が **UPDATED** のステータスを表示し、MachineConfigPools が **UPDATING** のステータスを表示しないことを確認します。すべての MachineConfigPools のステータスを表示するには、次のコマンドを実行します。

```
$ oc get mcp
```

出力例

わかりやすくするために、出力はトリミングされています。

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

3. 再起動をスキップする MachineConfigPools を一時停止するには、次のコマンドを実行します。



注記

マスタープールを一時停止することはできません。

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

4. **eus-4.10** チャンネルに変更し、次のコマンドを実行します。

```
$ oc adm upgrade channel eus-4.10
```

5. 4.9 に更新するには、次のコマンドを実行します。

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

出力例

```
Updating to latest version 4.9.18
```

6. 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 get clusterversion
```

出力例

```
NAME      VERSION AVAILABLE PROGRESSING SINCE STATUS
version  4.9.18  True     False     6m29s Cluster version is 4.9.18
```

- 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。
- 4.10 に更新するには、次のコマンドを実行します。

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

- 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 get clusterversion
```

出力例

```
NAME      VERSION AVAILABLE PROGRESSING SINCE STATUS
version  4.10.1  True     False     6m29s Cluster version is 4.10.1
```

- 以前に一時停止したすべての MachineConfigPools の一時停止を解除するには、次のコマンドを実行します。

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



注記

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への更新が許可されず、証明書のローテーションなどの保守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。

- 以前に一時停止したプールが更新され、クラスターが 4.10 への更新を完了したことを確認するには、次のコマンドを実行します。

```
$ oc get mcp
```

出力例

わかりやすくするために、出力はトリミングされています。

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

第6章 WEB コンソールを使用してクラスターを更新

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



注記

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。4.9 チャンネルに変更した後、[CLI を使用してクラスターを更新](#) 手順に従って、更新を完了することができます。

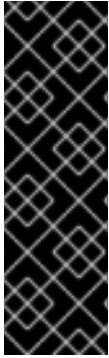
6.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできること。[RBAC の使用によるパーミッションの定義および適用](#) を参照してください。
- 更新が失敗し、[クラスターを直前の状態に復元する](#) 必要がある場合に最新の **etcd バックアップ** があること。

OpenShift Container Platform 4.9 では、etcd バージョン 3.4 から 3.5 に更新する必要があります。etcd Operator が更新を停止すると、アラートがトリガーされます。このアラートをクリアするには、次のコマンドで更新をキャンセルします。

```
$ oc adm upgrade --clear
```

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



重要

- 更新が完了しなかった場合、Cluster Version Operator (CVO) は、更新の調整を試みている間、ブロックしているコンポーネントのステータスを報告します。クラスターの以前のバージョンへのロールバックはサポートされていません。更新が完了しない場合は、Red Hat サポートに連絡してください。
- **unsupportedConfigOverrides** セクションを使用して Operator の設定を変更することはサポートされておらず、クラスターの更新をブロックする可能性があります。クラスターを更新する前に、この設定を削除する必要があります。

関連情報

- [管理外の Operator のサポートポリシー](#)

6.2. カナリアロールアウト更新の実行

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

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

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

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

6.3. 手動でメンテナンスされた認証情報を使用したクラスターのアップグレード

手動でメンテナンスされる認証情報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。

- 4.8 から 4.9 などのマイナーリリースの場合には、このステータスを使用することで、パーミッションを更新して **CloudCredential** リソースにアノテーションを付けてパーミッション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、アップグレードができませんようになります。このアノテーションは、**Upgradable** ステータスを **True** に変更します。
- 4.9.0 から 4.9.1 などの z-stream リリースの場合には、パーミッションは追加または変更されないため、アップグレードはブロックされません。

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認証情報を新規作成する必要があります。さらに、既存の認証情報に必要なパーミッションを確認し、これらのコンポーネントの新規リリースの新しいパーミッション要件に対応する必要があります。

