OpenShift Container Platform 4.12

クラスターの更新

OpenShift Container Platform クラスターの更新

Last Updated: 2024-06-29

OpenShift Container Platform 4.12 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

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

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. OPENSHIFT 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. 更新チャネルとリリースについて

更新チャネルとリリース: 更新チャネルを使用すると、更新戦略を選択できます。更新チャネルは OpenShift Container Platform のマイナーバージョン固有のものです。更新チャネルはリリースの選択のみを制御し、インストールするクラスターのバージョンには影響しません。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
- 条件更新について

1.3. クラスター OPERATOR の状態タイプについて

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

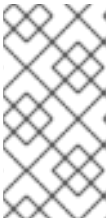
Cluster Version Operator (CVO) は、クラスター管理者が OpenShift Container Platform クラスターのステータス状態をよりよく理解できるように、クラスター Operator からステータス状態を収集します。

- Available: 条件タイプ **Available** は、Operator が機能しており、クラスターで使用可能であることを示します。ステータスが **False** の場合、オペランドの少なくとも 1 つの部分が機能していないため、管理者が介入する必要があります。

- Progressing: 条件タイプ **Progressing** は、Operator がアクティブに新しいコードをロールアウトしている、設定の変更を伝達している、またはある安定状態から別の安定状態に移行していることを示します。
  Operator が以前の既知の状態を調整している場合は、状態タイプ **Progressing** は **True** として報告されません。監視されたクラスターの状態が変化し、Operator がそれに反応している場合は、ある安定状態から別の安定状態に移行しているため、ステータスは **True** として報告されます。

- Degraded: 状態タイプ **Degraded** は、Operator の現在の状態が一定期間にわたって必要な状態に一致しないことを示します。期間はコンポーネントによって異なる場合があります が、**Degraded** ステータスは、Operator の状態が継続的に監視されていることを表します。そのため、Operator の **Degraded** 状態が変動することはありません。

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



注記

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

- Upgradeable: 条件タイプ **Upgradeable** は、Operator が、現在のクラスターの状態に基づいて、安全に更新できるかどうかを示します。メッセージフィールドには、クラスタ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含まれています。CVO は、この状態が **True**、**Unknown**、または状態がない場合に更新を許可します。**Upgradeable** ステータスが **False** の場合、マイナー更新のみが影響を受け、CVO は、強制されない限り、影響を受ける更新をクラスタ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** フィールドが最新および正確である場合は、**True** です。
- ReleaseAccepted: **True** ステータスのクラスターバージョン条件タイプ **ReleaseAccepted** は、要求されたリリースペイロードが、イメージの検証および前提条件のチェック中、失敗せずに、正常に読み込まれたことを示します。
- ImplicitlyEnabledCapabilities: **True** ステータスのクラスターバージョン条件タイプ **ImplicitlyEnabledCapabilities** は、ユーザーが現在 **spec.capabilities** を介して要求していない有効な機能があることを示します。関連するリソースが以前に CVO によって管理されていた場合、CVO は機能の無効化をサポートしません。

1.5. 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 4.10 から 4.11 に更新してから、4.12 に更新する必要があります。OpenShift

Container Platform 4.10 から 4.12 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. CLI を使用したクラスターの更新

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

- 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生じる場合があります。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 コンピュートマシンを含むクラスターの更新

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

1.10. 非接続環境でのクラスターの更新

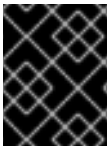
非接続環境でのクラスターの更新について: ミラーホストがインターネットとクラスターの両方にアクセスできない場合は、その環境から切断されたファイルシステムにイメージをミラーリングできます。続いて、そのホストまたはリムーバブルメディアをそのギャップを越えて移動させることができます。ローカルコンテナーレジストリーとクラスターが、レジストリーのミラーホストに接続されている場合は、リリースイメージをローカルレジストリーに直接プッシュできます。

- 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 15 以降は、クラスター内の vSphere 仮想マシンでサポートされます。詳細は、以下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

OpenShift Container Platform のバージョン 4.12 には、VMware 仮想ハードウェアバージョン 15 以降が必要です。

第2章 OPENSHIFT の更新について

2.1. OPENSHIFT の更新の概要

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

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

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

注記

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

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

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

2.1.1. 更新の可用性に関するよくある質問

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

各更新チャネルの違いは何ですか?

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

注記

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

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

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

- **fast** チャネルのリリースで回帰が特定された場合、その回帰が **stable** チャネルのリリースで特定された場合と同じ程度に解決され、管理されます。

- **fast** チャネルと **stable** チャネルのリリースの唯一の違いは、リリースが **fast** チャネル上にしばらく存在した後にのみ **stable** チャネルに表示されることです。これにより、新しい更新のリスクが発見されるまでの時間が長くなります。

- この遅延の後、**fast** チャネルで利用可能なリリースは必ず **安定** チャネルでも利用可能になります。

更新はサポートされているが推奨されていないとはどういう意味ですか?

- Red Hat は、複数のソースからのデータを継続的に評価して、あるバージョンから別のバージョンへの更新が問題を引き起こすかどうかを判断します。問題が特定された場合、更新パスはユーザーに推奨されなくなる場合があります。ただし、更新パスが推奨されない場合でも、更新を実行する場合はサポートが提供されます。

- Red Hat は、ユーザーが特定のバージョンに更新することをブロックしません。Red Hat は条件付き更新のリスクを宣言する場合がありますが、これは特定のクラスターに適用される場合と適用されない場合があります。

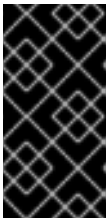
    - 宣言されたリスクにより、クラスター管理者はサポートされている更新に関する詳細なコンテキストが得られます。クラスター管理者はリスクを受け入れて、その特定のターゲットバージョンに更新することができます。この更新プログラムは、条件付きリスクの観点から推奨されていないにもかかわらず、常にサポートされています。

特定のリリースへの更新が推奨されなくなった場合はどうすればよいですか?

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

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

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



重要

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

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

関連情報

- 更新チャネルとリリースについて

2.1.2. OpenShift Update Service について

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

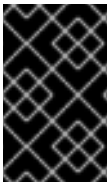
クラスター内の Cluster Version Operator (CVO) は、OpenShift Update Service をチェックして、グラフの現在のコンポーネントバージョンとグラフの情報に基づき、有効な更新および更新パスを確認します。更新をリクエストすると、CVO は対応するリリースイメージを使用してクラスターを更新します。リリースアーティファクトは、コンテナーイメージとして Quay でホストされます。

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

重要

OpenShift Update Service は、現在のクラスターに推奨される更新をすべて表示します。OpenShift Update Service が推奨する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、失敗しているかに応じて Operator の制御されたロールアウトのステータスを出力します。2 つ目のコントローラーは OpenShift Update Service をポーリングして、更新が利用可能かどうかを判別します。



重要

新しいバージョンへの更新のみがサポートされています。クラスターを以前のバージョンに戻すまたはロールバックすることはサポートされていません。更新が失敗した場合は、Red Hat サポートに連絡してください。

更新プロセスで、Machine Config Operator (MCO) は新規設定をクラスターマシンに適用します。MCO は、マシン設定プールの **maxUnavailable** フィールドで指定されたノードの数を制限し、それら

を使用不可としてマークします。デフォルトで、この値は **1** に設定されます。MCO は、**topology.kubernetes.io/zone** ラベルに基づいて、影響を受けるノードをゾーンごとにアルファベット順に更新します。ゾーンに複数のノードがある場合、最も古いノードが最初に更新されます。ベアメタルデプロイメントなど、ゾーンを使用しないノードの場合、ノードは経過時間ごとに更新され、最も古いノードが最初に更新されます。MCO は、マシン設定プールの **maxUnavailable** フィールドで指定されたノード数を一度に更新します。次に、MCO は新しい設定を適用して、マシンを再起動します。

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。

2.1.3. 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、コンピュートマシン (ワーカーマシンとしても知られる) のワークロードを管理します。

Cluster Version Operator

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

Machine Config Operator

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

OpenShift Update Service

OpenShift Update Service (OSUS) は、Red Hat Enterprise Linux CoreOS (RHCOS) を含む OpenShift Container Platform に OTA (over-the-air) 更新を提供します。コンポーネント Operator のグラフ、または頂点とそれらを結ぶ辺を含む図表が提示されます。

チャネル

チャネル は、OpenShift Container Platform のマイナーバージョンに関連付けられ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ネル、現在のバージョン、既知のバグ、およびその他の情報によって異なります。OSUS は、推奨されるエッジを、すべてのクラスターで実行される CVO に伝達します。

延長更新サポート (EUS)

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

EUS から EUS への OpenShift Container Platform リリースの更新戦略を策定できます。
詳細は、Red Hat OpenShift 延長更新サポート (EUS) の概要 を参照してください。

関連情報

- マシン設定の概要
- 非接続環境での OpenShift Update Service の使用
- 更新チャネル

2.1.4. 関連情報

- 更新プロセスの主要点に関する詳細は、クラスターの更新の仕組み を参照してください。

2.2. クラスターの更新の仕組み

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

2.2.1. Cluster Version Operator

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

2.2.1.1. ClusterVersion オブジェクト

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

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

注記

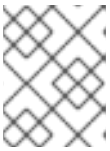
ClusterVersion オブジェクトは直接変更しないでください。代わりに、**oc** CLI や Web コンソールなどのインターフェイスを使用して、更新ターゲットを宣言します。

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

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

- 以下のコマンドを使用して、利用可能なすべての更新を確認できます。

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



注記

追加の **--include-not-recommended** パラメーターには、利用可能ではあるが、クラスターに適用される既知のリスクのため推奨されない更新が含まれます。

出力例

```
Cluster version is 4.10.22

Upstream is unset, so the cluster will use an appropriate default.
Channel: fast-4.11 (available channels: candidate-4.10, candidate-4.11, eus-4.10, fast-4.10, fast-4.11, stable-4.10)

Recommended updates:

  VERSION     IMAGE
  4.10.26     quay.io/openshift-release-dev/ocp-release@sha256:e1fa1f513068082d97d78be643c369398b0e6820afab708d26acda2262940954
  4.10.25     quay.io/openshift-release-dev/ocp-release@sha256:ed84fb3fbe026b3bbb4a2637ddd874452ac49c6ead1e15675f257e28664879cc
  4.10.24     quay.io/openshift-release-dev/ocp-release@sha256:aab51636460b5a9757b736a29bc92ada6e6e6282e46b06e6fd483063d590d62a
  4.10.23     quay.io/openshift-release-dev/ocp-release@sha256:e40e49d722cb36a95fa1c03002942b967ccbd7d68de10e003f0baa69abad457b

Supported but not recommended updates:

  Version: 4.11.0
  Image: quay.io/openshift-release-dev/ocp-release@sha256:300bce8246cf880e792e106607925de0a404484637627edf5f517375517d54a4
  Recommended: False
  Reason: RPMOSTreeTimeout
  Message: Nodes with substantial numbers of containers and CPU contention may not reconcile machine configuration https://bugzilla.redhat.com/show_bug.cgi?id=2111817#c22
```

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

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

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

出力例

```
[
```

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


- 同様のコマンドを使用して条件付き更新を確認できます。

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

出力例


```
[
  {
    "conditions": [
      {
        "lastTransitionTime": "2023-05-30T16:28:59Z",
        "message": "The 4.11.36 release only resolves an installation issue
https://issues.redhat.com//browse/OCPBUGS-11663 , which does not affect already running
clusters. 4.11.36 does not include fixes delivered in recent 4.11.z releases and therefore
upgrading from these versions would cause fixed bugs to reappear. Red Hat does not
recommend upgrading clusters to 4.11.36 version for this reason.
https://access.redhat.com/solutions/7007136",
        "reason": "PatchesOlderRelease",
        "status": "False",
        "type": "Recommended"
      }
    ],
    "release": {
      "channels": [...],
      "image": "quay.io/openshift-release-dev/ocp-
release@sha256:8c04176b771a62abd801fcda3e952633566c8b5ff177b93592e8e8d2d1f8471d
",
      "url": "https://access.redhat.com/errata/RHBA-2023:1733",
      "version": "4.11.36"
    },
    "risks": [...]
  },
  ...
]
```


2.2.1.2. 更新の可用性評価

Cluster Version Operator (CVO) は、OpenShift Update Service (OSUS) に対して、更新の可能性に関

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

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

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

Web コンソールまたは OpenShift CLI (**oc**) のユーザーインターフェイスは、この情報をセクションの見出しで表示します。**サポートされているが推奨されていない** 更新の推奨事項には、管理者が情報に基づいて更新に関する決定を下せるように、リスクに関する詳細リソースへのリンクが含まれています。

関連情報

- 更新推奨の削除と条件付き更新プログラム

2.2.2. リリースイメージ

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

次のコマンドを実行して、特定のリリースイメージの内容を検査できます。

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

出力例

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

$ ls
0000_03_authorization-openshift_01_rolebindingrestriction.crd.yaml
0000_03_config-operator_01_proxy.crd.yaml
0000_03_marketplace-operator_01_operatorhub.crd.yaml
0000_03_marketplace-operator_02_operatorhub.cr.yaml
0000_03_quota-openshift_01_clusterresourcequota.crd.yaml 1
...
0000_90_service-ca-operator_02_prometheusrolebinding.yaml 2
0000_90_service-ca-operator_03_servicemonitor.yaml
```

```
0000_99_machine-api-operator_00_tombstones.yaml
image-references 3
release-metadata
```

1 Runlevel 03 に適用される **ClusterResourceQuota** CRD のマニフェスト

2 ランレベル 90 に適用される、**service-ca-operator** の **PrometheusRoleBinding** リソースのマニフェスト

3 必要なすべてのイメージへの SHA ダイジェストバージョン参照リスト

2.2.3. プロセスワークフローの更新

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

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

10. 更新は、コントロールプレーン全体が新しいバージョンに更新されるまで続行されます。個々のクラスター Operator は、クラスターのドメインで更新タスクを実行することがあり、その場合は実行中に、**Progressing=True** 状態を通して状態を報告します。

11. Machine Config Operator (MCO) マニフェストはプロセスの最後に適用されます。その後、更新された MCO は、すべてのノードのシステム設定とオペレーティングシステムの更新を開始します。各ノードは、再びワークロードの受け入れを開始する前に、ドレイン、更新、および再起動される可能性があります。

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

2.2.4. 更新時のマニフェストの適用方法について

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

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

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

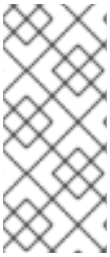
以下に例を示します。

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

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

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

次に、CVO は生成された依存関係グラフの順にマニフェストを適用します。



注記

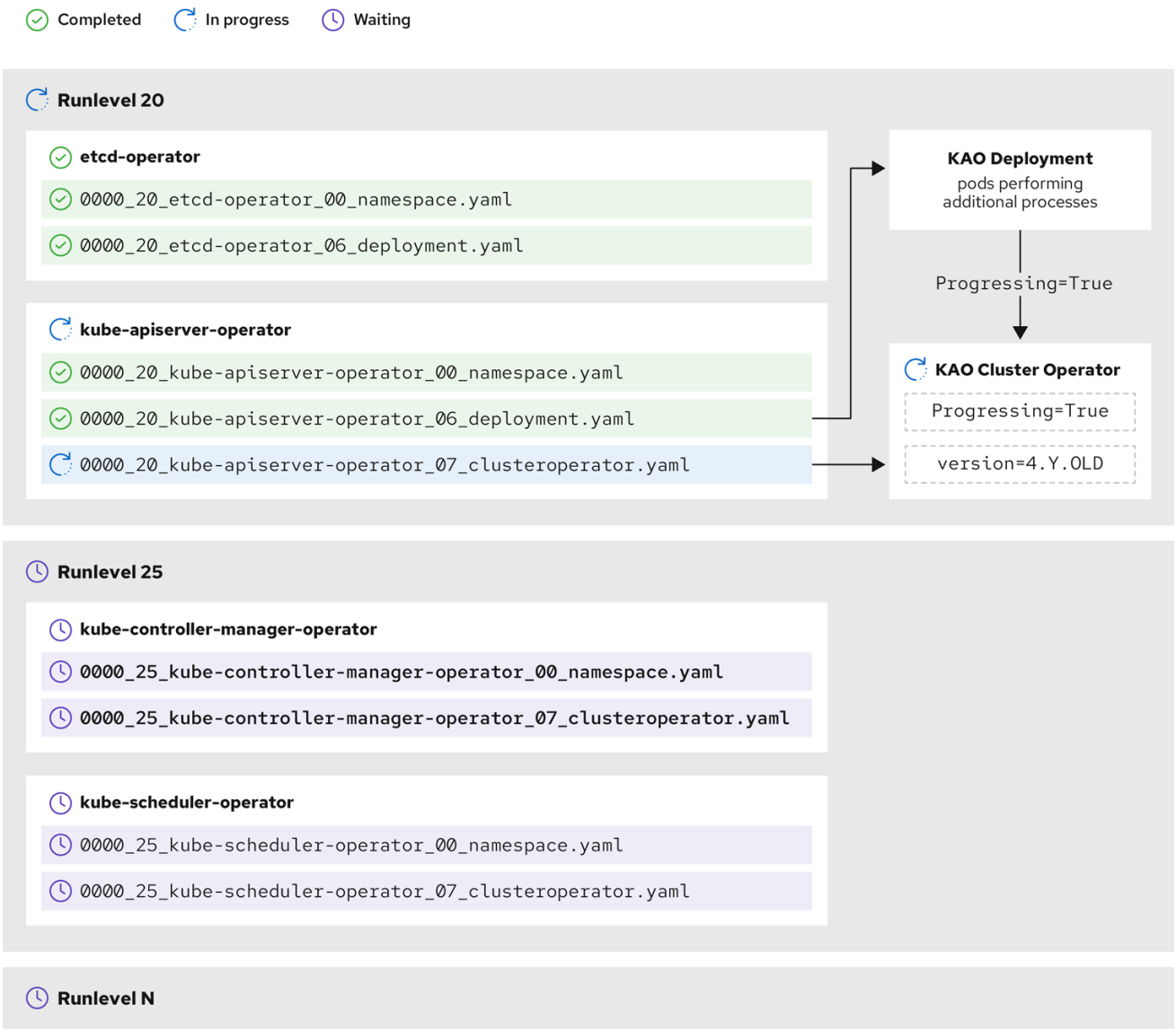
一部のリソースタイプでは、CVO はマニフェストの適用後にリソースを監視し、リソースが安定した状態に達して場合に限り正常に更新されたとみなします。この状態に達するまでに時間がかかる場合があります。これは特に **ClusterOperator** リソースに当てはまりますが、CVO はクラスター Operator が自身を更新するのを待ってから、その **ClusterOperator** ステータスを更新します。

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

- クラスター Operator は **Available=True** の状態です。
- クラスター Operator は **Degraded=False** の状態です。

- クラスター Operator は、ClusterOperator リソースで必要なバージョンになったことを宣言します。

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



前のサンプル図では、CVO は Runlevel 20 ですべての作業が完了するまで待機しています。CVO はすべてのマニフェストを Runlevel の Operator に適用しましたが、**kube-apiserver-operator ClusterOperator** は一部のアクションを新しいバージョンがデプロイされた後に実行します。**kube-apiserver-operator ClusterOperator は、 Progressing=True** の状態であることと、新しいバージョンを **status.versions** で調整済みとして宣言しないことによって、進捗を宣言します。CVO は、ClusterOperator が許容可能なステータスを報告するまで待機し、その後、Runlevel 25 でマニフェストの調整を開始します。

関連情報

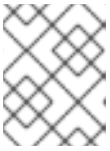
- OpenShift Container Platform の更新期間について

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

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

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

1. ノードを遮断してドレインします。



注記

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

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

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

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

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

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

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

次のコマンドを実行して、マシン設定の更新ステータスを確認できます。