手順

1. 新規リリースの **CredentialsRequest** カスタムリソースを抽出して検査します。
クラウドプロバイダーのインストールコンテンツの IAM の手動作成についてのセクションでは、クラウドに必要な認証情報を取得し、使用方法について説明します。
2. クラスターで手動でメンテナンスされる認証情報を更新します。
 - 新規リリースイメージによって追加される **CredentialsRequest** カスタムリソースの新規のシークレットを作成します。
 - 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。
3. 新規リリースですべてのシークレットが正しい場合は、クラスターをアップグレードする準備が整っていることを示します。
 - a. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。
 - b. **CloudCredential** リソースを編集して、**metadata** フィールドに **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>
  ...
```

ここで、**<version_number>** は、アップグレードするバージョン (**x.y.z** 形式) に置き換えます。例: OpenShift Container Platform **4.8.2** (OpenShift Container Platform 4.8.2 の場合)

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。

4. CCO がアップグレードできることを確認します。
 - a. 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。
 - b. CCO ステータスの詳細を表示するには、**Cluster Operators** 一覧で **cloud-credential** をクリックします。
 - c. **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。

Conditions セクションの **Upgradeable** ステータスが **True** の場合には、OpenShift Container Platform のアップグレードを開始できます。

関連情報

- [AWS の IAM の手動作成](#)
- [Azure の IAM の手動作成](#)
- [GCP の IAM の手動作成](#)

6.4. WEB コンソールを使用した MACHINEHEALTHCHECK リソースの一時停止


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

前提条件

- **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。

6.5. 単一ノードの OPENSIFT CONTAINER PLATFORM の更新

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

ただし、以下の制限事項に注意してください。

- 他にヘルスチェックを実行するノードがないので、**MachineHealthCheck** リソースを一時停止する時に課される前提条件は必要ありません。
- etcd バックアップを使用した単一ノードの OpenShift Container Platform クラスターの復元は、正式にはサポートされていません。ただし、アップグレードに失敗した場合には、etcd バックアップを実行することが推奨されます。コントロールプレーンが正常である場合には、バックアップを使用してクラスターを以前の状態に復元できる場合があります。
- 単一ノードの OpenShift Container Platform クラスターを更新するには、ダウンタイムが必要です。更新には、自動再起動も含まれる可能性があります。ダウンタイムの時間は、以下のシナリオのように更新ペイロードによって異なります。
 - 更新ペイロードに再起動が必要なオペレーティングシステムの更新が含まれる場合には、ダウンタイムは、クラスター管理およびユーザーのワークロードに大きく影響します。
 - 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

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

6.6. WEB コンソールを使用したクラスターの更新

更新が利用可能な場合、Web コンソールからクラスターを更新できます。

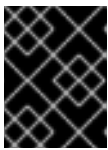
利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータル の [エラータ](#) のセクションを参照してください。

前提条件

- admin** 権限を持つユーザーとして Web コンソールにアクセスできる。
- すべての **MachineHealthCheck** リソースを一時停止します。

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。
- 本番クラスターの場合は、**チャンネル** が、**stable-4.9** など、更新するバージョンの正しいチャンネルに設定されていることを確認します。



重要

実稼働クラスターの場合は、**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。

- Update Status** が **Updates Available** ではない場合、クラスターをアップグレードすることはできません。
 - Select Channel** は、クラスターが実行されているか、または更新されるクラスターのバージョンを示します。
- 更新するバージョンを選択し、**Save** をクリックします。
入力チャンネルの **Update Status** が **Update to <product-version> in progress** 切り替わり、Operator およびノードの進捗バーを監視して、クラスター更新の進捗を確認できます。



注記

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

- 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。

6.7. WEB コンソールを使用した更新サーバーの変更

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

手順

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

出力例

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

第7章 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 クラスターをマイナーバージョン内で更新するか、またはアップグレードすることができます。同じ手順に従って、マイナーバージョン間でクラスターを更新することもできます。

7.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできること。[RBAC の使用によるパーミッションの定義および適用](#) を参照してください。
- 更新が失敗し、[クラスターを直前の状態に復元する](#) 必要がある場合に最新の **etcd** バックアップがあること。
- Operator Lifecycle Manager (OLM) で以前にインストールされたすべての Operator が、最新チャンネルの最新バージョンに更新されていることを確認します。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの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。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。
- クラスターで手動で保守される認証情報を使用する場合は、Cloud Credential Operator (CCO) がアップグレード可能な状態であることを確認します。[AWS](#)、[Azure](#)、または [GCP](#) の [手動で保守される認証情報を使用したクラスターのアップグレード](#) を参照してください。
- クラスターが、AWS Secure Token Service (STS) で手動でメンテナンスされる認証情報を使用する場合は、更新するリリースイメージから **ccoctl** ユーティリティのコピーを取得し、これを使用して更新された認証情報を処理します。詳細は、[Upgrading an OpenShift Container Platform cluster configured for manual mode with STS](#) を参照してください。
- クラスターで次のマイナーバージョンへの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。**oc adm upgrade** コマンドを実行して、すべての **Upgradeable=False** 条件の出力と、マイナーバージョン更新の準備に役立つ条件推論を実行できます。
- 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、アップグレード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 サポートに連絡してください。
- **unsupportedConfigOverrides** セクションを使用して Operator の設定を変更することはサポートされておらず、クラスターの更新をブロックする可能性があります。クラスターを更新する前に、この設定を削除する必要があります。

関連情報

- [管理外の Operator のサポートポリシー](#)

7.2. MACHINEHEALTHCHECK リソースの一時停止

アップグレードプロセスで、クラスタ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、クラスタを更新する前にすべての **MachineHealthCheck** リソースを一時停止します。

前提条件

- OpenShift CLI (**oc**) をインストールしている。

手順

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

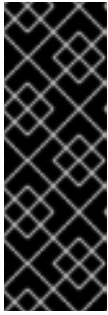
```
$ oc get machinehealthcheck -n openshift-machine-api
```

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

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

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

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



重要

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

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

7.3. 単一ノードの OPENSIFT CONTAINER PLATFORM の更新

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

ただし、以下の制限事項に注意してください。

- 他にヘルスチェックを実行するノードがないので、**MachineHealthCheck** リソースを一時停止する時に課される前提条件は必要ありません。
- `etcd` バックアップを使用した単一ノードの OpenShift Container Platform クラスターの復元は、正式にはサポートされていません。ただし、アップグレードに失敗した場合には、`etcd` バックアップを実行することが推奨されます。コントロールプレーンが正常である場合には、バックアップを使用してクラスターを以前の状態に復元できる場合があります。
- 単一ノードの OpenShift Container Platform クラスターを更新するには、ダウンタイムが必要です。更新には、自動再起動も含まれる可能性があります。ダウンタイムの時間は、以下のシナリオのように更新ペイロードによって異なります。
 - 更新ペイロードに再起動が必要なオペレーティングシステムの更新が含まれる場合には、ダウンタイムは、クラスター管理およびユーザーのワークロードに大きく影響します。
 - 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

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

7.4. CLI を使用したクラスターの更新

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

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

前提条件

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

手順

1. クラスタが利用可能であることを確認します。