```
$ oc get mcp
```

出力例

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

関連情報

- マシン設定の概要

第3章 更新チャネルとリリースについて

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

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

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

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

4.10 更新チャネルでは、4.11 以降のリリースへの更新は推奨されません。この戦略により、管理者は OpenShift Container Platform の次のマイナーバージョンに更新することを明示的に決定する必要があります。

更新チャネルはリリースの選択のみを制御し、インストールするクラスターのバージョンには影響しません。OpenShift Container Platform の特定のバージョンの **openshift-install** バイナリーファイルは、常にそのバージョンをインストールします。

OpenShift Container Platform 4.12 は、以下の更新チャネルを提供します。

- **stable-4.12**
- **eus-4.y** (EUS バージョンでのみ提供され、EUS バージョン間の更新を容易にするためのもの)
- **fast-4.12**
- **candidate-4.12**

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ネルを設定できます。この設定は、クラスターがネットワークアクセスが制限された状況で、ローカルで到達可能な更新に関する推奨サービスがない場合に役立ちます。

警告

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

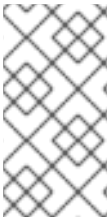
3.1. 更新チャネル

3.1.1. fast-4.12 チャネル

fast-4.12 チャネルは、Red Hat が OpenShift Container Platform 4.12 の新しいバージョンを一般公開 (GA) リリースとして宣言するとすぐに更新されます。そのため、これらのリリースは完全にサポートされており、実稼働環境での使用を目的としています。

3.1.2. stable-4.12 チャネル

fast-4.12 チャネルにはエラータが公開されるとすぐにリリースが含まれますが、リリースは遅れて **stable-4.12** チャネルに追加されます。この遅延の間に、複数のソースからデータが収集され、製品のリグレッションの兆候がないか分析されます。相当数のデータポイントが収集されると、これらのリリースは stable チャネルに追加されます。



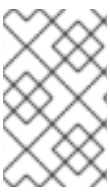
注記

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

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

3.1.3. EUS-4.y チャネル

stable チャネルのほかに、番号が偶数の OpenShift Container Platform マイナーバージョンはすべて Extended Update Support (延長更新サポート) (EUS) を提供します。stable チャネルに昇格したリリースは、同時に EUS チャネルにも昇格されます。EUS チャネルの主な目的は、EUS から EUS への更新を実行するクラスターの利便性を高めることです。



注記

標準サブスクライバーと非 EUS サブスクライバーの両方が、すべての EUS リポジトリーと必要な 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。

3.1.4. candidate-4.12 チャネル

candidate-4.12 チャネルは、リリースがビルドされるとすぐに、サポートなしですが、早期にその機能が使用できます。candidate チャネルのみに存在するリリースには、最終的な GA リリースの完全な機

能セットが含まれていないか、GA の前に機能が削除される可能性があります。さらに、これらのリリースは完全な Red Hat 品質保証の対象ではなく、後の GA リリースへの更新パスが提供されない可能性があります。これらの考慮事項を鑑みると、candidate チャネルは、クラスターの破棄と再作成が許容されるテスト目的にのみ適しています。

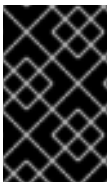
3.1.5. チャネルでの更新推奨

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

お使いのチャネルでは、以下のリリースが確認できます。

- 4.12.0
- 4.12.1
- 4.12.3
- 4.12.4

このサービスは、テスト済みで重大なリグレッションが確認されていない更新のみを推奨します。たとえば、クラスターが 4.12.1 にあり、OpenShift Container Platform が 4.12.4 を提案している場合は、4.12.1 から 4.12.4 に更新しても問題はありません。



重要

パッチの連続する番号のみに依存しないようにしてください。今回の例では、4.12.2 はこのチャネルでは今も、これまでも利用できなかったため、4.12.2 では推奨またはサポートされていません。

3.1.6. 更新の推奨と条件付き更新

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

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

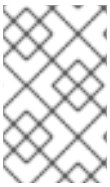
OpenShift Container Platform 4.10 以降、確認された更新リスクは、該当する更新の条件付き更新リスクとして宣言されます。既知の各リスクは、すべてのクラスターに適用される場合もあれば、特定の条件に一致するクラスターのみに適用される場合もあります。たとえば、**Platform** を **None** に、CNI プロバイダーを **OpenShiftSDN** に設定しています。Cluster Version Operator (CVO) は、現在のクラスター状態に対する既知のリスクを継続的に評価します。該当するリスクがない場合は、更新を推奨します。リスクが一致する場合、更新はサポートされますが、推奨はされません。また、参照リンクが提供されます。参照リンクは、クラスター管理者がリスクを受け入れて更新するかどうかを決定するのに役立ちます。

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

3.1.7. クラスターに適したチャネルの選択

適切なチャネルを選択する際には、2 つの点を決定する必要があります。

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



注記

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

次に、目的とするロールアウト戦略を選択する必要があります。Red Hat がリリース GA を宣言したらすぐに fast チャネルから選択して更新するか、Red Hat がリリースを stable チャネルにプロモートするのを待つかを選択できます。**fast-4.12** と **stable-4.12** で提供される更新の推奨はいずれも完全にサポートされており、同じように進行中のデータ分析からの恩恵を受けます。リリースを stable チャネルに昇格させる前の昇格の遅延は、2 つのチャネルの唯一の違いです。最新の z-stream への更新は通常、1~2 週間以内に stable チャネルに昇格されますが、最新のマイナーへの更新を最初にロールアウトするまでの時間は、通常 45 ~ 90 日と、はるかに長くなります。安定したチャネルへの昇格を待つとスケジュール計画に影響する可能性があるため、希望のチャネルを選択する際は昇格の遅延を考慮してください。

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

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

3.1.8. ネットワークが制限された環境のクラスター

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

3.1.9. CLI プロファイル間の切り替え

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

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

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

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

- **stable-4.12** チャネルから **fast-4.12** チャネルに切り換える場合も、クラスターは引き続きサポートされます。
- **candidate-4.12** チャネルにいつでも切り換えることはできますが、このチャネルの一部のリリースはサポートされない可能性があります。
- 現在のリリースが一般公開リリースの場合、**candidate-4.12** チャネルから **fast-4.12** チャネルに切り換えることができます。
- **fast-4.12** チャネルから **stable-4.12** チャネルに常に切り換えることができます。現在のリリースが最近プロモートされた場合は、リリースが **stable-4.12** 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。

関連情報

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

第4章 OPENSHIFT CONTAINER PLATFORM の更新期間について

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

4.1. 前提条件

- OpenShift Container Platform の アーキテクチャー および OpenShift Container Platform の更新 に精通していること。

4.2. 更新期間に影響する要因

次の要因は、クラスターの更新期間に影響を与える可能性があります。

- Machine Config Operator (MCO) による新しいマシン設定へのコンピュートノードの再起動
  - マシン設定プールの **MaxUnavailable** の値
  - Pod 中断バジェット (PDB) に設定されたレプリカの最小数またはパーセンテージ
- クラスター内のノード数
- クラスターノードの可用性

4.3. クラスターの更新フェーズ

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

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

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

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

注記

更新の CVO フェーズでは、ノードは再起動されません。

関連情報

- OpenShift の更新の概要

4.3.2. Machine Config Operator ノードの更新

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

1. すべてのノードを遮断してドレインする
2. オペレーティングシステム (OS) を更新する
3. ノードを再起動します。
4. すべてのノードのコードを解除し、ノードでワークロードをスケジュールします

注記

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

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

MCO に加えて、次のパラメーターの影響を考慮する必要があります。

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

```
$ oc get node
```

出力例

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

```
ip-10-0-204-102.us-east-2.compute.internal  Ready                 master  12d
v1.23.5+3afdacb
ip-10-0-207-224.us-east-2.compute.internal  Ready                 worker  12d
v1.23.5+3afdacb
```

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

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

関連情報

- マシン設定の概要
- Pod の Disruption Budget (停止状態の予算)

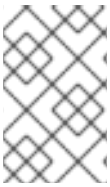
4.4. クラスター更新時間の概算

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

```
Cluster update time = CVO target update payload deployment time + (# node update iterations x
MCO node update time)
```

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

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



注記

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

シナリオ 1:

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

```
Cluster update time = 60 + (6 x 5) = 90 minutes
```

シナリオ 2

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

```
Cluster update time = 60 + (3 x 5) = 75 minutes
```

重要

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

4.5. RED HAT ENTERPRISE LINUX (RHEL) コンピュートノード

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

関連情報

- RHEL コンピュートマシンの更新

第5章 OPENSHIFT CONTAINER PLATFORM 4.12 への更新の準備

OpenShift Container Platform 4.12 は Kubernetes 1.25 を使用します。これにより、いくつかの非推奨 API が削除されました。

クラスター管理者は、クラスターを OpenShift Container Platform 4.11 から 4.12 にアップグレードする前に、手動で確認を行う必要があります。削除された API が、クラスター上で実行されている、またはクラスターと対話しているワークロード、ツール、またはその他のコンポーネントによって引き続き使用される OpenShift Container Platform 4.12 にアップグレードした後の問題を防ぐ上で役立ちます。管理者は、削除が予定されている使用中の API に対するクラスターの評価を実施し、影響を受けるコンポーネントを移行して適切な新規 API バージョンを使用する必要があります。この評価および移行が完了したら、管理者は確認応答を提供できます。

OpenShift Container Platform 4.11 クラスターを 4.12 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。

5.1. KUBERNETES API の削除

OpenShift Container Platform 4.12 は Kubernetes 1.25 を使用します。これにより、以下の非推奨 API が削除されました。適切な API バージョンを使用するには、マニフェストと API クライアントを移行する必要があります。削除された API の移行についての詳細は、Kubernetes documentation を参照してください。

表5.1 Kubernetes 1.25 から削除された API

リソース	削除された API	移行先	大きな変更
CronJob	**batch/v1beta1**	**batch/v1**	いいえ
EndpointSlice	**discovery.k8s.io/v1beta1**	**discovery.k8s.io/v1**	はい
イベント	**events.k8s.io/v1beta1**	**events.k8s.io/v1**	はい
HorizontalPodAutoscaler	**autoscaling/v2beta1**	**autoscaling/v2**	いいえ
PodDisruptionBudget	**policy/v1beta1**	**policy/v1**	はい
PodSecurityPolicy	**policy/v1beta1**	Pod セキュリティーアドミッション [1]	はい
RuntimeClass	**node.k8s.io/v1beta1**	**node.k8s.io/v1**	いいえ

1. OpenShift Container Platform での Pod セキュリティーアドミッションの詳細については、**Pod セキュリティーアドミッションの理解と管理** を参照してください。

関連情報

- Pod セキュリティーアドミッションの理解と管理

5.2. 削除された API に対するクラスターの評価

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

5.2.1. 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。

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

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

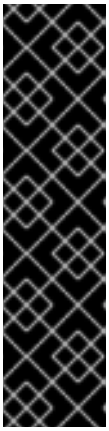
- 以下のコマンドを実行し、出力された **REMOVEDINRELEASE** 列を確認して、現在使用中の削除された API を特定します。

```
$ oc get apirequestcounts
```

出力例

```
NAME                                                                                    REMOVEDINRELEASE
REQUESTSINCURRENTHOUR   REQUESTSINLAST24H
...
poddisruptionbudgets.v1.policy                                                                                391                     8114
poddisruptionbudgets.v1beta1.policy                                                                   1.25      2                     23
podmonitors.v1.monitoring.coreos.com                                                                          3                     70
podnetworkconnectivitychecks.v1alpha1.controlplane.operator.openshift.io                                                      612
11748
pods.v1                                                                                                    1531                   38634
podsecuritypolicies.v1beta1.policy                                                                    1.25      3                     39
podtemplates.v1                                                                                               2                     79
```

```
preprovisioningimages.v1alpha1.metal3.io                                  2                 39
priorityclasses.v1.scheduling.k8s.io                                      12                248
prioritylevelconfigurations.v1beta1.flowcontrol.apiserver.k8s.io          1.26              3
86
...
```



重要

結果に表示される以下のエントリーは無視しても問題はありません。

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

-o jsonpath を使用して結果をフィルタリングすることもできます。

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

出力例

```
1.26 flowschemas.v1beta1.flowcontrol.apiserver.k8s.io
1.26 horizontalpodautoscalers.v2beta2.autoscaling
1.25 poddisruptionbudgets.v1beta1.policy
1.25 podsecuritypolicies.v1beta1.policy
1.26 prioritylevelconfigurations.v1beta1.flowcontrol.apiserver.k8s.io
```

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

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

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

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

以下に例を示します。

```
$ oc get apirequestcounts poddisruptionbudgets.v1beta1.policy -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 poddisruptionbudgets.v1beta1.policy \
  -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  system:serviceaccount:openshift-operators:3scale-operator
manager/v0.0.0
watch  system:serviceaccount:openshift-operators:datadog-operator-controller-manager
manager/v0.0.0
```

5.3. 削除された API インスタンスの移行

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

5.4. 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 4.11 から 4.12 にアップグレードできることを確認できます。

警告

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

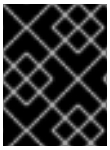
手順

- 以下のコマンドを実行して、評価が完了し、クラスターが OpenShift Container Platform 4.12 で Kubernetes API を削除する準備ができていることを確認します。

```
$ oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.11-kube-1.25-api-
removals-in-4.12":"true"}}' --type=merge
```

第6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 の更新をシリアライズする必要があります。OpenShift Container Platform <4.y> から <4.y+1> に更新してから、<4.y+2> に更新する必要があります。OpenShift Container Platform <4.y> から <4.y+2> に直接更新することはできません。ただし、2 つの Extended Update Support (EUS) バージョン間で更新したい管理者は、非コントロールプレーンホストを 1 回再起動するだけで更新できます。



重要

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

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

- EUS から EUS への更新は、関連するすべてのバージョン間の更新が **stable** チャ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る場合があります。
- ワーカーまたはカスタムプールノードを更新して、メンテナンスにかかる時間に対応することにより、部分的な更新を行うことができます。
- 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。
- マシン設定プールの一時停止が解除され、更新が完了するまで、OpenShift Container Platform の <4.y+1> および <4.y+2> の一部の機能およびバグ修正は利用できません。
- すべてのクラスターは、プールを一時停止せずに従来の更新に EUS チャ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。

6.1. EUS から EUS への更新

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

前提条件

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

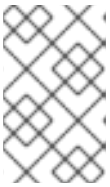
6.1.1. Web コンソールを使用した EUS から EUS への更新

前提条件

- マシン設定プールの一時停止が解除されている。
- **admin** 権限を持つユーザーとして Web コンソールにアクセスできる。

手順

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



注記

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

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

重要

プールの一時停止が解除されていない場合、クラスターは今後のマイナーバージョン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。

11. 以前に一時停止したプールが更新され、クラスターがバージョン <4.y+2> への更新を完了したことを確認します。
Compute ページの **MachineConfigPools** タブで、**Update status** の値が **Up to date** になっていることを確認して、プールが更新されたことを確認できます。

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

関連情報

- Operator 更新の準備
- Web コンソールを使用したクラスターの更新
- インストール済み Operator の更新

6.1.2. CLI を使用した EUS から EUS への更新

前提条件

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

重要

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

手順

1. Web コンソールの管理者パースペクティブを使用して、任意の Operator Lifecycle Manager (OLM) Operator を、目的の更新バージョンと互換性のあるバージョンに更新します。このアクションを実行する方法は、インストール済み Operator の更新を参照してください。

2. すべてのマシン設定プールが **UPDATED** のステータスを表示し、マシン設定プールが **UPDATING** のステータスを表示していないことを確認します。すべてのマシン設定プールのステータスを表示するには、以下のコマンドを実行します。

```
$ oc get mcp
```

出力例

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

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

```
$ oc adm upgrade channel eus-<4.y+2>
```

注記

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

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

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

注記

マスタープールを一時停止することはできません。

5. 次のコマンドを実行して、最新バージョンに更新します。

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

出力例

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

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 adm upgrade
```

出力例

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

7. 次のコマンドを実行して、バージョン <4.y+2> に更新します。

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

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

```
$ oc adm upgrade
```

出力例

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

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

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

重要

プールの一時停止が解除されていない場合、クラスターは将来のマイナーバージョンへの更新が許可されず、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。

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

```
$ oc get mcp
```

出力例

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

関連情報

- インストール済み Operator の更新

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

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

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

レイヤード製品とは

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

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

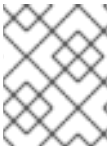
1. これまで　OLM でインストールされたすべての Operator が、最新チャネルの最新バージョンに更新されていることを確認します。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの更新時に現行のマイナーバージョンから次のマイナーバージョンに切り替わる際、確実に有効な更新パスがあるようにします。Operator の更新方法については、関連情報の 「Operator 更新の準備」を参照してください。

2. 現在の Operator バージョンと更新後の Operator バージョン間のクラスターバージョン互換性を確認します。Red Hat OpenShift Container Platform Operator Update Information Checker を使用して、OLM Operator と互換性があるバージョンを確認できます。

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

ワークフローの例

1. ワーカーマシンプールを一時停止します。
2. OpenShift <4.y> → OpenShift <4.y+1> にアップグレードします。
3. ODF <4.y> → ODF <4.y+1> にアップグレードします。
4. OpenShift <4.y+1> → OpenShift <4.y+2> にアップグレードします。
5. ODF <4.y+2> にアップグレードします。
6. ワーカーマシンプールの一時停止を解除します。