```
$ oc get clusterversion
```

出力例

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

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

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

出力例

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



重要

実稼働クラスタの場合は、**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。

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

```
$ oc adm upgrade
```

出力例

```
Cluster version is 4.8.13
```

```
Updates:
```

VERSION IMAGE

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

4. 更新を適用します。

- 最新バージョンに更新するには、以下を実行します。

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

- 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> ❶
```

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

5. クラスターバージョン Operator を確認します。

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

出力例

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

    "version": "4.9.0" ❶
  }
}
```

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

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

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

出力例

```
[
  {
    "completionTime": null,
    "image": "quay.io/openshift-release-dev/ocp-
release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b
7",
    "startedTime": "2021-01-28T20:30:50Z",
```

```

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

```

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

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

```
$ oc get clusterversion
```

出力例

```

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

```



注記

PROGRESSING ステータスが **True** のときに **oc get clusterversion** コマンドで次のエラーが表示された場合は、エラーを無視できます。

```

NAME      VERSION  AVAILABLE  PROGRESSING  SINCE   STATUS
version  4.10.26 True       True         24m     Unable to apply 4.11.0-rc.7: an
unknown error has occurred: MultipleErrors

```

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

```
$ oc get nodes
```

出力例

```

NAME                                     STATUS  ROLES  AGE  VERSION
ip-10-0-168-251.ec2.internal             Ready  master  82m  v1.22.1
ip-10-0-170-223.ec2.internal             Ready  master  82m  v1.22.1
ip-10-0-179-95.ec2.internal               Ready  worker  70m  v1.22.1

```

```
ip-10-0-182-134.ec2.internal Ready worker 70m v1.22.1
ip-10-0-211-16.ec2.internal Ready master 82m v1.22.1
ip-10-0-250-100.ec2.internal Ready worker 69m v1.22.1
```

7.5. CLI を使用した更新サーバーの変更

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

手順

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

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

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

出力例

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


第8章 カナリアロールアウト更新の実行

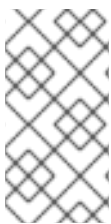
更新プロセスによってアプリケーションが失敗した場合でも、更新全体を通じてミッションクリティカルなアプリケーションを利用できるようにするために、ワーカーノードへの更新のより制御されたロールアウトが必要なシナリオがいくつかある場合があります。組織のニーズによっては、ワーカーノードの小規模なサブセットを更新し、一定期間でクラスターおよびワークロードの正常性を評価し、残りのノードを更新する必要が生じる場合があります。通常、これは **カナリア更新** と呼ばれます。または、クラスター全体を一度に更新するために大きなメンテナンスウィンドウを使用できない場合は、ホストの再起動が必要になることが多いワーカー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

これらのシナリオでは、複数のカスタムマシン設定プール (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。

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

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

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



注記

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



重要

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

8.1. カナリアロールアウト更新プロセスおよび MCP について

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

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

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



注記

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

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

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

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

8.2. カナリアロールアウト更新の実行について

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

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



注記

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

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



注記

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

3. 更新プロセスの一部として更新しない MCP を一時停止します。



注記

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

4. クラスターの更新を実行します。更新プロセスでは、コントロールプレーンノードを含む、一時停止されない MCP を更新します。
5. 更新されたノードでアプリケーションをテストし、想定通りに機能していることを確認します。
6. 残りの MCP を1つずつ一時停止解除し、すべてのワーカーノードが更新されるまでそれらのノードでアプリケーションをテストします。MCP の一時停止を解除すると、その MCP に関連付けられたノードの更新プロセスが開始されます。**Administration** → **Cluster settings** をクリックして、Web コンソールから更新の進捗を確認できます。または、**oc get machineconfigpools** CLI コマンドを使用します。
7. 必要に応じて、更新されたノードからカスタムラベルを削除し、カスタム MCP を削除します。

8.3. カナリアロールアウト更新を実行するためのマシン設定プールの作成

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

1. ワーカーノードから MCP を作成します。
 - a. クラスターのワーカーノードを一覧表示します。

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

出力例

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

- b. 遅延させるノードの場合は、カスタムラベルをノードに追加します。

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

以下に例を示します。