注記

ODF <4.y+2> へのアップグレードは、ワーカーマシンプールの一時停止が解除される前または後に実行できます。

関連情報

- Operator 更新の準備
- Web コンソールを使用した EUS から EUS への更新
- CLI を使用した EUS から EUS への更新

第7章 手動で維持された認証情報でクラスターを更新する準備

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

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

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

7.1. 手動で維持された認証情報を使用したクラスターの更新要件

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

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

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

注記

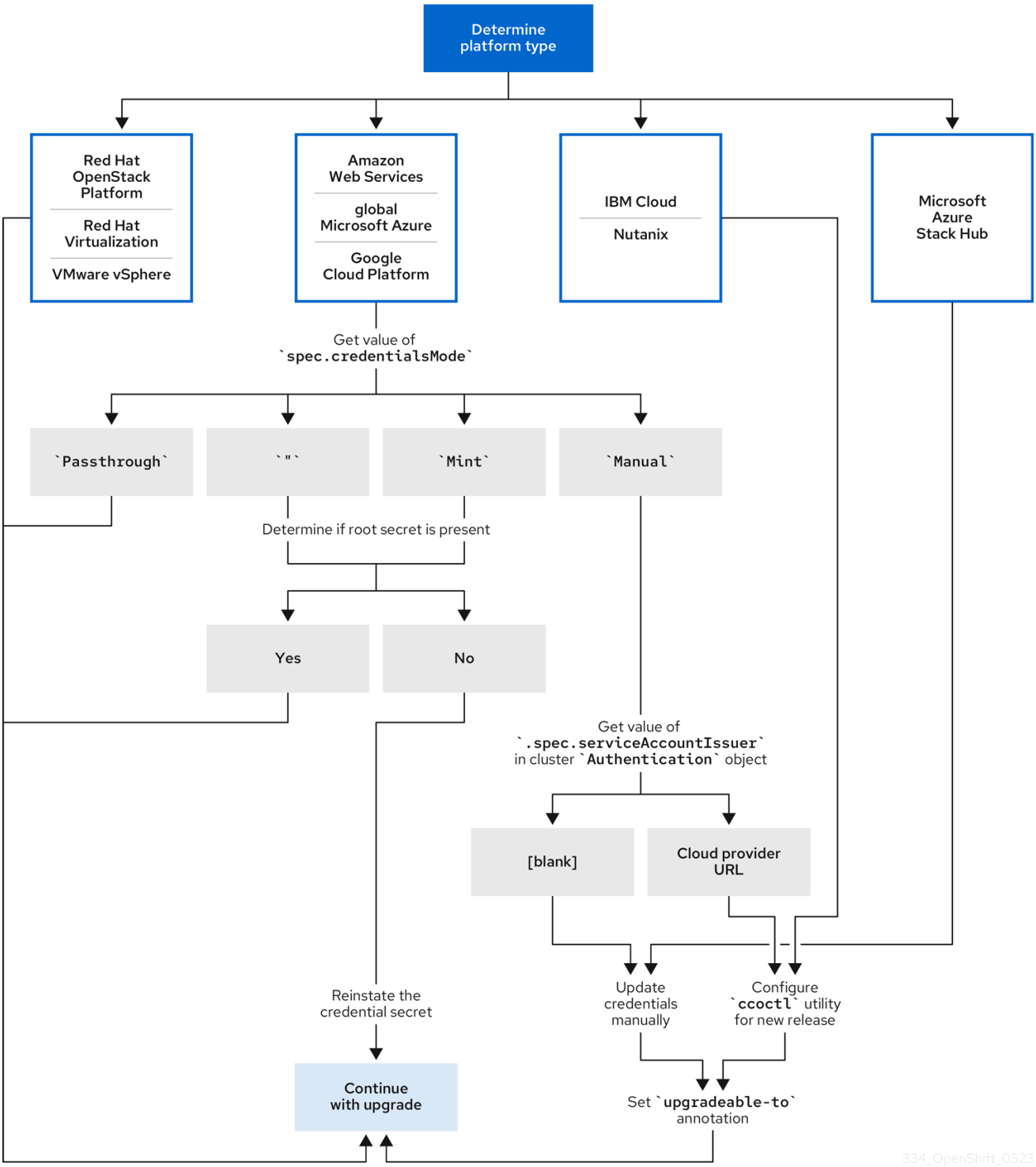
クラウドプロバイダーリソースと **upgradeable-to** アノテーションを更新するプロセスは、コマンドラインツールを使用しなければ完了できません。

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

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

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

図7.1 プラットフォームタイプ別の認証情報の更新要件



Red Hat OpenStack Platform (RHOSP)、Red Hat Virtualization (RHV)、VMware vSphere

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

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

{IBM-cloud-title} および Nutanix

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

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

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

Microsoft Azure Stack Hub

これらのクラスターは、有効期間の長い認証情報と手動モードを使用し、**ccoctl** ユーティリティーは使用しません。
これらのプラットフォーム上にあるクラスターの管理者は、以下のアクションを実行する必要があります。

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

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

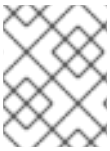
これらのプラットフォームにインストールされたクラスターは、複数の CCO モードをサポートします。
必要な更新プロセスは、クラスターが使用するように設定されたモードにより異なります。CCO がクラスターで使用するように設定されたモードが不明な場合は、Web コンソールまたは CLI を使用して判別できます。

関連情報

- Web コンソールを使用した Cloud Credential Operator モードの判別
- CLI を使用した Cloud Credential Operator モードの判別
- クラスター更新のための Cloud Credential Operator ユーティリティーの設定
- 手動で維持された認証情報によるクラウドプロバイダーリソースの更新
- Cloud Credential Operator について

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

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



注記

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

前提条件

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

手順

手順

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

重要

spec.credentialsMode が **''**、**Mint**、または **Manual** である AWS または GCP クラスターの特定の設定を特定するには、さらに調査する必要があります。

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

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

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

注記

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

プラットフォーム	シークレット名
AWS	**aws-creds**

プラットフォーム	シークレット名
GCP	**gcp-credentials**

- これらの値のいずれかが表示される場合、クラスターはルートシークレットが存在するミントモードまたはパススルーモードを使用しています。
- これらの値が表示されない場合、クラスターはルートシークレットが削除されたミントモードで CCO を使用しています。

8. 手動モードのみを使用する AWS または GCP クラスター: クラスターがクラスターの外部からクラウド認証情報を作成および管理するように設定されているかどうかを判断するには、クラスター **Authentication** オブジェクトの YAML 値を確認する必要があります。
   a. **Administration** → **Cluster Settings** に移動します。
   b. **Cluster Settings** ページで、**Configuration** タブを選択します。
   c. **Configuration resource** で **Authentication** を選択します。
   d. **Authentication details** ページで、**YAML** タブを選択します。
   e. YAML ブロックで、**.spec.serviceAccountIssuer** パラメーターの値を確認します。
      - クラウドプロバイダーに関連付けられている URL を含む値は、CCO が AWS STS または GCP Workload Identity で手動モードを使用して、クラスターの外部からクラウド認証情報を作成および管理していることを示します。これらのクラスターは、**ccoctl** ユーティリティーを使用して設定されます。
      - 空の値 ('') は、クラスターが手動モードで CCO を使用しているが、**ccoctl** ユーティリティーを使用して設定されていないことを示します。

次のステップ

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

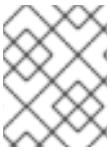
b. **upgradeable-to** アノテーションを更新して、クラスターの更新準備が完了していることを示します。

関連情報

- クラスター更新のための Cloud Credential Operator ユーティリティーの設定
- 手動で維持された認証情報によるクラウドプロバイダーリソースの更新

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

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



注記

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

前提条件

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

手順

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

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

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

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

重要

spec.credentialsMode が **''**、**Mint**、または **Manual** である AWS または GCP クラスターの特定の設定を特定するには、さらに調査する必要があります。

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

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

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

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

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

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

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

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

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

- クラウドプロバイダーに関連付けられている URL の出力は、CCO が AWS STS または GCP Workload Identity で手動モードを使用して、クラスターの外部からクラウド認証情報を作成および管理していることを示しています。これらのクラスターは、**ccoctl** ユーティリティーを使用して設定されます。

- 空の出力は、クラスターが手動モードで CCO を使用しているが、**ccoctl** ユーティリティーを使用して設定されていないことを示します。

次のステップ

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

- クラスターが、root シークレットが削除された状態で mint モードの CCO を使用している場合、更新プロセスの次の手順に進む前に、管理者レベルの認証情を使用して認証情報シークレットを元に戻す必要があります。

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

関連情報

- クラスター更新のための Cloud Credential Operator ユーティリティーの設定
- 手動で維持された認証情報によるクラウドプロバイダーリソースの更新

7.2. クラスター更新のための CLOUD CREDENTIAL OPERATOR ユーティリティーの設定

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

注記

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

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- クラスターは、クラスターの外からクラウド認証情報を作成および管理するために**ccoctl** ユーティリティーを使用して設定されています。

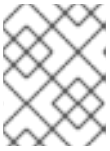
手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージを取得します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

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

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



注記

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

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

```
$ oc image extract $CCO_IMAGE --file="/usr/bin/ccoctl" -a ~/.pull-secret
```

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

```
$ chmod 775 ccoctl
```

検証

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

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  alibabacloud Manage credentials objects for alibaba cloud
  aws          Manage credentials objects for AWS cloud
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for IBM Cloud
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

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

7.3. CLOUD CREDENTIAL OPERATOR ユーティリティーを使用したクラウドプロバイダーリソースの更新

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

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

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

前提条件

- アップグレードするバージョンの OpenShift Container Platform リリースイメージを取得します。
- リリースイメージから **ccoctl** バイナリーを抽出して準備します。

手順

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

```
$ oc adm release extract --credentials-requests \
  --cloud=<provider_type> \
  --to=<path_to_directory_with_list_of_credentials_requests>/credrequests \
  quay.io/<path_to>/ocp-release:<version>
```

   ここでは、以下のようになります。

   - **<provider_type>** は、クラウドプロバイダーの値です。有効な値は **alibabacloud**、**aws**、**gcp**、**ibmcloud**、**nutanix** です。
   - **credrequests** は、**CredentialsRequest** オブジェクトのリストが格納されるディレクトリーです。ディレクトリーが存在しない場合、このコマンドはディレクトリーを作成します。

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

   サンプル AWS CredentialsRequest オブジェクト

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

```
    action:
    - iam:GetUser
    - iam:GetUserPolicy
    - iam:ListAccessKeys
    resource: "*"
  secretRef:
    name: cloud-credential-operator-iam-ro-creds
    namespace: openshift-cloud-credential-operator
```
(1)

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

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

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

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

4. **ccoctl** ツールを使用し、適切なクラウドプロバイダーのコマンドを実行して **credrequests** ディレクトリー内のすべての **CredentialsRequest** オブジェクトを処理します。以下のコマンドは **CredentialsRequest** オブジェクトを処理します。

   - Alibaba Cloud: **ccoctl alibabacloud create-ram-users**
   - Amazon Web Services (AWS): **ccoctl aws create-iam-roles**
   - Google Cloud Platform (GCP): **ccoctl gcp create-all**
   - IBM Cloud: **ccoctl ibmcloud create-service-id**
   - Nutanix: **ccoctl nutanix create-shared-secrets**

重要

プラットフォームによって異なる必要な引数および特別な考慮事項について、詳しくはクラウドプロバイダーのインストールコンテンツで **ccoctl** ユーティリティーの手順を参照してください。

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

5. 次のコマンドを実行して、シークレットをクラスターに適用します。

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

検証

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

次のステップ

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

関連情報

- **ccoctl** ツールを使用した OpenShift Container Platform コンポーネントの Alibaba Cloud 認証情報の作成
- Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成
- Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成
- IBM Cloud VPC 用の IAM を手動で作成する
- Nutanix の IAM の設定
- クラスターがアップグレードの準備ができていることを示す

7.4. 手動で維持された認証情報によるクラウドプロバイダーリソースの更新

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。また、既存の認証情報に必要なアクセス許可を確認し、それらのコンポーネントの新しいリリースでの新しいアクセス許可要件に対応する必要があります。

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、使用する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. クラスターでクラスター機能を使用して 1 つ以上のオプションコンポーネントを無効にする場合は、無効なコンポーネントの **CredentialsRequest** カスタムリソースを削除します。

AWS 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_30_machine-api-operator_00_credentials-request.yaml 1
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml 2
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml 3
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 4
0000_50_cluster-network-operator_02-cncc-credentials.yaml 5
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml 6
```

1. Machine API Operator CR が必要です。
2. Cloud Credential Operator CR が必要です。
3. Image Registry Operator CR が必要です。
4. Ingress Operator CR が必要です。
5. Network Operator CR が必要です。
6. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

GPC 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_26_cloud-controller-manager-operator_16_credentialsrequest-gcp.yaml 1
0000_30_machine-api-operator_00_credentials-request.yaml 2
0000_50_cloud-credential-operator_05-gcp-ro-credentialsrequest.yaml 3
0000_50_cluster-image-registry-operator_01-registry-credentials-request-gcs.yaml 4
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 5
0000_50_cluster-network-operator_02-cncc-credentials.yaml 6
0000_50_cluster-storage-operator_03_credentials_request_gcp.yaml 7
```

1. Cloud Controller Manager Operator CR が必要です。
2. Machine API Operator CR が必要です。
3. Cloud Credential Operator CR が必要です。
4. Image Registry Operator CR が必要です。
5. Ingress Operator CR が必要です。
6. Network Operator CR が必要です。
7. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

次のステップ

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

関連情報

- AWS の IAM の手動作成
- Azure の IAM の手動作成
- Azure Stack Hub の IAM の手動作成
- GCP の IAM の手動作成

- クラスターがアップグレードの準備ができていることを示す

7.5. クラスターがアップグレードの準備ができていることを示す

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

前提条件

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

手順

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

```
$ oc edit cloudcredential cluster
```

追加するテキスト

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

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

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

検証

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

第8章 WEB コンソールを使用してクラスターを更新

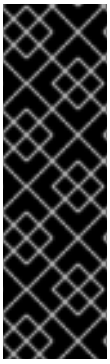
Web コンソールを使用して、OpenShift Container Platform クラスターでマイナーバージョンおよびパッチの更新を実行できます。

注記

Web コンソールまたは **oc adm upgrade channel <channel>** を使用して更新チャネルを変更します。4.16 チャネルに変更した後、CLI を使用したクラスターの更新 の手順に従って更新を完了できます。

8.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできる。RBAC の使用によるパーミッションの定義および適用 を参照してください。
- 更新が失敗し、クラスターを以前の状態に復元する 必要がある場合に、最新の etcd バックアップ がある こと。
- RHEL7 ワーカーのサポートは OpenShift Container Platform 4.12 では削除されています。OpenShift Container Platform 4.12 にアップグレードする前に、RHEL7 ワーカーを RHEL8 または RHCOS ワーカーに置き換える必要があります。Red Hat は、RHEL ワーカーの RHEL7 から RHEL8 のインプレース更新をサポートしません。このホストは、クリーンなオペレーティングシステムインストールに置き換える必要があります。
- Operator Lifecycle Manager (OLM) で以前にインストールされたすべての Operator が、最新チャネルの最新バージョンに更新されていることを確認します。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの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。
- 更新にかかる時間に対応するために、ワーカーノードまたはカスタムプールノードを更新することで部分的な更新を行うことができます。各プールのプログレスバー内で一時停止および再開できます。
- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは 手動で維持された認証情報でクラスターを更新する準備 を参照してください。
- Kubernetes 1.25 で削除された API のリストを確認し、影響を受けるすべてのコンポーネントを移行して新しい API バージョンを使用し、管理者に承認を提供します。詳細は、OpenShift Container Platform 4.16 への更新の準備 を参照してください。
- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合、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

8.2. カナリアロールアウト更新の実行

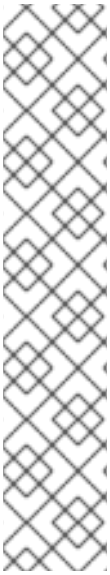
特定のユースケースでは、特定ノードを残りのクラスターと同時に更新しない、制御された更新プロセスが必要になる場合があります。これらのユースケースには、以下のようなものがありますが、これに限定されません。

- 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と、新規証明書が作成されますが、適切なマシン設定プールのノード全体では適用されません。これにより、**oc debug**、**oc logs**、**oc exec**、**oc attach** などの複数の **oc** コマンドでエラーが発生します。証明書がローテーションされたときに MCP が一時停止された場合、OpenShift Container Platform コンソールのアラート UI でアラートを受け取ります。

MCP の一時停止は、**kube-apiserver-to-kubelet-signer** CA 証明書の有効期限を慎重に考慮して、短期間のみ行う必要があります。

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。

8.3. 手動で維持された認証情報によるクラウドプロバイダーリソースの更新

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。また、既存の認証情報に必要なアクセス許可を確認し、それらのコンポーネントの新しいリリースでの新しいアクセス許可要件に対応する必要があります。

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、使用する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. クラスターでクラスター機能を使用して 1 つ以上のオプションコンポーネントを無効にする場合は、無効なコンポーネントの **CredentialsRequest** カスタムリソースを削除します。

AWS 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_30_machine-api-operator_00_credentials-request.yaml 1
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml 2
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml 3
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 4
0000_50_cluster-network-operator_02-cncc-credentials.yaml 5
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml 6
```

1. Machine API Operator CR が必要です。
2. Cloud Credential Operator CR が必要です。
3. Image Registry Operator CR が必要です。
4. Ingress Operator CR が必要です。
5. Network Operator CR が必要です。
6. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

GPC 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_26_cloud-controller-manager-operator_16_credentialsrequest-gcp.yaml 1
0000_30_machine-api-operator_00_credentials-request.yaml 2
0000_50_cloud-credential-operator_05-gcp-ro-credentialsrequest.yaml 3
0000_50_cluster-image-registry-operator_01-registry-credentials-request-gcs.yaml 4
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 5
0000_50_cluster-network-operator_02-cncc-credentials.yaml 6
0000_50_cluster-storage-operator_03_credentials_request_gcp.yaml 7
```

1. Cloud Controller Manager Operator CR が必要です。
2. Machine API Operator CR が必要です。
3. Cloud Credential Operator CR が必要です。
4. Image Registry Operator CR が必要です。
5. Ingress Operator CR が必要です。
6. Network Operator CR が必要です。
7. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

次のステップ

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

関連情報

- AWS の IAM の手動作成
- Azure の IAM の手動作成
- GCP の IAM の手動作成

8.4. 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。

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

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

ただし、以下の制限事項に注意してください。

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

- 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 が短時間してすぐに解決します。

重要

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

関連情報

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

8.6. 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。

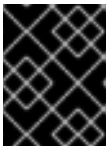
利用可能な OpenShift Container Platform アドバイザリーおよび更新については、カスタマーポータルの エラータ のセクションを参照してください。

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

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、**チャネル** が、**stable-4.12** など、更新するバージョンの正しいチャネルに設定されていることを確認してください。



重要

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

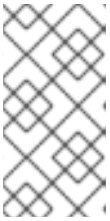
注記

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

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

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

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



注記

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

4. 更新が完了し、Cluster Version Operator が利用可能な更新を更新したら、追加の更新が現在のチャネルで利用可能かどうかを確認します。
   - 更新が利用可能な場合は、更新ができなくなるまで、現在のチャネルでの更新を継続します。
   - 利用可能な更新がない場合は、**チャネル** を次のマイナーバージョンの **stable-***、**eus-*** または **fast-*** チャネルに変更し、そのチャネルで必要なバージョンに更新します。

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

8.7. 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>' 1
...
```

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

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

3. **Save** をクリックします。

関連情報

- 更新チャネルとリリースについて

第9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 クラスターでマイナーバージョンおよびパッチの更新を実行できます。

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。RBAC の使用によるパーミッションの定義および適用 を参照してください。
- 更新が失敗し、クラスターを以前の状態に復元する 必要がある場合に、最新の etcd バックアップ がある こと。
- RHEL7 ワーカーのサポートは OpenShift Container Platform 4.12 では削除されています。OpenShift Container Platform 4.12 にアップグレードする前に、RHEL7 ワーカーを RHEL8 または RHCOS ワーカーに置き換える必要があります。Red Hat は、RHEL ワーカーの RHEL7 から RHEL8 のインプレース更新をサポートしません。このホストは、クリーンなオペレーティングシステムインストールに置き換える必要があります。
- Operator Lifecycle Manager (OLM) で以前にインストールされたすべての Operator が、最新チャネルの最新バージョンに更新されていることを確認します。Operator を更新することで、デフォルトの OperatorHub カタログが、クラスターの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 を一時停止できる。
- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは 手動で維持された認証情報でクラスターを更新する準備 を参照してください。
- クラスターで次のマイナーバージョンへの更新ができるように、すべての **Upgradeable=False** 条件に対応してください。アラートは、アップグレードできない 1 つ以上のクラスター Operator がある場合に **Cluster Settings** ページの上部に表示されます。引き続き、現在使用しているマイナーリリースについて、次に利用可能なパッチ更新に更新できます。
- Kubernetes 1.25 で削除された API のリストを確認し、影響を受けるすべてのコンポーネントを移行して新しい API バージョンを使用し、管理者に承認を提供します。詳細は、OpenShift Container Platform 4.16 への更新の準備 を参照してください。
- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合、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

9.2. MACHINEHEALTHCHECK リソースの一時停止

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

前提条件

- OpenShift CLI (**oc**) がインストールされている。

手順

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

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

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

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

アノテーション付きの **MachineHealthCheck** リソースは以下の YAML ファイルのようになります。

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

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

重要

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

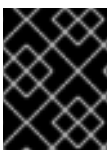
```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused-
```

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

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

ただし、以下の制限事項に注意してください。

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



重要

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

関連情報

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

9.4. CLI を使用したクラスターの更新

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

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

前提条件

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

手順

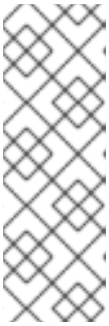
1. 利用可能な更新を確認し、適用する必要のある更新のバージョン番号をメモします。

```
$ oc adm upgrade
```

出力例

```
Cluster version is 4.9.23
Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.9 (available channels: candidate-4.10, candidate-4.9, fast-4.10, fast-4.9,
stable-4.10, stable-4.9, eus-4.10)
Recommended updates:
VERSION IMAGE
4.9.24  quay.io/openshift-release-dev/ocp-
release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056df6d018032
4.9.25  quay.io/openshift-release-dev/ocp-
release@sha256:2eafde815e543b92f70839972f585cc52aa7c37aa72d5f3c8bc886b0fd45707a

4.9.26  quay.io/openshift-release-dev/ocp-
release@sha256:3ccd09dd08c303f27a543351f787d09b83979cd31cf0b4c6ff56cd68814ef6c8

4.9.27  quay.io/openshift-release-dev/ocp-
release@sha256:1c7db78eec0cf05df2cead44f69c0e4b2c3234d5635c88a41e1b922c3bedae16

4.9.28  quay.io/openshift-release-dev/ocp-
release@sha256:4084d94969b186e20189649b5affba7da59f7d1943e4e5bc7ef78b981eafb7a8

4.9.29  quay.io/openshift-release-dev/ocp-
release@sha256:b04ca01d116f0134a102a57f86c67e5b1a3b5da1c4a580af91d521b8fa0aa6ec

4.9.31  quay.io/openshift-release-dev/ocp-
release@sha256:2a28b8ebb53d67dd80594421c39e36d9896b1e65cb54af81fbb86ea9ac3bf2d
```

```
7
4.9.32  quay.io/openshift-release-dev/ocp-
release@sha256:ecdb6d0df547b857eaf0edb5574ddd64ca6d9aff1fa61fd1ac6fb641203bedfa
```



注記

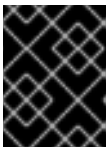
- 利用可能な更新がない場合でも、サポート対象であるが推奨はされない更新が利用できる可能性があります。詳細は、**条件付き更新パスに沿った更新** を参照してください。
- **EUS-to-EUS** チャンネル更新を実行する方法の詳細と情報は、関連情報セクションにリストされている **EUS-to-EUS アップグレードを実行するための準備** ページを参照してください。

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

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

たとえば、チャネルを **stable-4.12** に設定するには、以下を実行します。

```
$ oc adm upgrade channel stable-4.12
```



重要

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

注記

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

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

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

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

   - 特定のバージョンに更新するには、以下を実行します。

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

(1) (1) **<version>** は、**oc adm upgrade** コマンドの出力から取得した更新バージョンです。

4. クラスターバージョン Operator を確認します。

```
$ oc adm upgrade
```

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

```
$ oc get clusterversion
```

出力例

```
Cluster version is <version>

Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.10 (available channels: candidate-4.10, candidate-4.11, eus-4.10, fast-4.10,
fast-4.11, stable-4.10)

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

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

```
$ oc get nodes
```

出力例

```
NAME                         STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.25.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.25.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.25.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.25.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.25.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.25.0
```

関連情報

- 条件付き更新パスに沿った更新
- EUS から EUS への更新を実行するための準備
- 更新チャネルとリリースについて

9.5. 条件付きアップグレードパスに沿った更新

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

手順

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

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

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

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

<.> **<version>** は、前のコマンドの出力から取得した、サポートされているが推奨されていない更新バージョンです。

関連情報

- 更新チャネルとリリースについて

9.6. 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10章 カナリアロールアウト更新の実行

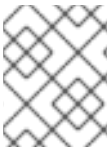
カナリア更新 は、すべてのワーカーノードを同時に更新するのではなく、ワーカーノードの更新を個別の段階に分けて順次実行する更新ストラテジーです。このストラテジーは、次の場合に役立ちます。

- ワーカーノード更新のロールアウトをより細かく制御して、更新プロセスによってアプリケーションが失敗した場合でも、更新全体を通じてミッションクリティカルなアプリケーションが利用可能な状態を維持する必要がある。
- 少数のワーカーノードを更新し、一定期間にわたりクラスターとワークロードの健全性を評価してから、残りのノードを更新する必要がある。
- クラスター全体を一度に更新するために長いメンテナンス期間を取ることができない場合に、通常はホストの再起動が必要となるワーカーノードの更新を、より短い一定のメンテナンス期間内に収める必要がある。

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

10.1. カナリア更新ストラテジーの例

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



注記

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

カスタムマシン設定プールの定義
ワーカーノードの更新を個別の段階に分けて編成するために、まず次の MCP を定義します。

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

カナリアワーカープールの更新
最初のメンテナンス期間中、**workerpool-A**、**workerpool-B**、および **workerpool-C** の MCP を一時停止してから、クラスターの更新を開始します。これにより、OpenShift Container Platform 上で実行されるコンポーネントと、一時停止されていない **workerpool-canary** MCP に含まれる 10 個のノードが更新されます。他の 3 つの MCP は一時停止されているため、更新されません。

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

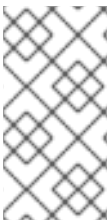
カスタム MCP を使用してワーカーノードの更新を管理すると、柔軟性が得られます。一方で、複数のコマンドを実行する必要があるため、時間のかかるプロセスになる可能性があります。この複雑さにより、クラスター全体に影響を及ぼす可能性のあるエラーが発生する可能性があります。開始する前に、組織のニーズを慎重に検討し、プロセスの実装を慎重に計画することを推奨します。

重要

マシン設定プールを一時停止にすると、Machine Config Operator が関連付けられたノードに設定変更を適用できなくなります。MCP を一時停止することにより、**kube-apiserver-to-kubelet-signer** CA 証明書の自動 CA ローテーションを含め、自動的にローテーションされる証明書が関連付けられたノードにプッシュされないようにします。

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

MCP の一時停止は、**kube-apiserver-to-kubelet-signer** CA 証明書の有効期限を慎重に考慮して、短期間のみ行う必要があります。



注記

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

10.2. カナリアロールアウト更新プロセスおよび MCP について

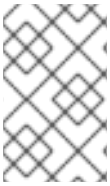
OpenShift Container Platform では、ノードは個別に考慮されません。代わりに、ノードはマシン設定プール (MCP) にグループ化されます。デフォルトでは、OpenShift Container Platform クラスター内のノードは 2 つの MCP にグループ化されます。1 つはコントロールプレーンノード用、もう 1 つはワーカーノード用です。OpenShift Container Platform の更新は、すべての MCP に同時に影響します。

更新中、Machine Config Operator (MCO) は、最大数が指定されている場合、指定された **maxUnavailable** ノード数まで MCP 内のすべてのノードをドレインし、遮断します。デフォルトでは、**maxUnavailable** は **1** に設定されます。ノードがドレイン (解放) および遮断し、ノード上のすべての Pod のスケジュールを解除し、ノードをスケジュール対象外としてマークします。

ノードがドレイン (解放) されると、Machine Config Daemon は新規マシン設定を適用します。これには、オペレーティングシステム (OS) の更新を含めることができます。OS を更新するには、ホストを再起動する必要があります。

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

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

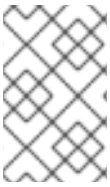


注記

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

カスタムマシン設定プールを使用する場合の考慮事項
ワークロードのデプロイメントトポロジーに基づいて、作成する MCP の数と各 MCP 内のノードの数を慎重に検討してください。たとえば、更新を特定のメンテナンス期間に合わせる必要がある場合は、OpenShift Container Platform が一定期間内に更新できるノードの数を把握しておく必要があります。この数は、それぞれのクラスターとワークロードの特性によって異なります。

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



注記

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

10.3. カナリアロールアウト更新の実行について

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

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

注記

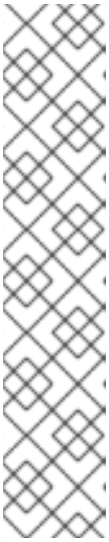
MCP の **maxUnavailable** 設定を変更して、任意の時点で更新できるパーセンテージまたはマシン数を指定できます。デフォルトは **1** です。

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

重要

ノードからデフォルトのワーカーラベルを削除しないでください。クラスター内でノードを適切に機能させるには、ノードにロールラベルが必要です。

3. 更新プロセスの一部として更新しない MCP を一時停止します。



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発生したときに、プールが一時停止されていないことを確認してください。MCP が一時停止されている場合、MCO は新しくローテーションされた証明書をそれらのノードにプッシュできません。これにより、クラスターが劣化し、**oc debug**、**oc logs**、**oc exec**、**oc attach** などの複数の **oc** コマンドで障害が発生します。証明書がローテーションされたときに MCP が一時停止された場合、OpenShift Container Platform コンソールのアラート UI でアラートを受け取ります。

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

5. 更新されたノードでアプリケーションをテストし、期待どおりに動作することを確認します。

6. 残りの MCP の 1 つを一時停止解除し、そのプール内のノードの更新が完了するのを待ち、それらのノードでアプリケーションをテストします。すべてのワーカーノードが更新されるまで、このプロセスを繰り返します。

7. オプション: 更新されたノードからカスタムラベルを削除し、カスタム MCP を削除します。

10.4. カナリアロールアウト更新を実行するためのマシン設定プールの作成

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

手順

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

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

出力例

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

2. 更新を遅らせるノードごとに、次のコマンドを実行してカスタムラベルをノードに追加します。

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

以下に例を示します。

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

-

出力例

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

3. 新規 MCP を作成します。

   a. MCP の YAML ファイルを作成します。

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

   (1) MCP の名前を指定します。

   (2) **worker** およびカスタム MCP 名を指定します。

   (3) このプールで必要なノードに追加したカスタムラベルを指定します。

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

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

   出力例

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

4. 次のコマンドを実行して、クラスター内の MCP のリストとその現在の状態を表示します。

```
$ oc get machineconfigpool
```

出力例

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

```
True      False      False      1          1          1          0          2m42s
worker    rendered-worker-87ba3dec1ad78cb6aecebf7fbb476a36      True       False
False     2          2          2          0          97m
```

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

10.5. マシン設定プールの一時停止

カスタムマシン設定プール (MCP) を作成したら、それらの MCP を一時停止します。MCP を一時停止にすると、Machine Config Operator (MCO) がその MCP に関連付けられたノードを更新できなくなります。

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発生したときに、プールが一時停止されていないことを確認してください。MCP が一時停止されている場合、MCO は新しくローテーションされた証明書をそれらのノードにプッシュできません。これにより、クラスターが劣化し、**oc debug**、**oc logs**、**oc exec**、**oc attach** などの複数の **oc** コマンドで障害が発生します。証明書がローテーションされたときに MCP が一時停止された場合、OpenShift Container Platform コンソールのアラート UI でアラートを受け取ります。

手順

1. 次のコマンドを実行して、一時停止する MCP にパッチを適用します。

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

以下に例を示します。

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

出力例

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

10.6. クラスターの更新の実行

マシン設定プール (MCP) が準備完了状態になったら、クラスターの更新を実行できます。クラスターに合わせて、以下の更新方法のいずれかを参照してください。

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

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

10.7. マシン設定プールの一時停止の解除

OpenShift Container Platform の更新が完了したら、カスタムマシン設定プール (MCP) の一時停止を 1 つずつ解除します。MCP の一時停止を解除すると、Machine Config Operator(MCO) はその MCP に関連付けられたノードを更新できます。

手順

1. 一時停止を解除する MCP にパッチを適用します。

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

以下に例を示します。

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

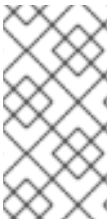
出力例

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

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

```
$ oc get machineconfigpools
```

3. 更新されたノードでアプリケーションをテストし、想定通りに機能していることを確認します。
4. 他の一時停止中の MCP についても、1 つずつこのプロセスを繰り返します。

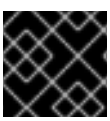


注記

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

10.8. ノードを元のマシン設定プールに移動する

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



重要

ノードには、クラスター内で適切に機能するロールが必要です。

手順

1. カスタム MCP 内のノードごとに、次のコマンドを実行してノードからカスタムラベルを削除します。

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

以下に例を示します。

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

出力例

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

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

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

```
$ oc get mcp
```

出力例

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

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

3. オプション: 次のコマンドを実行してカスタム MCP を削除します。

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

第11章 BOOTUPD を使用して RHCOS ノード上のブートローダーを更新する

bootupd を使用して RHCOS ノード上のブートローダーを更新するには、RHCOS マシン上で **bootupctl update** コマンドを手動で実行するか、**systemd** ユニットを使用してマシン設定を指定する必要があります。

grubby またはその他のブートローダーツールとは異なり、**bootupd** はカーネル引数を渡すなどのカーネル領域の設定を管理しません。カーネル引数を設定するには、ノードへのカーネル引数の追加 を参照し てください。

注記

bootupd を使用してブートローダーを更新すると、BootHole 脆弱性から保護できます。

11.1. ブートローダーを手動で更新する

bootupctl コマンドラインツールを使用して、システムのステータスを手動で検査し、ブートローダーを更新できます。

1. システムのステータスを検査します。

```
# bootupctl status
```

x86_64 の出力例

```
Component EFI
  Installed: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
  Update: At latest version
```

aarch64 の出力例

```
Component EFI
  Installed: grub2-efi-aa64-1:2.02-99.el8_4.1.aarch64,shim-aa64-15.4-2.el8_1.aarch64
  Update: At latest version
```

2. 最初にバージョン 4.4 以前にインストールされた OpenShift Container Platform クラスターには、明示的な導入フェーズが必要です。
システムのステータスが **Adoptable** の場合に、導入を実行します。

```
# bootupctl adopt-and-update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
```

3. 更新が利用可能な場合は、更新を適用して、次回の再起動時に変更が有効になるようにします。

```
# bootupctl update
```

出力例

```
Updated: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
```

11.2. マシン設定を通してブートローダーを自動更新する

bootupd を使用してブートローダーを自動更新するもう 1 つの方法は、必要に応じて起動するたびにブートローダーを更新する systemd サービスユニットを作成することです。このユニットは、ブートプロセス中に **bootupctl update** コマンドを実行し、マシン設定を通してノードにインストールされます。

注記

更新操作が予期せず中断されるとノードが起動不能になる可能性があるため、この設定はデフォルトでは有効になっていません。この設定を有効にする場合は、ブートローダーを更新する間、ブートプロセス中にノードが中断されないように注意してください。通常、ブートローダーの更新操作はすぐに完了するため、リスクは低くなります。

1. **bootupctl-update.service** systemd ユニットの内容を含む Butane 設定ファイル **99-worker-bootupctl-update.bu** を作成します。

注記

Butane の詳細は、「Butane を使用したマシン設定の作成」を参照してください。

出力例

```
variant: openshift
version: 4.12.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
systemd:
  units:
  - name: bootupctl-update.service
    enabled: true
    contents: |
      [Unit]
      Description=Bootupd automatic update

      [Service]
      ExecStart=/usr/bin/bootupctl update
      RemainAfterExit=yes

      [Install]
      WantedBy=multi-user.target
```

(1) (2) コントロールプレーンノードでは、これらの両方の場所で **worker** の代わりに **master** を使用します。

2. Butane を使用して、ノードに配信される設定を含む **MachineConfig** オブジェクトファイル (**99-worker-bootupctl-update.yaml**) を生成します。

```
$ butane 99-worker-bootupctl-update.bu -o 99-worker-bootupctl-update.yaml
```

3. 以下の 2 つの方法のいずれかで設定を適用します。

   - クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、**MachineConfig** オブジェクトファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を続行します。
   - クラスターがすでに実行中の場合は、ファイルを適用します。

```
$ oc apply -f ./99-worker-bootupctl-update.yaml
```

第12章 RHEL コンピュートマシンを含むクラスターの更新

OpenShift Container Platform クラスターでマイナーバージョンおよびパッチの更新を実行できます。クラスターに Red Hat Enterprise Linux (RHEL) マシンが含まれる場合は、それらのマシンを更新するために追加の手順を実行する必要があります。

12.1. 前提条件

- **admin** 権限を持つユーザーとしてクラスターにアクセスできる。RBAC の使用によるパーミッションの定義および適用 を参照してください。
- 更新が失敗し、クラスターを以前の状態に復元する 必要がある場合に、最新の etcd バックアップ がある こと。
- RHEL7 ワーカーのサポートは OpenShift Container Platform 4.12 では削除されています。OpenShift Container Platform 4.12 にアップグレードする前に、RHEL7 ワーカーを RHEL8 または RHCOS ワーカーに置き換える必要があります。Red Hat は、RHEL ワーカーの RHEL7 から RHEL8 のインプレース更新をサポートしません。このホストは、クリーンなオペレーティングシステムインストールに置き換える必要があります。
- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは 手動で維持された認証情報でクラスターを更新する準備 を参照してください。
- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合、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

12.2. 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。

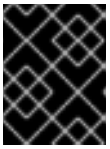
前提条件

- **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、**チャネル** が、**stable-4.12** など、更新するバージョンの正しいチャネルに設定されていることを確認してください。



重要

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

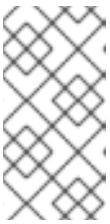
注記

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

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

- Update Status が Updates Available ではない場合、クラスターを更新することはできません。
- Select Channel は、クラスターが実行されているか、更新されるクラスターのバージョンを示します。

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

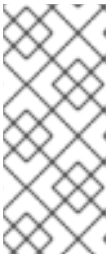


注記

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

4. 更新が完了し、Cluster Version Operator が利用可能な更新を更新したら、追加の更新が現在のチャネルで利用可能かどうかを確認します。
   - 更新が利用可能な場合は、更新ができなくなるまで、現在のチャネルでの更新を継続します。
   - 利用可能な更新がない場合は、**チャネル** を次のマイナーバージョンの **stable-***、**eus-*** または **fast-*** チャネルに変更し、そのチャネルで必要なバージョンに更新します。