```
$ 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-spl7-worker-a-xk76k labeled
```

- c. 新規 MCP を作成します。

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

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

出力例

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

d. クラスター内の MCP およびそれらの現在の状態を表示します。

```
$ oc get machineconfigpool
```

出力例

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

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

8.4. マシン設定プールの一時停止

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

注記

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

MCP を一時停止するには、以下を実行します。

1. 一時停止する MCP にパッチを適用します。

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

以下に例を示します。

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

出力例

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

■

8.5. クラスターの更新の実行

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

- [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](#)

更新が完了したら、MCP の 1 回の一時停止を解除することができます。

8.6. マシン設定プールの一時的停止の解除

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

MCP の一時停止を解除するには、以下を実行します。

1. 一時停止を解除する MCP にパッチを適用します。

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

以下に例を示します。

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

出力例

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

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

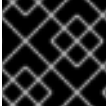
2. 更新されたノードでアプリケーションをテストし、想定通りに機能していることを確認します。
3. 一時停止した他の MCP の一時停止を解除すると、1 回目でアプリケーションが機能することを確認します。

8.6.1. アプリケーション障害発生時

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

8.7. ノードを元のマシン設定プールに移行

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



重要

ノードには、クラスター内で適切に機能するロールが必要です。

ノードを元の MCP に移動するには、以下を実行します。

1. ノードからカスタムラベルを削除します。

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

以下に例を示します。

```
$ 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
```

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

2. クラスター内の MCP およびそれらの現在の状態を表示します。

```
$ oc get mcp
```

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

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

3. 必要に応じて、カスタム MCP を削除します。

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

第9章 RHEL コンピュータマシンを含むクラスターの更新

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

9.1. 前提条件

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

9.2. WEB コンソールを使用したクラスターの更新

更新が利用可能な場合、Web コンソールからクラスターを更新できます。

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

前提条件

- **admin** 権限を持つユーザーとして Web コンソールにアクセスできる。
- すべての `MachineHealthCheck` リソースを一時停止します。

手順

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



重要

実稼働クラスタの場合は、**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。

- **Update Status** が **Updates Available** ではない場合、クラスタをアップグレードすることはできません。
 - **Select Channel** は、クラスタが実行されているか、または更新されるクラスタのバージョンを示します。
3. 更新するバージョンを選択し、**Save** をクリックします。
入力チャンネルの **Update Status** が **Update to <product-version> in progress** 切り替わり、Operator およびノードの進捗バーを監視して、クラスタ更新の進捗を確認できます。



注記

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

4. 更新が完了し、Cluster Version Operator が利用可能な更新を更新したら、追加の更新が現在のチャ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。



注記

Red Hat Enterprise Linux (RHEL) ワーカーマシンを含むクラスタを更新する場合、それらのワーカーは、更新プロセス時に一時的に使用できなくなります。クラスタの更新の終了において各 RHEL マシンの状態が **NotReady** になる際に、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。

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

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

9.3.1. アップグレード用の Ansible Hook について

OpenShift Container Platform の更新時に **フック** を使用し、特定操作の実行中に Red Hat Enterprise Linux (RHEL) ノードでカスタムタスクを実行できます。フックを使用して、特定の更新タスクの前後に実行するタスクを定義するファイルを指定できます。OpenShift Container Platform クラスタで

RHEL コンピュートノードを更新する際に、フックを使用してカスタムインフラストラクチャーを検証したり、変更したりすることができます。

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

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

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

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

前提条件

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

手順

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

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

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

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

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

インベントリーファイルのフック定義の例

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

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

9.3.3. 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。
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。

9.4. 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

Red Hat Enterprise Linux (RHEL) バージョン 8.4 およびバージョン 8.5 は、RHEL ワーカー (コンピューティング) マシンで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 コンピュータマシンを RHEL 8 にアップグレードすることはできません。新しい RHEL 8 ホストをデプロイする必要があり、古い RHEL 7 ホストを削除する必要があります。

前提条件

- クラスターが更新されていること。



重要

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

- RHEL コンピュー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` スパッドが有効になります。ホストで `firewalld` サービスを有効にすると、ワーカーで OpenShift Container Platform ログにアクセスできなくなります。ワーカーの OpenShift Container Platform ログへのアクセスを継続する場合は、`firewalld` を後で有効にしないでください。

2. OpenShift Container Platform 4.9 で必要なリポジトリを有効にします。
 - 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=rhel-7-server-ose-4.8-rpms \
--enable=rhel-7-server-ansible-2.9-rpms \
--enable=rhel-7-server-ose-4.9-rpms
```

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

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

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

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

3. RHEL ワーカーマシンを更新します。

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

```
# oc get node
```

出力例

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

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

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

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

openshift_kubeconfig_path=~/.kube/config"

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

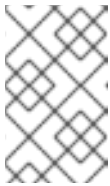
- c. **openshift-ansible** ディレクトリーに移動します。

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

- d. **upgrade** Playbook を実行します。

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

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



注記

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

- 直前の手順で実行したプロセスに従って、クラスター内の各 RHEL ワーカーマシンを更新します。
- すべてのワーカーを更新したら、すべてのクラスターノードが新規バージョンに更新されていることを確認します。

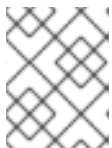
```
# oc get node
```

出力例

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

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

```
# yum update
```



注記

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

第10章 非接続環境でのクラスタの更新

10.1. 非接続環境でのクラスタの更新について

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

切断されたネットワーク内の複数のクラスタ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。

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

非接続環境でクラスタを更新するには、クラスタ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 イメージリポジトリのミラーリング](#)

10.1.2. 非接続環境でのクラスタの更新の実行

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

10.1.3. クラスタからの 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からアンインストールできます。

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

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

切断された環境でクラスタを更新する前に、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で実行するようにすることもできます。

10.2.1. 前提条件

- Red Hat Quay など、OpenShift Container Platform クラスタ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。

10.2.2. 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。

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

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



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.9 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。切断された環境でクラスターをアップグレードする場合は、アップグレード先の **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. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.9 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開します。

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

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

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

4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATHを確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

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

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

macOS への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルでの [OpenShift Container Platform ダウンロードページ](#) に移動します。
2. **Version** ドロップダウンメニューで適切なバージョンを選択します。
3. **OpenShift v4.9 MacOSX Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATHを確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

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

```
$ oc <command>
```

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



警告

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



警告

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

前提条件

- 切断された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の場所を特定している。
- イメージのイメージリポジトリへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. registry.redhat.io プルシークレットを [Red Hat OpenShift Cluster Manager](#) からダウンロードし、`.json` ファイルに保存します。
2. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

```
$ echo -n '<user_name>:<password>' | base64 -w0 ❶  
BGVtbYk3ZHAtdXs=
```

- 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。

3. JSON 形式でプルシークレットのコピーを作成します。

```
$ cat ./pull-secret.text | jq . > <path>/<pull_secret_file_in_json> 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。

4. 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。
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": "b3B1bnNo...",  
      "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"
}
}
}
}

```

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

```

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

```

- 1 **<mirror_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例:
registry.example.com または **registry.example.com:8443**
- 2 **<credentials>** については、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

ファイルは以下の例のようになります。

```

{
  "auths": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHA tqXs=",
      "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"
    }
  }
}

```

```

}
}
}

```

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



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にミラーリングする必要があります。

前提条件

- 切断された環境で使用するミラーレジストリーを設定し、設定した証明書と認証情報にアクセスできるようになりました。
- [Red Hat OpenShift Cluster Manager](#) から [プルシークレット](#) をダウンロードし、ミラーリポジトリへの認証を含めるようにこれを変更している。
- 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ドメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。

- c. ローカルリポジトリ名をエクスポートします。

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

<local_repository_name> については、**ocp4/openshift4** などのレジストリーに作成するリポジトリの名前を指定します。

- d. 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の名前を指定します。

- e. ミラーリングするリポジトリの名前をエクスポートします。

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

実稼働環境のリリースの場合には、**openshift-release-dev** を指定する必要があります。

- f. パスをレジストリープルシークレットにエクスポートします。

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

<path_to_pull_secret> については、作成したミラーレジストリーのプルシークレットの絶対パスおよびファイル名を指定します。



注記

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

- g. リリースミラーをエクスポートします。

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

実稼働環境のリリースについては、**ocp-release** を指定する必要があります。

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

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

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

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

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

3. 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。

- 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。
- 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}
```

- メディアを切断された環境に移動し、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。

- oc** コマンドラインインターフェイス (CLI) を使用して、アップグレードしているクラスターにログインします。
- 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**)。

- 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ーにミラーリングします。

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

- ローカルコンテナーレジストリーとクラスターがミラーホストに接続されている場合は、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。

- ii. 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}:${OCP_REL
EASE}-${ARCHITECTURE}
```

10.3. OPENSIFT UPDATE SERVICE を使用した非接続環境でのクラスタの更新

接続されたクラスタ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 をインストールおよび設定できます。

10.3.1. 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 をローカルに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。

次のセクションでは、ローカル 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](#)
- [アップグレードチャンネルとリリースを理解](#)

10.3.2. 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 イメージリポジトリのミラーリング](#) 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。

10.3.3. 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って署名されている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](#) の手順と、更新サービスに以下の変更を加えます。