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



注記

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

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

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

12.3.1. アップグレード用の Ansible Hook について

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

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

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

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

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

前提条件

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

手順

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

12.3.3. RHEL コンピュートマシンで利用できるフック

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

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

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

12.4. クラスター内の RHEL コンピュートマシンの更新

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

重要

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

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

重要

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

前提条件

- クラスターが更新されていること。

重要

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

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

手順

1. ホストで firewalld を停止し、無効にします。

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

注記

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

2. OpenShift Container Platform 4.12 で必要なリポジトリーを有効にします。

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

```
# subscription-manager repos --disable=rhocp-4.11-for-rhel-8-x86_64-rpms \
                           --disable=ansible-2.9-for-rhel-8-x86_64-rpms \
                           --enable=rhocp-4.12-for-rhel-8-x86_64-rpms
```

重要

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

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

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

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

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

d. 各 RHEL コンピュートノードで、必要なリポジトリーを更新します。

```
# subscription-manager repos --disable=rhocp-4.11-for-rhel-8-x86_64-rpms \
                            --enable=rhocp-4.12-for-rhel-8-x86_64-rpms
```

3. RHEL ワーカーマシンを更新します。

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

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

openshift_kubeconfig_path="~/.kube/config"

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

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

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

c. **upgrade** Playbook を実行します。

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

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

注記

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

4. 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   v1.25.0
mycluster-control-plane-1   Ready                      master   145m   v1.25.0
mycluster-control-plane-2   Ready                      master   145m   v1.25.0
mycluster-rhel8-0           Ready                      worker   98m    v1.25.0
```

```
mycluster-rhel8-1   Ready   worker   98m   v1.25.0
mycluster-rhel8-2   Ready   worker   98m   v1.25.0
mycluster-rhel8-3   Ready   worker   98m   v1.25.0
```

5. 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。4.12 にないパッケージを更新するには、以下のコマンドを使用します。

```
# yum update
```

注記

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

第13章 非接続環境でのクラスターの更新

13.1. 非接続環境でのクラスターの更新について

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

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

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

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

13.1.2. 非接続環境でのクラスターの更新の実行

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

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

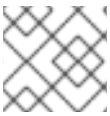
13.1.3. クラスターからの OpenShift Update Service のアンインストール

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

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

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

非接続環境でクラスターを更新する前に、コンテナー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ーイメージのみをクラスターで実行するようにすることもできます。



注記

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

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

1. リリースイメージの取得およびプッシュに使用されるすべてのデバイスに OpenShift CLI (**oc**) をインストールします。

2. レジストリープルシークレットをダウンロードし、クラスターに追加します。

3. oc-mirror OpenShift CLI (**oc**) プラグイン を使用する場合:

    a. リリースイメージの取得およびプッシュに使用されるすべてのデバイスに oc-mirror プラグインをインストールします。

    b. ミラーリングするリリースイメージを決定する際に、使用するプラグイン用のイメージセット設定ファイルを作成します。この設定ファイルは後で編集して、プラグインがミラーリングするリリースイメージを変更できます。

    c. ターゲットのリリースイメージをミラーレジストリーに直接ミラーリングするかリムーバブルメディアにミラーリングしてからミラーレジストリーにミラーリングします。

    d. oc-mirror プラグインが生成したリソースを使用するようにクラスターを設定します。

    e. 必要に応じてこれらの手順を繰り返し、ミラーレジストリーを更新します。

4. **oc adm release mirror** コマンド を使用する場合:

    a. 使用している環境とミラーリングするリリースイメージに対応する環境変数を設定します。

    b. ターゲットのリリースイメージをミラーレジストリーに直接ミラーリングするかリムーバブルメディアにミラーリングしてからミラーレジストリーにミラーリングします。

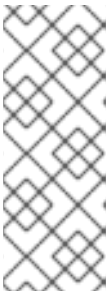
    c. 必要に応じてこれらの手順を繰り返し、ミラーレジストリーを更新します。

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

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

13.2.1. 前提条件

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



注記

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

コンテナーイメージレジストリーの既存のソリューションがない場合、Red Hat Openshift 導入用のミラーレジストリー は OpenShift Container Platform サブスクリプションに含まれます。**Red Hat OpenShift のミラーレジストリー** は、非接続インストールおよび更新で

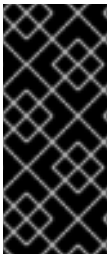
OpenShift Container Platform コンテナーイメージをミラーリングするために使用できる小規模なコンテナーレジストリーです。

13.2.2. 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。

13.2.2.1. 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 4.12 のすべてのコマンドを実行することはできません。新しいバージョンの **oc** をダウンロードしてインストールしてください。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の OpenShift Container Platform ダウンロードページ に移動します。
2. **Product Variant** ドロップダウンリストからアーキテクチャーを選択します。
3. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
4. **OpenShift v4.12 Linux Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。
5. アーカイブを展開します。

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

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

```
$ echo $PATH
```

検証

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

```
$ oc <command>
```

Windows への OpenShift CLI のインストール

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.12 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
```

検証

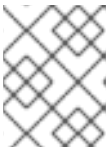
- OpenShift CLI のインストール後に、**oc** コマンドを使用して利用できます。

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

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

手順

1. Red Hat カスタマーポータルの OpenShift Container Platform ダウンロードページ に移動します。
2. **バージョン** ドロップダウンリストから適切なバージョンを選択します。
3. **OpenShift v4.12 macOS Client** エントリーの横にある **Download Now** をクリックして、ファイルを保存します。



注記

macOS arm64 の場合は、**OpenShift v4.12 macOS arm64 Client** エントリーを選択します。

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

```
$ echo $PATH
```

検証

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

```
$ oc <command>
```

関連情報

- CLI プラグインのインストールおよび使用

13.2.2.2. 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ーイメージレジストリーの認証情報ファイルを作成します。

警告

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

警告

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

前提条件

- 非接続環境で使用するミラーレジストリーを設定しました。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。
- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから **registry.redhat.io** プルシークレットをダウンロードします。
2. JSON 形式でプルシークレットのコピーを作成します。

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

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

ファイルの内容は以下の例のようになります。

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

3. オプション: oc-mirror プラグインを使用している場合は、ファイルを ~/**.docker/config.json** または **$XDG_RUNTIME_DIR/containers/auth.json** として保存します。

   a. **.docker** または **$XDG_RUNTIME_DIR/containers** ディレクトリーが存在しない場合は、以下のコマンドを入力して作成します。

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

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

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

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

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

4. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

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

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

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

```
"auths": {
  "<mirror_registry>": { 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": "BGVtbYk3ZHAtqXs=",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

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

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

13.2.3.1. oc-mirror プラグインについて

oc-mirror OpenShift CLI(**oc**) プラグインを使用すると、単一のツールを使用して、必要なすべての OpenShift Container Platform コンテンツおよびその他のイメージをミラーレジストリーにミラーリングできます。次の機能を提供します。

- OpenShift Container Platform のリリース、Operator、ヘルムチャート、およびその他のイメージをミラーリングするための一元化された方法を提供します。
- OpenShift Container Platform および Operator の更新パスを維持します。
- 宣言型イメージセット設定ファイルを使用して、クラスターに必要な OpenShift Container Platform リリース、Operator、およびイメージのみを含めます。
- 将来のイメージセットのサイズを縮小するインクリメンタルミラーリングを実行します。
- 前回の実行以降にイメージセット設定から除外されたターゲットミラーレジストリーからのイメージをプルーニングします。
- オプションで、OpenShift Update Service (OSUS) を使用する際のサポートアーティファクトを生成します。

oc-mirror プラグインを使用する場合、イメージセット設定ファイルでミラーリングするコンテンツを指定します。この YAML ファイルでは、クラスターに必要な OpenShift Container Platform リリースと Operator のみを含めるように設定を微調整できます。これにより、ダウンロードして転送する必要のあるデータの量が減ります。oc-mirror プラグインは、任意のヘルムチャートと追加のコンテナーイメージをミラーリングして、ユーザーがワークロードをミラーレジストリーにシームレスに同期できるようにすることもできます。

oc-mirror プラグインを初めて実行すると、非接続クラスターのインストールまたは更新を実行するために必要なコンテンツがミラーレジストリーに入力されます。非接続クラスターが更新を受信し続けるには、ミラーレジストリーを更新しておく必要があります。ミラーレジストリーを更新するには、最初に実行したときと同じ設定を使用して oc-mirror プラグインを実行します。oc-mirror プラグインは、ストレージバックエンドからメタデータを参照し、ツールを最後に実行してからリリースされたもののみをダウンロードします。これにより、OpenShift Container Platform および Operator の更新パスが提供され、必要に応じて依存関係の解決が実行されます。

重要

oc-mirror CLI プラグインを使用してミラーレジストリーにデータを入力する場合、ミラーレジストリーをさらに更新するには、oc-mirror ツールを使用する必要があります。

13.2.3.2. oc-mirror の互換性とサポート

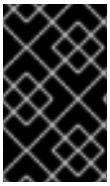
oc-mirror プラグインは、OpenShift Container Platform バージョン 4.9 以降の OpenShift Container Platform ペイロードイメージと Operator カタログのミラーリングをサポートします。

ミラーリングする必要がある OpenShift Container Platform のバージョンに関係なく、使用可能な最新バージョンの oc-mirror プラグインを使用してください。

13.2.3.3. ミラーレジストリーについて

OpenShift Container Platform のインストールとその後の製品更新に必要なイメージを、Red Hat Quay などの Docker v2-2 をサポートするコンテナーミラーレジストリーにミラーリングできます。大規模なコンテナーレジストリーにアクセスできない場合は、OpenShift Container Platform サブスクリプションに含まれる小規模なコンテナーレジストリーである**Red Hat Openshift 導入用のミラーレジストリー**を使用できます。

選択したレジストリーに関係なく、インターネット上の Red Hat がホストするサイトから分離されたイメージレジストリーにコンテンツをミラーリングする手順は同じです。コンテンツをミラーリングした後に、各クラスターをミラーレジストリーからこのコンテンツを取得するように設定します。



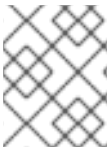
重要

OpenShift イメージレジストリーはターゲットレジストリーとして使用できません。これは、ミラーリングプロセスで必要となるタグを使わないプッシュをサポートしないためです。

Red Hat OpenShift 導入用のミラーレジストリー 以外のコンテナーレジストリーを選択する場合は、プロビジョニングするクラスター内の全マシンから到達可能である必要があります。レジストリーに到達できない場合、インストール、更新、またはワークロードの再配置などの通常の操作が失敗する可能性があります。そのため、ミラーレジストリーは可用性の高い方法で実行し、ミラーレジストリーは少なくとも OpenShift Container Platform クラスターの実稼働環境の可用性の条件に一致している必要があります。

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

ミラーリングされたレジストリーの場合は、プルされたイメージのソースを表示するには、CRI-O ログで **Trying to access** のログエントリーを確認する必要があります。ノードで **crictl images** コマンドを使用するなど、イメージのプルソースを表示する他の方法では、イメージがミラーリングされた場所からプルされている場合でも、ミラーリングされていないイメージ名を表示します。



注記

Red Hat は、OpenShift Container Platform を使用してサードパーティーのレジストリーをテストしません。

関連情報

- CRI-O ログを表示してイメージソースを表示する方法の詳細は、Viewing the image pull source を参照してください。

13.2.3.4. oc-mirror OpenShift CLI プラグインのインストール

oc-mirror OpenShift CLI プラグインを使用してレジストリーイメージをミラーリングするには、プラグインをインストールする必要があります。完全な非接続環境でイメージセットをミラーリングする場合は、インターネットにアクセスできるホストと、ミラーレジストリーにアクセスできる非接続環境のホストに oc-mirror プラグインをインストールしてください。

前提条件

- OpenShift CLI (**oc**) がインストールされている。

手順

1. oc-mirror CLI プラグインをダウンロードします。
   a. OpenShift Cluster Manager Hybrid Cloud Console の ダウンロード ページに移動します。
   b. **OpenShift 切断インストールツール** セクションで、**OpenShift Client (oc) ミラープラグイン** の **ダウンロード** をクリックしてファイルを保存します。

2. アーカイブを抽出します。

```
$ tar xvzf oc-mirror.tar.gz
```

3. 必要に応じて、プラグインファイルを更新して実行可能にします。

```
$ chmod +x oc-mirror
```

注記

oc-mirror ファイルの名前を変更しないでください。

4. ファイルを **PATH** に配置して、oc-mirror CLI プラグインをインストールします (例: **/usr/local/bin**):。

```
$ sudo mv oc-mirror /usr/local/bin/.
```

検証

- **oc mirror help** を実行して、プラグインが正常にインストールされたことを確認します。

```
$ oc mirror help
```

13.2.3.5. イメージセット設定の作成

oc-mirror プラグインを使用してイメージセットをミラーリングする前に、イメージセット設定ファイルを作成する必要があります。このイメージセット設定ファイルは、ミラーリングする OpenShift Container Platform リリース、Operator、およびその他のイメージと、oc-mirror プラグインの他の設定を定義します。

イメージセット設定ファイルでストレージバックエンドを指定する必要があります。このストレージバックエンドは、 Docker v2-2 をサポートするローカルディレクトリーまたはレジストリーにすることができます。oc-mirror プラグインは、イメージセットの作成中にこのストレージバックエンドにメタデータを保存します。

重要

oc-mirror プラグインによって生成されたメタデータを削除または変更しないでください。同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、同じストレージバックエンドを使用する必要があります。

前提条件

- コンテナーイメージレジストリーの認証情報ファイルを作成している。手順については、**イメージのミラーリングを可能にする認証情報の設定** を参照してください。

手順

1. **oc mirror init** コマンドを使用して、イメージセット設定のテンプレートを作成し、それを **imageset-config.yaml** というファイルに保存します。

```
$ oc mirror init --registry example.com/mirror/oc-mirror-metadata > imageset-config.yaml 1
```

1. **example.com/mirror/oc-mirror-metadata** をストレージバックエンドのレジストリーの場所に置き換えます。

2. ファイルを編集し、必要に応じて設定を調整します。

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
archiveSize: 4 (1)
storageConfig: (2)
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata (3)
    skipTLS: false
mirror:
  platform:
    channels:
    - name: stable-4.12 (4)
      type: ocp
    graph: true (5)
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12 (6)
    packages:
    - name: serverless-operator (7)
      channels:
      - name: stable (8)
  additionalImages:
  - name: registry.redhat.io/ubi8/ubi:latest (9)
  helm: {}
```

1. **archiveSize** を追加して、イメージセット内の各ファイルの最大サイズを GiB 単位で設定します。

2. イメージセットのメタデータを保存するバックエンドの場所を設定します。この場所は、レジストリーまたはローカルディレクトリーにすることができます。Technology Preview OCI 機能を使用していない場合は、**storageConfig** 値を指定する必要があります。

3. ストレージバックエンドのレジストリー URL を設定します。

4. OpenShift Container Platform イメージを取得するためのチャネルを設定します。

5. **graph: true** を追加して、グラフデータイメージをビルドし、ミラーレジストリーにプッシュします。OpenShift Update Service (OSUS) を作成するには、graph-data イメージが必要です。**graph: true** フィールドは **UpdateService** カスタムリソースマニフェストも生成します。**oc** コマンドラインインターフェイス (CLI) は、**UpdateService** カスタムリソースマニフェストを使用して OSUS を作成できます。詳細については、**OpenShift Update Service について** を参照してください。

6. OpenShift Container Platform イメージを取得するための Operator カタログを設定します。

7. イメージセットに含める特定の Operator パッケージのみを指定します。カタログ内のすべてのパッケージを取得するには、このフィールドを削除してください。

8.

イメージセットに含める Operator パッケージの特定のチャネルのみを指定します。そのチャネルでバンドルを使用しない場合も、常に Operator パッケージのデフォルトチャネ

9 イメージセットに含める追加のイメージを指定します。

注記

graph: true フィールド は、他のミラーリングされたイメージとともに **ubi-micro** イメージもミラーリングします。

パラメーターの完全なリストについては、**イメージセットの設定パラメーター** を参照してください。また、さまざまなミラーリングのユースケースについては、**イメージセットの設定例** を参照してください。

3. 更新したファイルを保存します。
このイメージセット設定ファイルは、コンテンツをミラーリングするときに **oc mirror** コマンドで必要になります。

関連情報

- Image set configuration parameters
- Image set configuration examples
- OpenShift Update Service について

13.2.3.6. イメージセットをミラーレジストリーにミラーリングする

oc-mirror CLI プラグインを使用して、部分的な非接続環境 または 完全な非接続環境 でイメージをミラーレジストリーにミラーリングできます。

以下の手順は、ミラーレジストリーがすでに設定されていることを前提としています。

13.2.3.6.1. 部分的な非接続環境でのイメージセットのミラーリング

部分的な非接続環境では、イメージセットをターゲットミラーレジストリーに直接ミラーリングできます。

13.2.3.6.1.1. ミラーからミラーへのミラーリング

oc-mirror プラグインを使用して、イメージセットの作成中にアクセス可能なターゲットミラーレジストリーにイメージセットを直接ミラーリングできます。

イメージセット設定ファイルでストレージバックエンドを指定する必要があります。このストレージバックエンドは、ローカルディレクトリーまたは Dockerv2 レジストリーにすることができます。oc-mirror プラグインは、イメージセットの作成中にこのストレージバックエンドにメタデータを保存します。

重要

oc-mirror プラグインによって生成されたメタデータを削除または変更しないでください。同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、同じストレージバックエンドを使用する必要があります。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。
- イメージセット設定ファイルを作成している。

手順

- **oc mirror** コマンドを実行して、指定されたイメージセット設定から指定されたレジストリーにイメージをミラーリングします。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  docker://registry.example:5000 2
```

1. 作成されたイメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. イメージセットファイルをミラーリングするレジストリーを指定します。レジストリーは **docker://** で始まる必要があります。ミラーレジストリーに最上位の namespace を指定する場合は、これ以降の実行でもこれと同じ namespace を使用する必要があります。

検証

1. 生成された **oc-mirror-workspace/** ディレクトリーに移動します。
2. results ディレクトリーに移動します (例: **results-1639608409/**。
3. **ImageContentSourcePolicy** および **CatalogSource** リソースに YAML ファイルが存在することを確認します。

次のステップ

- oc-mirror が生成したリソースを使用するようにクラスターを設定します。

トラブルシューティング

- Unable to retrieve source image .

13.2.3.6.2. 完全な非接続環境でのイメージセットのミラーリング

完全な非接続環境でイメージセットをミラーリングするには、最初に イメージセットをディスクにミラーリング してから、ディスク上のイメージセットファイルをミラーにミラーリング する必要があります。

13.2.3.6.2.1. ミラーからディスクへのミラーリング

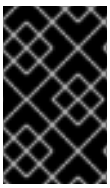
oc-mirror プラグインを使用して、イメージセットを生成し、コンテンツをディスクに保存できます。生成されたイメージセットは、非接続環境に転送され、ターゲットレジストリーにミラーリングされます。

重要

イメージセット設定ファイルで指定されている設定によっては、oc-mirror を使用してイメージをミラーリングすると、数百ギガバイトのデータがディスクにダウンロードされる場合があります。

多くの場合、ミラーレジストリーにデータを入力するときの最初のイメージセットのダウンロードが、最も大きなものとなります。最後にコマンドを実行した後に変更されたイメージのみをダウンロードするため、oc-mirror プラグインを再度実行すると、生成されるイメージセットは小さいことが多いです。

イメージセット設定ファイルでストレージバックエンドを指定する必要があります。このストレージバックエンドは、ローカルディレクトリーまたは docker v2 レジストリーにすることができます。oc-mirror プラグインは、イメージセットの作成中にこのストレージバックエンドにメタデータを保存します。



重要

oc-mirror プラグインによって生成されたメタデータを削除または変更しないでください。同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、同じストレージバックエンドを使用する必要があります。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。
- イメージセット設定ファイルを作成している。

手順

- **oc mirror** コマンドを実行して、指定されたイメージセット設定からディスクにイメージをミラーリングします。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  file://<path_to_output_directory>           2
```

1. 作成されたイメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. イメージセットファイルを出力するターゲットディレクトリーを指定します。ターゲットディレクトリーのパスは、**file://** で始まる必要があります。

検証

1. 出力ディレクトリーに移動します。

```
$ cd <path_to_output_directory>
```

2. イメージセットの **.tar** ファイルが作成されたことを確認します。

```
$ ls
```

出力例

```
mirror_seq1_000000.tar
```

次のステップ

- イメージセットの.tar ファイルを非接続環境に転送します。

トラブルシューティング

- Unable to retrieve source image .