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

更新サービス向けのイメージレジストリー CA の設定マップの例

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

- 1** OpenShift Update Service Operator では、設定マップのキー名 **updateservice-registry** がレジストリー CA 証明書に必要です。
- 2** レジストリーにポートがある場合 (例: **registry-with-port.example.com:5000**)、: は .. に置き換える必要があります。

10.3.4. 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> 1
```


- 1 プルシークレットファイルへのパスを指定します。

b. 以下のコマンドを実行して、新しいプルシークレットを追加します。

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

- 1 新しいレジストリーを指定します。同じレジストリー内に複数のリポジトリを含めることができます (例: `--registry="1<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=1<pull_secret_location>
```

- 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。

10.3.5. 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

切断された環境 (非接続クラスタとして知られる) にインストールされているクラスタの場合には、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。

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

10.3.5.2. 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" ①
```

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

- b. namespace を作成します。

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

以下に例を示します。

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

2. 以下のオブジェクトを作成して OpenShift Update Service Operator をインストールします。

- a. **OperatorGroup** オブジェクト YAML ファイルを作成します (例: **update-service-operator-group.yaml**)。

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

- b. **OperatorGroup** オブジェクトを作成します。

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

以下に例を示します。

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

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

Subscription の例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: update-service-subscription
spec:
  channel: v1
  installPlanApproval: "Automatic"
  source: "redhat-operators" 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の場合には、**redhat-operators** を指定します。OpenShift Container Platform クラスタが切断された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。

- d. **Subscription** オブジェクトを作成します。

■

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

以下に例を示します。

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

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

3. Operator のインストールを確認します。

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

出力例

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

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

関連情報

- [namespace への Operator のインストール](#)

10.3.6. 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、グラフデータはアップグレードグラフデータリポジトリから直接取得します。インターネット接続が利用できない場合には、グラフデータを OpenShift Update Service で利用できるようにする別の方法として init コンテナからこの情報を読み込むことができます。init コンテナのロールとして、グラフデータのローカルコピーを提供し、Pod の初期化時に init コンテナはデータをサービスがアクセスできるボリュームにコピーすることが挙げられます。

手順

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

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

RUN curl -L -o cincinnati-graph-data.tar.gz
https://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"]
```

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

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

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

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



注記

切断された環境でグラフデータイメージをローカル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。

10.3.7. 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。

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

手順

1. Web コンソールで **Operators** → **Installed Operators** をクリックします。
2. インストールされた Operator の一覧から **OpenShift Update Service** を選択します。
3. **Update Service** タブをクリックします。
4. **Create UpdateService** をクリックします。
5. **service** など、**Name** フィールドに名前を入力します。
6. **Graph Data Image** フィールドに OpenShift Update Service グラフデータコンテナイメージの作成で作成した graph-data コンテナイメージにローカルの pullspec を入力します (例: **registry.example.com/openshift/graph-data:latest**)。
7. **Releases** フィールドに、OpenShift Container Platform イメージリポジトリのミラーリングでリリースイメージを含むように作成したローカルのレジストリーとリポジトリ (例: **registry.example.com/ocp4/openshift4-release-images**) を入力します。

8. **Replicas** フィールドに **2** と入力します。
9. **Create** をクリックして OpenShift Update Service アプリケーションを作成します。
10. OpenShift Update Service アプリケーションを検証します。
 - **Update Service** タブの **UpdateServices** 一覧から、作成した Update Service アプリケーションをクリックします。
 - **Resources** タブをクリックします。
 - 各アプリケーションリソースのステータスが **Created** であることを確認します。

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

手順

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

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

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

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

```
$ NAME=service
```

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

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

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

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

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

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

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

a. 以下のコマンドを使用してポリシーエンジンルートを取得します。

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

コマンドが成功するまでポーリングが必要になる場合があります。

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

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

これにより、グラフ要求が成功するまでポーリングされます。ただし、ミラーリングしたり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。

10.3.7.2.1. Cluster Version Operator (CVO) の設定

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

前提条件

- OpenShift Update Service Operator がインストールされている。
- OpenShift Update Service の graph-data コンテナイメージを作成して、OpenShift Update Service がアクセスできるリポジトリにプッシュしておく。

- 現在のリリースおよび更新ターゲットリリースがローカルアクセス可能なレジストリーにミラーリングされている。
- OpenShift Update Service アプリケーションが作成されている。

手順

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

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

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

```
$ NAME=service
```

3. ポリシーエンジンルートを取得します。

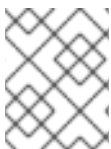
```
$ POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice "${NAME}")"
```

4. プルグラフデータのパッチを設定します。

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

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

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



注記

クラスター全体のプロキシーを有効にして、更新サーバーを信頼するように CA を設定するを参照してください。

10.3.8. 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、ローカルに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。
- 現在のリリースと更新ターゲットリリースのイメージが、ローカルでアクセス可能なレジストリーにミラーリングされている。
- 最近のグラフデータコンテナイメージがローカルレジストリーにミラーリングされている。