13.2.3.6.2.2. ディスクからミラーへのミラーリング

oc-mirror プラグインを使用して、生成されたイメージセットの内容をターゲットミラーレジストリーにミラーリングできます。

前提条件

- 非接続環境に OpenShift CLI (**oc**) をインストールしている。
- 非接続環境に **oc-mirror** CLI プラグインをインストールしている。
- **ocmirror** コマンドを使用してイメージセットファイルを生成している。
- イメージセットファイルを非接続環境に転送しました。

手順

- **oc mirror** コマンドを実行して、ディスク上のイメージセットファイルを処理し、その内容をターゲットミラーレジストリーにミラーリングします。

```
$ oc mirror --from=./mirror_seq1_000000.tar \ 1
  docker://registry.example:5000              2
```

1. この例では、**mirror_seq1_000000.tar** という名前のイメージセット.tar ファイルをミラーに渡します。イメージセット設定ファイルで **archiveSize** 値が指定されている場合、イメージセットは複数の.tar ファイルに分割される可能性があります。この状況では、イメージセットの.tar ファイルを含むディレクトリーを渡すことができます。
2. イメージセットファイルをミラーリングするレジストリーを指定します。レジストリーは **docker://** で始まる必要があります。ミラーレジストリーに最上位の namespace を指定する場合は、これ以降の実行でもこれと同じ namespace を使用する必要があります。

このコマンドは、ミラーレジストリーをイメージセットで更新し、**ImageContentSourcePolicy** および **CatalogSource** リソースを生成します。

検証

1. 生成された **oc-mirror-workspace/** ディレクトリーに移動します。

2. results ディレクトリーに移動します (例: **results-1639608409**/。
3. **ImageContentSourcePolicy** および **CatalogSource** リソースに YAML ファイルが存在することを確認します。

次のステップ

- oc-mirror が生成したリソースを使用するようにクラスターを設定します。

トラブルシューティング

- Unable to retrieve source image .

13.2.3.7. oc-mirror が生成したリソースを使用するためのクラスター設定

イメージセットをミラーレジストリーにミラーリングした後に、生成された **ImageContentSourcePolicy**、**CatalogSource**、およびリリースイメージの署名リソースをクラスターに適用する必要があります。

ImageContentSourcePolicy リソースは、ミラーレジストリーをソースレジストリーに関連付け、イメージプル要求をオンラインレジストリーからミラーレジストリーにリダイレクトします。**CatalogSource** リソースは、Operator Lifecycle Manager (OLM) によって使用され、ミラーレジストリーで使用可能な Operator に関する情報を取得します。リリースイメージの署名は、ミラーリングされたリリースイメージの検証に使用されます。

前提条件

- 非接続環境で、イメージセットをレジストリーミラーにミラーリングしました。
- **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. **cluster-admin** ロールを持つユーザーとして OpenShift CLI にログインします。
2. 以下のコマンドを実行して、results ディレクトリーからクラスターに YAML ファイルを適用します。

```
$ oc apply -f ./oc-mirror-workspace/results-1639608409/
```

3. リリースイメージをミラーリングした場合は、次のコマンドを実行して、リリースイメージの署名をクラスターに適用します。

```
$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/
```

注記

クラスターではなく Operator をミラーリングしている場合、**$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/** を実行する必要はありません。適用するリリースイメージ署名がないため、このコマンドを実行するとエラーが返されます。

検証

1. 以下のコマンドを実行して、**ImageContentSourcePolicy** リソースが正常にインストールされたことを確認します。

```
$ oc get imagecontentsourcepolicy
```

2. 以下のコマンドを実行して、**CatalogSource** リソースが正常にインストールされたことを確認します。

```
$ oc get catalogsource -n openshift-marketplace
```

13.2.3.8. ミラーレジストリーのコンテンツを最新の状態に保つ

ターゲットミラーレジストリーに初期イメージセットを入力したら、最新のコンテンツが含まれるように定期的に更新する必要があります。可能な場合は、定期的にミラーレジストリーを更新する cron ジョブを設定できます。

イメージセット設定を更新して、必要に応じて OpenShift Container Platform および Operator リリースを追加または削除してください。削除したイメージはミラーレジストリーからプルーニングされます。

13.2.3.8.1. ミラーレジストリーコンテンツの更新について

oc-mirror プラグインを再度実行すると、前回の実行以降に新しく更新されたイメージのみを含むイメージセットが生成されます。前に作成されたイメージセットとの違いのみを取り込むため、生成されたイメージセットは、多くの場合、最初のイメージセットよりも小さく、迅速に処理されます。

重要

生成されたイメージセットはシーケンシャルであり、ターゲットミラーレジストリーに順番にプッシュする必要があります。シーケンス番号は、生成されたイメージセットアーカイブファイルのファイル名から取得できます。

新規イメージおよび更新されたイメージの追加
イメージセット設定の設定に応じて、oc-mirror を今後実行すると、追加の新しいイメージと更新されたイメージがミラーリングされます。イメージセット設定の設定を確認して、必要に応じて新しいバージョンを取得していることを確認します。たとえば、特定のバージョンに制限する場合は、Operator の最小バージョンと最大バージョンをミラーリングするように設定できます。または、最小バージョンをミラーリングの開始点として設定することもできますが、バージョン範囲は開いたままにして、oc-mirror の今後の実行時に新しい Operator バージョンを受け取り続けることができます。最小または最大バージョンを省略すると、チャネル内の Operator の完全なバージョン履歴が得られます。明示的に名前付けされたチャネルを省略すると、指定された Operator のすべてのチャネルのすべてのリリースが提供されます。名前付き Operator を省略すると、これまでにリリースされたすべての Operator とそのすべてのバージョンのカタログ全体が提供されます。

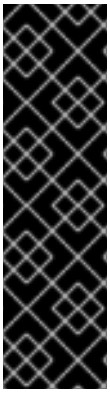
これらすべての制約と条件は、oc-mirror が呼び出されるたびに Red Hat によって公開されたコンテンツに対して評価されます。このようにして、新しいリリースとまったく新しい Operator を自動的にピックアップします。制約は、必要な Operator のセットをリストするだけで指定できます。これにより、新しくリリースされた他の Operator がミラーセットに自動的に追加されることはありません。特定のリリースチャネルを指定することもできます。これにより、ミラーリングは追加された新しいチャネルではなく、このチャネルのみに制限されます。これは、マイナーリリースに異なるリリースチャネルを使用する Red Hat Quay などの Operator 製品にとって重要です。最後に、特定の Operator の最大バージョンを指定できます。これにより、ツールは指定されたバージョン範囲のみをミラーリングするため、ミラーリングされた最大バージョンを超えた新しいリリースが自動的に取得されることはありま

せん。これらのすべての場合において、イメージセット設定ファイルを更新して Operator のミラーリングの範囲を広げ、他の Operator、新しいチャネル、および Operator の新しいバージョンをターゲットレジストリーで使用できるようにする必要があります。

チャネル仕様やバージョン範囲などの制約を、特定の Operator が選択したリリースストラテジーに合わせることを推奨します。たとえば、Operator が **stable** チャネルを使用している場合、ミラーリングをそのチャネルと可能ならば最小バージョンに制限し、ダウンロード量と定期的な安定した更新の取得との間の適切なバランスを見つける必要があります。Operator がリリースバージョンのチャネルスキーム (**stable-3.7** など) を選択した場合、そのチャネルのすべてのリリースをミラーリングする必要があります。これにより、Operator のパッチバージョン (**3.7.1** など) を引き続き受け取ることができます。また、イメージセットの設定を定期的に調整して、新製品リリース (**stable-3.8** など) 用のチャネルを追加することもできます。

イメージのプルーニング
イメージは、生成およびミラーリングされた最新のイメージセットに含まれなくなった場合、ターゲットミラーレジストリーから自動的にプルーニングされます。これにより、不要なコンテンツを簡単に管理およびクリーンアップし、ストレージリソースを解放することができます。

不要になった OpenShift Container Platform リリースまたは Operator バージョンがある場合、イメージセットの設定を変更してそれらを除外できます。これらはミラーリング時にミラーレジストリーからプルーニングされます。これは、イメージセット設定ファイルで Operator ごとに最小または最大バージョン範囲の設定を調整するか、カタログからミラーリングする Operator のリストから Operator を削除することによって実行できます。Operator カタログ全体または OpenShift Container Platform リリース全体を設定ファイルから削除することもできます。



重要

以下の状況では、イメージはターゲットミラーレジストリーから自動的にプルーニングされません。

- ミラーリングする新規イメージまたは更新されたイメージがない場合
- テクノロジープレビュー OCI 機能を使用している場合

さらに、Operator パブリッシャーがチャネルから Operator バージョンを削除すると、削除されたバージョンはターゲットミラーレジストリーからプルーニングされます。

ターゲットミラーレジストリーからのイメージの自動プルーニングを無効にするには、**--skip-pruning** フラグを **oc mirror** コマンドに渡します。

13.2.3.8.2. ミラーレジストリーコンテンツの更新

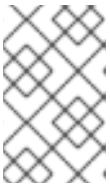
初期イメージセットをミラーレジストリーに公開した後、oc-mirror プラグインを使用して、切断されたクラスターを最新の状態に保つことができます。

イメージセットの設定に応じて、oc-mirror は、初期ミラーリングの完了後にリリースされた OpenShift Container Platform および選択した Operator の新しいリリースを自動的に検出します。たとえば、毎晩の cron ジョブなどで、定期的に oc-mirror を実行し、製品とセキュリティーの更新をタイムリーに受信することを推奨します。

前提条件

- oc-mirror プラグインを使用して、最初のイメージセットをミラーレジストリーにミラーリングしている。

- oc-mirror プラグインの最初の実行に使用されたストレージバックエンドにアクセスできる。



注記

同じミラーレジストリーに対して oc-mirror の最初の実行と同じストレージバックエンドを使用する必要があります。oc-mirror プラグインによって生成されたメタデータイメージを削除または変更しないでください。

手順

1. 必要に応じて、イメージセット設定ファイルを更新して、新しい OpenShift Container Platform および Operator バージョンを取得します。ミラーリングの使用例については、**イメージセットの設定例** を参照してください。

2. 初期イメージセットをミラーレジストリーにミラーリングしたときと同じ手順に従います。手順については、**部分的な非接続環境でのイメージセットのミラーリング** または **完全な非接続環境でのイメージセットのミラーリング** を参照してください。

重要

- 差分イメージセットのみが作成およびミラーリングされるように、同じストレージバックエンドを提供する必要があります。
- イメージセットの最初の作成時にミラーレジストリーにトップレベルの namespace を指定した場合は、同じミラーレジストリーに対して oc-mirror プラグインを実行するたびに、この同じ namespace を使用する必要があります。

3. oc-mirror が生成したリソースを使用するようにクラスターを設定します。

関連情報

- Image set configuration examples
- 部分的な非接続環境でのイメージセットのミラーリング
- 完全な非接続環境でのイメージセットのミラーリング
- oc-mirror が生成したリソースを使用するためのクラスター設定

13.2.3.9. ドライランの実行

実際にイメージをミラーリングせずに、oc-mirror を使用してドライランを実行できます。これにより、ミラーリングされるイメージのリストと、ミラーレジストリーからプルーニングされるイメージを確認できます。また、イメージセット設定のエラーを早期に検出したり、生成されたイメージのリストを他のツールで使用してミラーリング操作を実行したりすることもできます。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。

- イメージセット設定ファイルを作成している。

手順

1. **--dry-run** フラグを指定して **oc mirror** コマンドを実行し、ドライランを実行します。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  docker://registry.example:5000           \ 2
  --dry-run                                  3
```

1. 作成されたイメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. ミラーレジストリーを指定します。**--dry-run** フラグを使用している限り、このレジストリーには何もミラーリングされません。
3. **--dry-run** フラグを使用して、実際のイメージセットファイルではなく、ドライランアーティファクトを生成します。

出力例

```
Checking push permissions for registry.example:5000
Creating directory: oc-mirror-workspace/src/publish
Creating directory: oc-mirror-workspace/src/v2
Creating directory: oc-mirror-workspace/src/charts
Creating directory: oc-mirror-workspace/src/release-signatures
No metadata detected, creating new workspace
wrote mirroring manifests to oc-mirror-workspace/operators.1658342351/manifests-redhat-operator-index

...

info: Planning completed in 31.48s
info: Dry run complete
Writing image mapping to oc-mirror-workspace/mapping.txt
```

2. 生成されたワークスペースディレクトリーに移動します。

```
$ cd oc-mirror-workspace/
```

3. 生成された **mapping.txt** ファイルを確認します。
このファイルには、ミラーリングされるすべてのイメージのリストが含まれています。

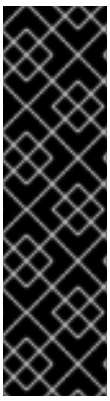
4. 生成された **pruning-plan.json** ファイルを確認します。
このファイルには、イメージセットの公開時にミラーレジストリーからプルーニングされるすべてのイメージのリストが含まれています。

注記

pruning-plan.json ファイルは、oc-mirror コマンドがミラーレジストリーを指し、プルーニングするイメージがある場合にのみ生成されます。

13.2.3.10. OCI 形式でのファイルベースのカタログ Operator イメージのミラーリング

oc-mirror プラグインを使用して、Docker v2 形式ではなく Open Container Initiative (OCI) イメージ形式で Operator をミラーリングできます。Operator イメージをディスク上のファイルベースのカタログに OCI 形式でコピーできます。次に、ローカル OCI イメージをターゲットミラーレジストリーにコピーできます。



重要

oc-mirror プラグインを使用して Operator イメージを OCI 形式でミラーリングすることは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

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

OCI 機能を使用している場合、イメージはターゲットミラーレジストリーから自動的にプルーニングされません。

前提条件

- 必要なコンテナーイメージを取得するためのインターネットへのアクセスがある。
- OpenShift CLI (**oc**) がインストールされている。
- **oc-mirror** CLI プラグインをインストールしている。

手順

1. オプション: 必要なカタログとイメージを取得し、ディスクに保存します。ディスク上に OCI 形式のカタログイメージがすでにある場合は、この手順を省略できます。
   a. イメージセット設定ファイルを作成します。

ディスクにコピーするためのイメージセット設定ファイルの例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
mirror:
 operators:
 - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
   packages:
   - name: aws-load-balancer-operator
```

注記

OCI 機能を使用する場合、**mirror.operators.catalog** 設定のみを使用できます。

storageConfig 設定は無視され、**oc mirror** コマンドに渡された場所が優先されます。

b. **oc mirror** コマンドを実行して、指定されたイメージセット設定からディスクにイメージをミラーリングします。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  --use-oci-feature \ 2
  --oci-feature-action=copy \ 3
  oci://my-oci-catalog 4
```

1. イメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. **--use-oci-feature** フラグを使用して OCI 機能を有効にします。
3. カタログをディスクにコピーするには、**--oci-feature-action** フラグを **copy** に設定します。
4. カタログを出力するディスク上のディレクトリーを指定します。この手順では、名前が **my-oci-catalog** であることを前提としています。パスは **oci:**// で始まる必要があります。指定されたディレクトリーがフルパスでない場合、ディレクトリーは **oc mirror** コマンドが実行される現在の作業ディレクトリーに作成されます。

注記

オプションで **--oci-registries-config** フラグを使用して、TOML 形式の **registries.conf** ファイルへのパスを指定できます。これを使用して、イメージセット設定ファイルを変更することなく、テスト用の運用前の場所など、別のレジストリーからミラーリングできます。

registries.conf ファイルの例

```
[[registry]]
 location = "registry.redhat.io:5000"
 insecure = false
 blocked = false
 mirror-by-digest-only = true
 prefix = ""
 [[registry.mirror]]
   location = "preprod-registry.example.com"
   insecure = false
```

registry.mirror セクションの **location** フィールドを、イメージを取得する別のレジストリーの場所に設定します。**registry** セクションの **location** フィールドは、イメージセット設定ファイルで指定したものと同じレジストリーの場所である必要があります。

c. ディレクトリーの内容を一覧表示し、次のディレクトリーが作成されたことを確認します。

```
$ ls -l
```

出力例

```
my-oci-catalog      1
oc-mirror-workspace 2
olm_artifacts       3
```

1. OCI カタログを含むディレクトリー。この手順では、名前が **my-oci-catalog** であることを前提としています。
2. カタログ内の各イメージを元の形式で含むディレクトリー。
3. このカタログが参照する Operator バンドルを記述するファイルを含むディレクトリー。

2. イメージセット設定ファイルを更新して、ターゲットミラーレジストリーにミラーリングするディスク上のカタログの場所を指定します。

レジストリーをミラーリングするためのイメージセット設定ファイルの例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
mirror:
 operators:
 - catalog: oci:///home/user/oc-mirror/my-oci-catalog/redhat-operator-index 1
   packages:
   - name: aws-load-balancer-operator
```

1. ディスク上の OCI カタログの場所への絶対パスを指定します。この手順は、ディレクトリーとして **my-oci-catalog** を使用し、**redhat-operator-index** カタログをミラーリングしたことを前提としています。パスは **oci:**// で始まる必要があります。

3. oc mirror コマンドを実行して、ディスク上のイメージセットファイルを処理し、その内容をターゲットミラーレジストリーにミラーリングします。

```
$ oc mirror --config=./imageset-config.yaml \ 1
  --use-oci-feature \                          2
  --oci-feature-action=mirror \                3
  docker://registry.example:5000               4
```

1. 更新されたイメージセット設定ファイルを渡します。この手順では、**imageset-config.yaml** という名前であることを前提としています。
2. **--use-oci-feature** フラグを使用して OCI 機能を有効にします。
3. カタログをターゲットミラーレジストリーにミラーリングするには、**--oci-feature-action** フラグを **mirror** に設定します。
4. イメージセットファイルをミラーリングするレジストリーを指定します。レジストリーは **docker:**// で始まる必要があります。ミラーレジストリーに最上位の namespace を指定する場合は、これ以降の実行でもこれと同じ namespace を使用する必要があります。