ローカルに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の更新](#)
- [EUS から EUS への更新を実行するための準備](#)
- [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の更新](#)

10.4. OPENSIFT UPDATE SERVICE を使用しない非接続環境でのクラスタの更新

以下の手順を使用して、OpenShift Update Service にアクセスせずに切断された環境でクラスタを更新します。

10.4.1. 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 イメージリポジトリのミラーリング](#) 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にアクセスできる。[RBAC の使用によるパーミションの定義および適用](#) を参照してください。
- 更新が失敗し、[クラスタを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** がある。
- 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 を一時停止することができます。
- クラスタで手動で保守される認証情報を使用する場合は、Cloud Credential Operator (CCO) がアップグレード可能な状態であることを確認する。[AWS](#)、[Azure](#)、または [GCP](#) の [手動で保守される認証情報を使用したクラスタのアップグレード](#) を参照してください。
- 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、アップグレード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。

10.4.2. MachineHealthCheck リソースの一時停止

アップグレードプロセスで、クラスタ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、クラスタを更新する前にすべての **MachineHealthCheck** リソースを一時停止します。

前提条件

- OpenShift CLI (**oc**) をインストールしている。

手順

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

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

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

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

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

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

重要

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

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

10.4.3. 切断されたクラスターのアップグレード

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。
- イメージ署名 ConfigMap からリリースの sha256 合計値を取得している。
- OpenShift CLI (**oc**) をインストールしている。
- すべての **MachineHealthCheck** リソースを一時停止します。

手順

- クラスターを更新します。

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

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

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



注記

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

10.4.4. 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

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

- ソースイメージのレジストリーのリポジトリ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で解決できるようにします。
- 各ターゲットリポジトリに対して複数のミラーリングされたりポジトリを特定し、1つのミラーがダウンした場合に別のミラーを使用できるようにします。

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

- イメージプルには、レジストリーのダウンタイムに対する回復性があります。
- 切断された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背後に配置することで、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 インストール時にミラーリングを設定しなくても、**ImageContentSourcePolicy** オブジェクトを使用して後で設定することができます。

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

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



注記

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

前提条件

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

```
$ skopeo copy \
```

```
docker://registry.access.redhat.com/ubi8/ubi-
minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eba
a6 \
docker://example.io/example/ubi-minimal
```

この例では、**example.io** という名前のコンテナイメージレジストリーと **example** という名前のイメージリポジトリーがあり、そこに **registry.access.redhat.com** から **ubi8/ubi-minimal** イメージをコピーします。レジストリーを作成した後、OpenShift Container Platform クラスターを設定して、ソースリポジトリーで作成される要求をミラーリングされたりリポジトリーにリダイレクトできます。

2. OpenShift Container Platform クラスターにログインします。
3. **ImageContentSourcePolicy** ファイル (例: **registryrepomirror.yaml**) を作成し、ソースとミラーを固有のレジストリー、およびリポジトリーのペアとイメージのものに置き換えます。

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: ubi8repo
spec:
  repositoryDigestMirrors:
  - mirrors:
    - example.io/example/ubi-minimal ①
    - example.com/example/ubi-minimal ②
    source: registry.access.redhat.com/ubi8/ubi-minimal ③
  - mirrors:
    - mirror.example.com/redhat
    source: registry.redhat.io/openshift4 ④
  - mirrors:
    - mirror.example.com
    source: registry.redhat.io ⑤
  - mirrors:
    - mirror.example.net/image
    source: registry.example.com/example/myimage ⑥
  - mirrors:
    - mirror.example.net
    source: registry.example.com/example ⑦
  - mirrors:
    - mirror.example.net/registry-example-com
    source: registry.example.com ⑧
```

- ① イメージレジストリーおよびリポジトリーの名前を示します。
- ② 各ターゲットリポジトリーの複数の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す。
- ③ ミラーリングされているコンテンツが含まれるレジストリーおよびリポジトリーを示します。
- ④ レジストリー内の namespace を、その namespace の任意のイメージを使用するように設定できます。レジストリードメインをソースとして使用する場合は、**ImageContentSourcePolicy** リソースはレジストリーからすべてのリポジトリーに適用されます。

- 5 レジストリー名を設定すると、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 **ImageContentSourcePolicy** リソースが適用されます。
- 6 イメージ **mirror.example.net/image@sha256:...** をプルします。
- 7 ミラー **mirror.example.net/myimage@sha256:...** からソースレジストリー namespace のイメージ **myimage** をプルします。
- 8 ミラーレジストリー **mirror.example.net/registry-example-com/example/myimage@sha256:...** からイメージ **registry.example.com/example/myimage** をプルします。 **ImageContentSourcePolicy** リソースは、ソースレジストリーからミラーレジストリー **mirror.example.net/registry-example-com** までのすべてのリポジトリーに適用されます。

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

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

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

5. ミラーリングされた設定が適用されていることを確認するには、ノードのいずれかで以下を実行します。
 - a. ノードの一覧を表示します。