注記

オプションで **--oci-insecure-signature-policy** フラグを使用して、署名をターゲットミラーレジストリーにプッシュしないようにすることができます。

次のステップ

- oc-mirror が生成したリソースを使用するようにクラスターを設定します。

関連情報

- ファイルベースのカタログ

13.2.3.11. Image set configuration parameters

oc-mirror プラグインには、ミラーリングするイメージを定義するイメージセット設定ファイルが必要です。次の表に、**ImageSetConfiguration** リソースで使用可能なパラメーターを示します。

表13.1 ImageSetConfiguration パラメーター

パラメーター	説明	値
apiVersion	**ImageSetConfiguration** コンテンツの API バージョン。	文字列。例: **mirror.openshift.io/v1alpha2**
archiveSize	イメージセット内の各アーカイブファイルの最大サイズ (GiB 単位)。	integer例: **4**
mirror	イメージセットの設定。	オブジェクト

mirror.additionalImages	イメージセットの追加のイメージ設定。	オブジェクトの配列。以下に例を示します。
		`additionalImages:`
		`  - name: registry.redhat.io/ubi8/ubi:latest`

mirror.additionalImages.name	ミラーリングするイメージのタグまたはダイジェスト。	文字列。例: **registry.redhat.io/ubi8/ubi:latest**
mirror.blockedImages	ミラーリングからブロックするイメージの完全なタグ、ダイジェスト、またはパターン。	文字列の配列例: **docker.io/library/alpine**

パラメーター	説明	値
mirror.helm	イメージセットのヘルム設定。oc-mirror プラグインは、レンダリング時にユーザー入力を必要としないヘルムチャートのみをサポートすることに注意してください。	オブジェクト
mirror.helm.local	ミラーリングするローカルヘルムチャート。	オブジェクトの配列。以下に例を示します。 local: - name: podinfo path: /test/podinfo-5.0.0.tar.gz
mirror.helm.local.name	ミラーリングするローカルヘルムチャートの名前。	文字列。例: **podinfo**。
mirror.helm.local.path	ミラーリングするローカルヘルムチャートのパス。	文字列。例: **/test/podinfo-5.0.0.tar.gz**
mirror.helm.repositories	ミラーリング元のリモートヘルムリポジトリー。	オブジェクトの配列。以下に例を示します。 repositories: - name: podinfo url: https://example.github.io/podinfo charts: - name: podinfo version: 5.0.0
mirror.helm.repositories.name	ミラーリング元のヘルムリポジトリーの名前。	文字列。例: **podinfo**。
mirror.helm.repositories.url	ミラーリング元の helm リポジトリーの URL。	文字列。例: **https://example.github.io/podinfo**

パラメーター	説明	値
mirror.helm.repositories.charts	ミラーリングするリモートヘルムチャート。	オブジェクトの配列。
mirror.helm.repositories.charts.name	ミラーリングするヘルムチャートの名前。	文字列。例: **podinfo**。
mirror.helm.repositories.charts.version	ミラーリングする名前付きヘルムチャートのバージョン。	文字列。例: **5.0.0**。
mirror.operators	イメージセットの Operators 設定。	オブジェクトの配列。以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12 packages: - name: elasticsearch-operator minVersion: '2.4.0'
mirror.operators.catalog	イメージセットに含める Operator カタログ。	文字列。例: **registry.redhat.io/redhat/redhat-operator-index:v4.12**
mirror.operators.full	**true** の場合、完全なカタログ、Operator パッケージ、または Operator チャネルをダウンロードします。	ブール値。デフォルト値は **false** です。

パラメーター	説明	値
mirror.operators.packages	Operator パッケージ設定	オブジェクトの配列。以下に例を示します。 operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
mirror.operators.packages.name	イメージセットに含める Operator パッケージ名	文字列。例: **elasticsearch-operator**
mirror.operators.packages.channels	Operator パッケージのチャネル設定。	オブジェクト
mirror.operators.packages.channels.name	イメージセットに含める、パッケージ内で一意の Operator チャネル名。	文字列。例:**fast** または **stable-v4.12**
mirror.operators.packages.channels.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**
mirror.operators.packages.channels.minBundle	含める最小バンドルの名前と、チャネルヘッドへのアップグレードグラフ内のすべてのバンドル。名前付きバンドルにセマンティックバージョンメタデータがない場合にのみ、このフィールドを設定します。	文字列。例 : **bundleName**
mirror.operators.packages.channels.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**
mirror.operators.packages.maxVersion	Operator が存在するすべてのチャネルでミラーリングする最上位バージョンの Operator。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**。

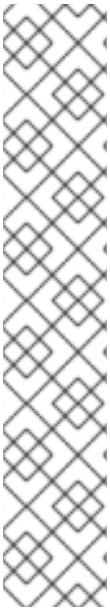
パラメーター	説明	値
mirror.operators.packages.minVersion	存在するすべてのチャネル間でミラーリングする Operator の最低バージョン。詳細は、以下の注記を参照してください。	文字列。例: **5.2.3-31**。
mirror.operators.skipDependencies	**true** の場合、バンドルの依存関係は含まれません。	ブール値。デフォルト値は **false** です。
mirror.operators.targetName	参照カタログをミラーリングするためのオプションの代替名。	文字列。例： **my-operator-catalog**
mirror.operators.targetTag	**targetName** に追加するオプションの代替タグ。	文字列。例: **v1**
mirror.platform	イメージセットのプラットフォーム設定。	オブジェクト

mirror.platform.architectures	ミラーリングするプラットフォームリリースペイロードのアーキテクチャー。	文字列の配列以下に例を示します。
		`architectures:`
		`- amd64`
		`- arm64`

mirror.platform.channels	イメージセットのプラットフォームチャネル設定。	オブジェクトの配列。以下に例を示します。
		`channels:`
		`- name: stable-4.10`
		`- name: stable-4.12`

mirror.platform.channels.full	**true** の場合、**minVersion** をチャネルの最初のリリースに設定し、**maxVersion** をチャネルの最後のリリースに設定します。	ブール値。デフォルト値は **false** です。
mirror.platform.channels.name	リリースチャネルの名前。	文字列。例: **stable-4.12**
mirror.platform.channels.minVersion	ミラーリングされる参照プラットフォームの最小バージョン。	文字列。例: **4.9.6**
mirror.platform.channels.maxVersion	ミラーリングされる参照プラットフォームの最上位バージョン。	文字列。例: **4.12.1**

パラメーター	説明	値
mirror.platform.channels.shortestPath	最短パスミラーリングまたはフルレンジミラーリングを切り替えます。	ブール値。デフォルト値は **false** です。
mirror.platform.channels.type	ミラーリングするプラットフォームのタイプ。	文字列。例： **ocp** または **okd**。デフォルトは **ocp** です。
mirror.platform.graph	OSUS グラフがイメージセットに追加され、その後ミラーに公開されるかどうかを示します。	ブール値。デフォルト値は **false** です。
storageConfig	イメージセットのバックエンド設定。	オブジェクト
storageConfig.local	イメージセットのローカルバックエンド設定。	オブジェクト
storageConfig.local.path	イメージセットのメタデータを含むディレクトリーのパス。	文字列。例: **./path/to/dir/**
storageConfig.registry	イメージセットのレジストリーバックエンド設定。	オブジェクト
storageConfig.registry.imageURL	バックエンドレジストリー URI。オプションで、URI に namespace 参照を含めることができます。	文字列。例: **quay.io/myuser/imageset:metadata**
storageConfig.registry.skipTLS	オプションで、参照されるバックエンドレジストリーの TLS 検証をスキップします。	ブール値。デフォルト値は **false** です。



注記

minVersion および **maxVersion** プロパティーを使用して特定の Operator バージョン範囲をフィルターすると、複数のチャネルヘッドエラーが発生する可能性があります。エラーメッセージには、**multiple channel heads** があることが表示されます。これは、フィルターが適用されると、Operator の更新グラフが切り捨てられるためです。

Operator Lifecycle Manager では、すべての Operator チャネルに、1 つのエンドポイント (最新バージョンの Operator) を持つ更新グラフを形成するバージョンが含まれている必要があります。グラフが 2 つ以上の別個のグラフ、または複数のエンドポイントを持つグラフに移動できるフィルター範囲を適用する場合。

このエラーを回避するには、最新バージョンの Operator を除外しないでください。それでもエラーが発生する場合は、Operator に応じて **maxVersion** プロパティーを増やすか、**minVersion** プロパティーを減らす必要があります。すべての Operator グラフは異なる可能性があるため、エラーがなくなるまで、手順に従ってこれらの値を調整する必要がある場合があります。

13.2.3.12. Image set configuration examples

次の **ImageSetConfiguration** ファイルの例は、さまざまなミラーリングのユースケースの設定を示しています。

ユースケース: 最短の OpenShift Container Platform アップグレードパスを含める
以下の **ImageSetConfiguration** ファイルは、ローカルストレージバックエンドを使用し、最小バージョン **4.11.37** から最大バージョン **4.12.15** への最短アップグレードパスに沿ってすべての OpenShift Container Platform バージョンを含めます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.11.37
        maxVersion: 4.12.15
        shortestPath: true
```

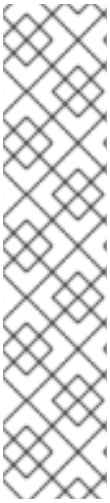
ユースケース: OpenShift Container Platform の最小バージョンから最新バージョンまでのすべてのバージョンを含める
以下の **ImageSetConfiguration** ファイルは、レジストリーストレージバックエンドを使用し、最小バージョン **4.10.10** からチャネルの最新バージョンまでのすべての OpenShift Container Platform バージョンを含みます。

このイメージセット設定で oc-mirror を呼び出すたびに、**stable-4.10** チャネルの最新リリースが評価されるため、定期的に oc-mirror を実行すると、OpenShift Container Platform イメージの最新リリースを自動的に受け取ることができます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  platform:
    channels:
      - name: stable-4.10
        minVersion: 4.10.10
```

ユースケース: 最小から最新までの Operator バージョンを含める
次の**ImageSetConfiguration**ファイルは、ローカルストレージバックエンドを使用し、これには、**stable** チャネルの Kubernetes Operator 用の Red Hat Advanced Cluster Security (4.0.1 以降のバージョン) のみが含まれています。



注記

最小または最大のバージョン範囲を指定した場合、その範囲内のすべての Operator バージョンを受信できない可能性があります。

デフォルトで、oc-mirror は、Operator Lifecycle Manager (OLM) 仕様でスキップされたバージョン、または新しいバージョンに置き換えられたバージョンを除外します。スキップされた Operator のバージョンは、CVE の影響を受けるか、バグが含まれている可能性があります。代わりに新しいバージョンを使用してください。スキップおよび置き換えられたバージョンの詳細は、OLM を使用した更新グラフの作成 を参照してください。

指定した範囲内のすべての Operator バージョンを受信するには、**mirror.operators.full** フィールドを **true** に設定します。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
            minVersion: 4.0.1
```

注記

最新バージョンではなく最大バージョンを指定するには、**mirror.operators.packages.channels.maxVersion** フィールドを設定します。

ユースケース: Nutanix CSI Operator を含める
次の **ImageSetConfiguration** ファイルは、ローカルストレージバックエンドを使用します。このファイルには、Nutanix CSI Operator、OpenShift Update Service (OSUS) グラフイメージ、および追加の Red Hat Universal Base Image (UBI) が含まれます。

ImageSetConfigurationファイルの例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  registry:
    imageURL: mylocalregistry/ocp-mirror/openshift4
    skipTLS: false
mirror:
  platform:
    channels:
    - name: stable-4.12
      type: ocp
    graph: true
  operators:
  - catalog: registry.redhat.io/redhat/certified-operator-index:v4.12
    packages:
    - name: nutanixcsioperator
      channels:
      - name: stable
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest
```

ユースケース: デフォルトの Operator チャネルを含める
次の **ImageSetConfiguration** ファイルには、OpenShift Elasticsearch Operator の **stable-5.7** および **stable** チャネルが含まれています。**安定版 5.7** チャネルのパッケージのみが必要な場合でも、**stable** チャネルは Operator のデフォルトチャネルであるため、**ImageSetConfiguration** ファイルにも含める必要があります。そのチャネルでバンドルを使用しない場合も、常に Operator パッケージのデフォルトチャネルを含める必要があります。

ヒント

コマンド **oc mirror list operators --catalog=<catalog_name> --package=<package_name>** を実行すると、デフォルトチャネルを見つけることができます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
    packages:
    - name: elasticsearch-operator
```

```
    channels:
    - name: stable-5.7
    - name: stable
```

ユースケース: カタログ全体を含める (すべてのバージョン)
次の **ImageSetConfiguration** ファイルは、**mirror.operators.full** フィールドを **true** に設定して、Operator カタログ全体のすべてのバージョンを含めます。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
      full: true
```

ユースケース: カタログ全体を含める (チャネルヘッドのみ)
次の **ImageSetConfiguration** ファイルには、Operator カタログ全体のチャネルヘッドが含まれています。

デフォルトでは、カタログ内の各 Operator において、oc-mirror にはデフォルトチャネルから Operator の最新バージョン (チャネルヘッド) が含まれています。チャネルヘッドだけでなく、すべての Operator バージョンをミラーリングする場合は、**mirror.operators.full** フィールドを **true** に設定する必要があります。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
```

ユースケース: 任意のイメージとヘルムチャートを含む
次の **ImageSetConfiguration** ファイルは、レジストリーストレージバックエンドを使用し、これにはヘルムチャートと追加の Red Hat Universal Base Image (UBI) が含まれています。

ImageSetConfigurationファイルの例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
archiveSize: 4
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
```

```
    skipTLS: false
mirror:
 platform:
   architectures:
     - "s390x"
   channels:
     - name: stable-4.12
 operators:
   - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
 helm:
   repositories:
     - name: redhat-helm-charts
       url: https://raw.githubusercontent.com/redhat-developer/redhat-helm-charts/master
       charts:
         - name: ibm-mongodb-enterprise-helm
           version: 0.2.0
 additionalImages:
   - name: registry.redhat.io/ubi9/ubi:latest
```

13.2.3.13. oc-mirror のコマンドリファレンス

以下の表は、**oc mirror** サブコマンドとフラグについて説明しています。

表13.2 oc mirror サブコマンド

サブコマンド	説明
completion	指定されたシェルのオートコンプリートスクリプトを生成します。
describe	イメージセットの内容を出力します。
help	サブコマンドに関するヘルプを表示します。
init	初期イメージセット設定テンプレートを出力します。
list	利用可能なプラットフォームと Operator のコンテンツとそのバージョンを一覧表示します。
version	oc-mirror バージョンを出力します。

表13.3 oc mirror フラグ

フラグ	説明
-c, **--config <string>**	イメージセット設定ファイルへのパスを指定します。
--continue-on-error	イメージのプルに関連しないエラーが発生した場合は、続行して、可能な限りミラーリングを試みます。
--dest-skip-tls	ターゲットレジストリーの TLS 検証を無効にします。

フラグ	説明
--dest-use-http	ターゲットレジストリーにはプレーン HTTP を使用します。
--dry-run	イメージをミラーリングせずにアクションを出力します。**mapping.txt** ファイルおよび **pruning-plan.json** ファイルを生成します。
--from <string>	oc-mirror の実行によって生成されたイメージセットアーカイブへのパスを指定して、ターゲットレジストリーにロードします。
-h、--help	ヘルプを表示します。
--ignore-history	イメージをダウンロードしてレイヤーをパックするときに、過去のミラーリングを無視します。増分ミラーリングを無効にし、より多くのデータをダウンロードする可能性があります。
--manifests-only	**ImageContentSourcePolicy** オブジェクトのマニフェストを生成して、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ますが、実際にはイメージをミラーリングしません。このフラグを使用するには、**--from** フラグでイメージセットアーカイブを渡す必要があります。
--max-nested-paths <int>	ネストされたパスを制限する宛先レジストリーのネストされたパスの最大数を指定します。デフォルトは **2** です。
--max-per-registry <int>	レジストリーごとに許可される同時要求の数を指定します。デフォルト値は **6** です。
--oci-feature-action	テクノロジープレビュー OCI 機能の使用時に実行するアクション。オプションは **copy** または **mirror** です。
--oci-insecure-signature-policy	テクノロジープレビュー OCI 機能を使用する場合は、署名をプッシュしないでください。
--oci-registries-config	テクノロジープレビュー OCI 機能を使用する場合にコピー元となる別のレジストリーの場所を指定するレジストリー設定ファイルを提供します。
--skip-cleanup	アーティファクトディレクトリーの削除を省略します。
--skip-image-pin	Operator カタログのイメージタグをダイジェストピンに置き換えないでください。
--skip-metadata-check	イメージセットの公開時にメタデータをスキップします。これは、イメージセットが **--ignore-history** で作成された場合にのみ推奨されます。

フラグ	説明
--skip-missing	イメージが見つからない場合は、エラーを報告して実行を中止する代わりにスキップします。イメージセット設定で明示的に指定されたカスタムイメージには適用されません。
--skip-pruning	ターゲットミラーレジストリーからのイメージの自動プルーニングを無効にします。
--skip-verification	ダイジェストの検証を省略します。
--source-skip-tls	ソースレジストリーの TLS 検証を無効にします。
--source-use-http	ソースレジストリーにはプレーン HTTP を使用します。
--use-oci-feature	OCI 形式のイメージをコピーするには、テクノロジープレビュー OCI 機能を使用します。
-v, --verbose <int>	ログレベルの詳細度の数値を指定します。有効な値は **0** - **9** です。デフォルトは **0** です。

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

重要

OpenShift Update Service アプリケーションによる過度のメモリー使用を回避するには、以下の手順で説明するように、リリースイメージを別のリポジトリーにミラーリングする必要があります。

前提条件

- 非接続環境で使用するミラーレジストリーを設定し、設定した証明書と認証情報にアクセスできるようになりました。
- Red Hat OpenShift Cluster Manager からプルシークレット をダウンロードし、ミラーリポジトリーへの認証を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 はチャネルのグラフと、現行バージョンと意図されるクラスターのバージョン間に更新パスがあることを確認する方法を提供します。
2. 必要な環境変数を設定します。
   a. リリースバージョンをエクスポートします。

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

<release_version> について、更新する OpenShift Container Platform のバージョンに対応するタグを指定します (例: **4.5.4**)。

b. ローカルレジストリー名とホストポートをエクスポートします。

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

<local_registry_host_name> については、ミラーレジストリーのレジストリードメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。

c. ローカルリポジトリー名をエクスポートします。

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

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

d. OpenShift Update Service を使用している場合は、追加のローカルリポジトリー名をエクスポートして、リリースイメージを含めます。

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

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

e. ミラーリングするリポジトリーの名前をエクスポートします。

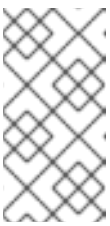
```
$ PRODUCT_REPO='openshift-release-dev'
```

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

f. パスをレジストリープルシークレットにエクスポートします。

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

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



注記

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

g. リリースミラーをエクスポートします。

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

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

h. 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。

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

ii. イメージおよび設定マニフェストをリムーバブルメディア上のディレクトリーにミラーリングします。

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

注記

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

iii. メディアを非接続環境に移動し、イメージをローカルコンテナーレジストリーにアップロードします。

```
$ oc image mirror  -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} 1
```