```
$ oc get node
```

出力例

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-137-44.ec2.internal	Ready	worker	7m	v1.24.0
ip-10-0-138-148.ec2.internal	Ready	master	11m	v1.24.0
ip-10-0-139-122.ec2.internal	Ready	master	11m	v1.24.0
ip-10-0-147-35.ec2.internal	Ready	worker	7m	v1.24.0
ip-10-0-153-12.ec2.internal	Ready	worker	7m	v1.24.0
ip-10-0-154-10.ec2.internal	Ready	master	11m	v1.24.0

Imagecontentsourcepolicy リソースはノードを再起動しません。

- 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
```

出力例

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

```
[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi8/ubi-minimal"
  mirror-by-digest-only = true
```

```
[[registry.mirror]]
  location = "example.io/example/ubi-minimal"
```

```
[[registry.mirror]]
  location = "example.com/example/ubi-minimal"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com"
  mirror-by-digest-only = true
```

```
[[registry.mirror]]
  location = "mirror.example.net/registry-example-com"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com/example"
  mirror-by-digest-only = true
```

```
[[registry.mirror]]
  location = "mirror.example.net"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com/example/myimage"
  mirror-by-digest-only = true
```

```
[[registry.mirror]]
  location = "mirror.example.net/image"
```

```
[[registry]]
  prefix = ""
  location = "registry.redhat.io"
  mirror-by-digest-only = true
```

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

```
[[registry]]
  prefix = ""
```

```
location = "registry.redhat.io/openshift4"
mirror-by-digest-only = true
```

```
[[registry.mirror]]
location = "mirror.example.com/redhat"
```

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

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eba
a6
```

リポジトリのミラーリングのトラブルシューティング

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

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

10.4.5. 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ップを設定できます。幅広いスコップの **ImageContentSourcePolicy** リソースにより、リソースの変更に対応するためにノードが再起動する必要がある回数が減ります。

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

前提条件

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

手順

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

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

ここでは、以下のようになります。

<local_registry>

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

<pull_spec>

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

<pull_secret_file>

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

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

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

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

検証

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

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

出力例

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

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

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

10.4.6. 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](#)

10.5. クラスタからの OPENSIFT 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。

10.5.1. 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。

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

10.5.1.2. 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
```

10.5.2. 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。

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

前提条件

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

手順

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

10.5.2.2. 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
```

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

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

出力例

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

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

```
$ oc get subscription
```

出力例

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

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

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

出力例

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

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

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

出力例

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

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

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

出力例

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

第11章 VSPHERE で稼働するノードでのハードウェアの更新

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

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



重要

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

11.1. 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。

11.1.1. 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 6.7U3 以降を使用している。

手順

1. クラスターのコントロールプレーンノードを一覧表示します。

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

出力例

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

コントロールプレーンノードの名前を書き留めておきます。

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

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

3. コントロールプレーンノードに関連付けられた仮想マシンをシャットダウンします。仮想マシンを右クリックし、**Power** → **Shut Down Guest OS**を選択して、vSphere クライアントでこれを実行します。安全にシャットダウンされない場合があるため、**Power Off** を使用して仮想マシンをシャットダウンしないでください。
4. vSphere クライアントで VM を更新します。詳細は、VMware ドキュメントの [Upgrading a virtual machine to the latest hardware version](#) を参照してください。
5. コントロールプレーンノードに関連付けられた仮想マシンの電源を入れます。仮想マシンを右クリックし、**Power On** を選択して、vSphere クライアントでこれを実行します。
6. ノードが **Ready** として報告されるまで待機します。

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

7. コントロールプレーンノードを再度スケジュール対象としてマークします。

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

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

11.1.2. vSphere でのコンピュータ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、コンピュータ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、複数のコンピュータノードを並行して更新できます。管理者が責任を持って、必要なコンピュータノードを利用できる状態にしてください。

前提条件

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

手順

1. クラスターのコンピュータノードを一覧表示します。

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

出力例

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

コンピュータ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。詳細は、VMware ドキュメントの [Upgrading a virtual machine to the latest hardware version](#) を参照してください。
6. コンピュートノードに関連付けられた仮想マシンの電源を入れます。仮想マシンを右クリックし、**Power On** を選択して、vSphere クライアントでこれを実行します。
7. ノードが **Ready** として報告されるまで待機します。

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

8. コンピュートノードを再度スケジュール対象としてマークします。

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

9. クラスタ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。

11.1.3. 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管理者パーミッションがある。
- vSphere ESXi ホストがバージョン 6.7U3 以降を使用している。

手順

1. 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。

1. vSphere クライアントで VM を更新します。詳細は、VMware ドキュメントの [Upgrading a virtual machine to the latest hardware version](#) を参照してください。

2. 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](#)

11.2. 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リング](#) に従い、仮想ハードウェアの更新だけを vCenter でスケジュールできます。

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