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

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

v. ミラーリングされたリリースイメージ署名設定マップを接続されたクラスターに適用します。

```
$ oc apply -f ${REMOVABLE_MEDIA_PATH}/mirror/config/<image_signature_file>
```
(1)

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

vi. OpenShift Update Service を使用している場合は、リリースイメージを別のリポジトリーにミラーリングします。

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

- ローカルコンテナーレジストリー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。

i. 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ーにミラーリングします。

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

13.3. OPENSHIFT UPDATE SERVICE を使用した非接続環境でのクラスターの更新

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

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

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

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

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

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

ただし、非接続環境のクラスターは、これらのパブリック API にアクセスして更新情報を取得することはできません。非接続環境で同じように更新を行うには、OpenShift Update Service を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。

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

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

次のセクションでは、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ーに提供するように設定する方法について説明します。

関連情報

- OpenShift Update Service について
- 更新チャネルとリリースについて

13.3.2. 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ーのミラーリング で説明されているように、更新用のコンテナーイメージを使用してローカルのコンテナーイメージレジストリーをプロビジョニングしている。

13.3.3. 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。

13.3.4. 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="<registry>" \ 1
--auth-basic="<username>:<password>" \ 2
--to=<pull_secret_location> 3
```

1. 新しいレジストリーを指定します。同じレジストリー内に複数のリポジトリーを含めることができます (例: **--registry="<registry/my-namespace/my-repository>"**)。
2. 新しいレジストリーの認証情報を指定します。
3. プルシークレットファイルへのパスを指定します。

または、プルシークレットファイルを手動で更新することもできます。

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

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

1. 新規プルシークレットファイルへのパスを指定します。

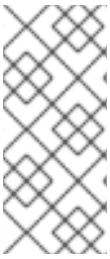
この更新はすべてのノードにロールアウトされます。これには、クラスターのサイズに応じて多少時間がかかる場合があります。

注記

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

13.3.5. OpenShift Update Service のインストール

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



注記

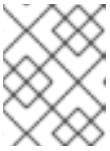
非接続環境 (非接続クラスターとして知られる) にインストールされているクラスターの場合には、デフォルトで Operator Lifecycle Manager はリモートレジストリーでホストされる Red Hat が提供する OperatorHub ソースにアクセスできません。それらのリモートソースには完全なインターネット接続が必要であるためです。詳細は、ネットワークが制限された環境での Operator Lifecycle Manager の使用 を参照してください。

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

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

手順

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



注記

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

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

13.3.5.2. 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ーの場合には、**redhat-operators** を指定します。OpenShift Container Platform クラスターが非接続環境にインストールされている場合、Operator Lifecycle Manager (OLM) を設定したときに作成された **CatalogSource** オブジェクトの名前を指定します。

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。バージョン番号は表示されているものと異なる場合があります。

関連情報

- namespace への Operator のインストール

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

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

注記

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

手順

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ーイメージ (例: **registry.example.com/openshift/graph-data:latest**) を構築します。

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

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

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

注記

非接続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ーイメージを、OpenShift Update Service からアクセス可能なリポジトリーにコピーします。利用可能なオプションについては、**oc image mirror --help** を実行します。

13.3.7. 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。

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

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

前提条件

- OpenShift Update Service Operator がインストールされている。
- OpenShift Update Service のグラフデータコンテナーイメージを作成して、OpenShift Update Service がアクセスできるリポジトリーにプッシュしている。
- 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 のリストから **OpenShift Update Service** を選択します。
3. **Update Service** タブをクリックします。
4. **Create UpdateService** をクリックします。
5. **service** など、**Name** フィールドに名前を入力します。

6. **Graph Data Image** フィールドに OpenShift Update Service グラフデータコンテナーイメージの作成で作成した graph-data コンテナーイメージにローカルの pullspec を入力します (例: **registry.example.com/openshift/graph-data:latest**)。

7. **Releases** フィールドに、OpenShift Container Platform イメージリポジトリーのミラーリングでリリースイメージを含むように作成したローカルのレジストリーとリポジトリー (例: **registry.example.com/ocp4/openshift4-release-images**) を入力します。

8. **Replicas** フィールドに **2** と入力します。

9. **Create** をクリックして OpenShift Update Service アプリケーションを作成します。

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

    - **Update Service** タブの **UpdateServices** リストから、作成した Update Service アプリケーションをクリックします。
    - **Resources** タブをクリックします。
    - 各アプリケーションリソースのステータスが **Created** であることを確認します。

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

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

前提条件

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

手順

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

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

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

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

   ```
   $ NAME=service
   ```

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

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

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

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

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

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

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

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

   a. 以下のコマンドを使用してポリシーエンジンルートを取得します。

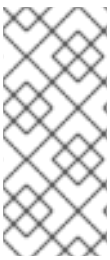
```
$ while sleep 1; do POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
"${NAME}")"; SCHEME="${POLICY_ENGINE_GRAPH_URI%%:*}"; if test "${SCHEME}"
= http -o "${SCHEME}" = https; then break; fi; done
```

   コマンドが成功するまでポーリングが必要になる場合があります。

   b. ポリシーエンジンからグラフを取得します。**チャネル** に有効なバージョンを指定してください。たとえば、OpenShift Container Platform 4.12 で実行している場合は、**stable-4.12** を使用します。

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

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



注記

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

13.3.7.2.1. 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 のグラフデータコンテナーイメージを作成して、OpenShift Update Service がアクセスできるリポジトリーにプッシュしている。
- 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。

13.3.8. 次のステップ

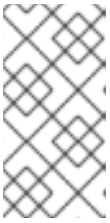
クラスターを更新する前に、次の条件が満たされていることを確認してください。

- 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。

注記

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

- 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ー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。
- 最新バージョンの OpenShift Update Service Operator がインストールされている。



注記

OpenShift Update Service Operator を最近インストールまたは更新していない場合は、さらに新しいバージョンが利用できる可能性があります。非接続環境で OLM カタログを更新する方法の詳細は、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。

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ーの更新
- 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 コンピュートマシンを含むクラスターの更新

13.4. OPENSHIFT UPDATE SERVICE を使用しない非接続環境でのクラスターの更新

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

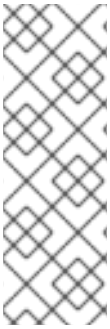
13.4.1. 前提条件

- **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ーのミラーリング で説明されているように、更新用のコンテナーイメージを使用してローカルのコンテナーイメージレジストリーをプロビジョニングしている。
- **admin** 権限を持つユーザーとしてクラスターにアクセスできる。RBAC の使用によるパーミッションの定義および適用 を参照してください。
- 更新が失敗し、クラスターを以前の状態に復元する 必要がある場合に備えて、最新の etcd バックアップ 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 を一時停止できる。

- クラスターが手動で維持された認証情報を使用している場合は、新しいリリース用にクラウドプロバイダーリソースを更新します。これがクラスターの要件かどうかを判断する方法などについて、詳しくは 手動で維持された認証情報でクラスターを更新する準備 を参照してください。

- Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合、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。



注記

Operator を実行している場合、または Pod 中断バジェットを使用してアプリケーションを設定している場合、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。

13.4.2. MachineHealthCheck リソースの一時停止

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

前提条件

- OpenShift CLI (**oc**) がインストールされている。

手順

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

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

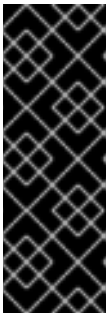
2. マシンヘルスチェックを一時停止するには、**cluster.x-k8s.io/paused=""** アノテーションを **MachineHealthCheck** リソースに追加します。以下のコマンドを実行します。

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

アノテーション付きの **MachineHealthCheck** リソースは以下の YAML ファイルのようになります。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
```

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



重要

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

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

13.4.3. リリースイメージダイジェストの取得

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

手順

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

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

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

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

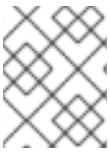
出力例

```
sha256:a8bfba3b6dddd1a2fbbead7dac65fe4fb8335089e4e7cae327f3bad334add31d
```

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

13.4.4. 切断されたクラスターの更新

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



注記

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

前提条件

- 新規リリースのイメージをレジストリーに対してミラーリングしている。
- 新規リリースのリリースイメージ署名 ConfigMap をクラスターに適用している。

注記

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

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

手順

- クラスターを更新します。

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

ここでは、以下のようになります。

<defined_registry>
イメージのミラーリング先であるミラーレジストリーの名前を指定します。

<defined_repository>
ミラーレジストリーで使用するイメージリポジトリーの名前を指定します。

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

注記

- ミラーレジストリーとリポジトリー名の定義を確認するには、「OpenShift Container Platform イメージのミラーリング」を参照してください。
- **ImageContentSourcePolicy** または **ImageDigestMirrorSet** を使用した場合は、定義した名前の代わりに標準的なレジストリー名とリポジトリー名を使用できます。標準的なレジストリー名は **quay.io**、標準的なリポジトリー名は **openshift-release-dev/ocp-release** です。
- **ImageContentSourcePolicy** オブジェクトを持つクラスターのグローバルプルシークレットのみを設定できます。プロジェクトにプルシークレットを追加することはできません。

関連情報

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

13.4.5. イメージレジストリーのリポジトリーミラーリングの設定

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

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

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

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

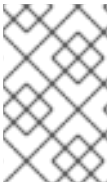
リポジトリーミラーリングのセットアップは次の方法で実行できます。

- OpenShift Container Platform のインストール時:
  OpenShift Container Platform に必要なコンテナーイメージをプルし、それらのイメージを会社のファイアウォールの背後に配置することで、非接続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。

次の手順では、インストール後のミラー設定を行います。ここでは、以下を特定する **ImageContentSourcePolicy** オブジェクトを作成します。

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



注記

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

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

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

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

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

   ```
   apiVersion: operator.openshift.io/v1alpha1
   kind: ImageContentSourcePolicy
   metadata:
   ```

```
  name: ubi8repo
spec:
  repositoryDigestMirrors:
  - mirrors:
    - example.io/example/ubi-minimal (1)
    - 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、**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. ノードのリストを表示します。

   ```
   $ oc get node
   ```

   出力例

   ```
   NAME                         STATUS                     ROLES    AGE  VERSION
   ip-10-0-137-44.ec2.internal    Ready                        worker   7m   v1.25.0
   ip-10-0-138-148.ec2.internal   Ready                         master   11m  v1.25.0
   ip-10-0-139-122.ec2.internal   Ready                         master   11m  v1.25.0
   ip-10-0-147-35.ec2.internal    Ready                        worker   7m   v1.25.0
   ip-10-0-153-12.ec2.internal    Ready                        worker   7m   v1.25.0
   ip-10-0-154-10.ec2.internal    Ready                        master   11m  v1.25.0
   ```

   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@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6
```

リポジトリーのミラーリングのトラブルシューティング

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

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

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

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

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

前提条件

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

手順

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

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

ここでは、以下のようになります。

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

<pull_spec>
: 非接続レジストリーで設定されるプル仕様です (例: **redhat/redhat-operator-index:v4.12**)。

<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ーに適用します。

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

検証

検証

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

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

出力例

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

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

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

13.4.7. 関連情報

- ネットワークが制限された環境での Operator Lifecycle Manager の使用
- マシン設定の概要

13.5. クラスターからの OPENSHIFT UPDATE SERVICE のアンインストール

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

13.5.1. 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。

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

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

前提条件

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

手順

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

13.5.1.2. 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
```

13.5.2. 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。

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

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

前提条件

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

手順

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

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

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

前提条件

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

手順

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

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

出力例

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

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

```
$ oc get operatorgroup
```

出力例

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

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

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

出力例

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

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

```
$ oc get subscription
```

出力例

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

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

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

出力例

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

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

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

出力例

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

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

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

出力例

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

第14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 15 以降は、クラスター内の vSphere 仮想マシンでサポートされます。

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

重要

OpenShift Container Platform のバージョン 4.12 には、VMware 仮想ハードウェアバージョン 15 以降が必要です。

14.1. 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。

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

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

前提条件

- OpenShift Container Platform クラスターをホストする vCenter インスタンスで必要なパーミッションを実行するためのクラスター管理者パーミッションがある。
- vSphere ESXi ホストがバージョン 7.0U2 以降を使用している。

手順

1. クラスターのコントロールプレーンノードをリスト表示します。

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

出力例

```
NAME                 STATUS   ROLES    AGE   VERSION
control-plane-node-0    Ready    master   75m   v1.25.0
control-plane-node-1    Ready    master   75m   v1.25.0
control-plane-node-2    Ready    master   75m   v1.25.0
```

コントロールプレーンノードの名前を書き留めておきます。

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

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

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

シンをシャットダウンしないでください。

4. vSphere クライアントで VM を更新します。詳細については、VMware ドキュメントの 仮想マシンの互換性を手動でアップグレードするに 従ってください。

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

6. ノードが **Ready** として報告されるまで待機します。

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

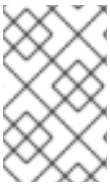
7. コントロールプレーンノードを再度スケジュール対象としてマークします。

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

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

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

ダウンタイムのリスクを軽減するには、コンピュートノードを順次更新することが推奨されます。



注記

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

前提条件

- OpenShift Container Platform クラスターをホストする vCenter インスタンスで必要なパーミッションを実行するためのクラスター管理者パーミッションがある。
- vSphere ESXi ホストがバージョン 7.0U2 以降を使用している。

手順

1. クラスターのコンピュートノードをリスト表示します。

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

出力例

```
NAME           STATUS   ROLES    AGE   VERSION
compute-node-0   Ready    worker   30m   v1.25.0
compute-node-1   Ready    worker   30m   v1.25.0
compute-node-2   Ready    worker   30m   v1.25.0
```

コンピュートノードの名前を書き留めておきます。

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

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

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

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

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

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

5. vSphere クライアントで VM を更新します。詳細については、VMware ドキュメントの 仮想マシンの互換性を手動でアップグレードするに 従ってください。

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

7. ノードが **Ready** として報告されるまで待機します。

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

8. コンピュートノードを再度スケジュール対象としてマークします。

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

9. クラスター内のコンピュートノードごとに、この手順を繰り返します。

14.1.3. vSphere 上のテンプレートの仮想ハードウェアの更新

前提条件

- OpenShift Container Platform クラスターをホストする vCenter インスタンスで必要なパーミッションを実行するためのクラスター管理者パーミッションがある。
- vSphere ESXi ホストがバージョン 7.0U2 以降を使用している。

手順

1. RHCOS テンプレートが vSphere テンプレートとして設定されている場合は、次のステップの前に、VMware ドキュメント のテンプレートを仮想マシンに変換するに 従ってください。

注記

テンプレートから変換したら、仮想マシンをパワーオンしないでください。

2. vSphere クライアントで VM を更新します。詳細については、VMware ドキュメントの 仮想マシンの互換性を手動でアップグレードするに 従ってください。

3. vSphere クライアントの VM を VM からテンプレートに変換します。詳細については、VMware ドキュメント の vSphere Client で仮想マシンをテンプレートに変換するに 従ってください。

関連情報

- ノード上の Pod を退避させる方法

14.2. VSPHERE での仮想ハードウェアの更新のスケジューリング

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

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

第15章 カーネルモジュール管理 (KMM) モジュールのプリフライト検証

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

15.1. 検証のキックオフ

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

```
type PreflightValidationOCPSpec struct {
 // releaseImage describes the OCP release image that all Modules need to be checked against.
 // +kubebuilder:validation:Required
 ReleaseImage string `json:"releaseImage"` (1)
 // Boolean flag that determines whether images build during preflight must also
 // be pushed to a defined repository
 // +optional
 PushBuiltImage bool `json:"pushBuiltImage"` (2)
}
```

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

(2) **PushBuiltImage** - **true** の場合は、ビルドおよび署名の検証中に作成されたイメージがリポジトリーにプッシュされます (デフォルトでは、**false**)。

15.2. 検証のライフサイクル

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

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

15.3. 検証のステータス

プリフライトは、検証を試みるクラスター内の各モジュールのステータスと進行状況を報告します。

```
type CRStatus struct {
 // Status of Module CR verification: true (verified), false (verification failed),
 // error (error during verification process), unknown (verification has not started yet)
 // +required
 // +kubebuilder:validation:Required
 // +kubebuilder:validation:Enum=True;False
 VerificationStatus string `json:"verificationStatus"` (1)
```

```
// StatusReason contains a string describing the status source.
// +optional
StatusReason string `json:"statusReason,omitempty"` (2)
// Current stage of the verification process:
// image (image existence verification), build(build process verification)
// +required
// +kubebuilder:validation:Required
// +kubebuilder:validation:Enum=Image;Build;Sign;Requeued;Done
VerificationStage string `json:"verificationStage"` (3)
// LastTransitionTime is the last time the CR status transitioned from one status to another.
// This should be when the underlying status changed.  If that is not known, then using the time when the API field changed is acceptable.
// +required
// +kubebuilder:validation:Required
// +kubebuilder:validation:Type=string
// +kubebuilder:validation:Format=date-time
LastTransitionTime metav1.Time `json:"lastTransitionTime" protobuf:"bytes,4,opt,name=lastTransitionTime"` (4)
}
```

次のフィールドが各モジュールに適用されます。

(1) **VerificationStatus** - **true** または **false**、検証済みかどうか。

(2) **StatusReason** - ステータスに関する口頭での説明。

(3) **VerificationStage** - 実行中の検証ステージ (イメージ、ビルド、署名) を説明します。

(4) **LastTransitionTime** - ステータスが最後に更新された時刻。

15.4. モジュールごとのプリフライト検証ステージ

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

1. イメージの検証ステージ
2. ビルドの検証ステージ
3. 署名の検証ステージ

15.4.1. イメージの検証ステージ

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

イメージの検証は、次の 2 つのステージで設定されます。

1. イメージの存在とアクセシビリティー。コードは、モジュール内のアップグレードされたカーネル用に定義されたイメージにアクセスし、そのマニフェストを取得しようとします。
2. 今後の **modprobe** の実行のために、**Module** で定義されたカーネルモジュールが正しいパスに存在することを確認します。正しいパスは **<dirname>/lib/modules/<upgraded_kernel>/** です。

この検証が成功した場合は、カーネルモジュールが正しい Linux ヘッダーでコンパイルされた可能性が高いです。

15.4.2. ビルドの検証ステージ

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

注記

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

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

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

注記

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

15.4.3. 署名の検証ステージ

署名の検証は、イメージの検証が失敗し、**Module** にアップグレードカーネルに関連する **sign** セクションがあり、アップグレードされたカーネルに関連する **Module** に **build** セクションがあった際にビルドの検証が正常に終了した場合のみ、実行されます。署名の検証では、署名ジョブの実行が試行され、正常に終了したことが検証されます。

PushBuiltImage フラグが **PreflightValidationOCP** CR で定義されている場合、署名の検証は結果のイメージをレジストリーにプッシュしようとします。

結果のイメージは、常に **Module** の **containerImage** フィールドで定義されたイメージです。入力イメージは、Build ステージの出力、または **UnsignedImage** フィールドで定義されたイメージのいずれかです。

注記

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

15.5. PREFLIGHTVALIDATIONOCP リソースの例

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

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

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

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

```
apiVersion: kmm.sigs.x-k8s.io/v1beta1
kind: PreflightValidationOCP
metadata:
 name: preflight
spec:
 releaseImage: quay.io/openshift-release-dev/ocp-
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
 pushBuiltImage: true
```