# OpenShift Container Platform 4.16

## マシン設定

OpenShift Container Platform のベースオペレーティングシステムとコンテナーランタイムの設定と更新の管理と適用

Last Updated: 2024-07-02

# OpenShift Container Platform 4.16 マシン設定

OpenShift Container Platform のベースオペレーティングシステムとコンテナーランタイムの設定と更新の管理と適用

## 法律上の通知



## 概要

このドキュメントでは、MachineConfig、KubeletConfig、ContainerRuntimeConfig オブジェクトを使用して、systemd、CRI-O、Kubelet、カーネル、およびその他のシステム機能への変更を管理する手順について説明します。さらに、イメージのレイヤー化により、クラスターワーカーノードのベースイメージの上に追加のイメージを重ねることで、基盤となるノードのオペレーティングシステムを簡単にカスタマイズできます。

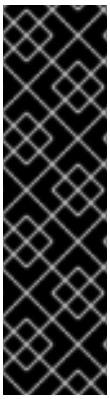
# 目次

# 第1章 マシン設定の概要

OpenShift Container Platform ノードで実行しているオペレーティングシステムに変更を加える必要がある場合があります。これには、ネットワークタイムサービスの設定変更、カーネル引数の追加、特定の方法でのジャーナルの設定などが含まれます。

いくつかの特殊な機能のほかに、OpenShift Container Platform ノードのオペレーティングシステムへの変更のほとんどは、Machine Config Operator によって管理される **MachineConfig** オブジェクトというオブジェクトを作成することで実行できます。たとえば、Machine Config Operator (MCO) とマシン設定を使用して、systemd、CRI-O、kubelet、カーネル、Network Manager、その他のシステム機能の更新を管理できます。

このセクションのタスクでは、Machine Config Operator の機能を使用して OpenShift Container Platform ノードでオペレーティングシステム機能を設定する方法を説明します。



**重要**

NetworkManager は、新しいネットワーク設定を鍵ファイル形式で **/etc/NetworkManager/system-connections/** に保存します。

以前は、NetworkManager が、新しいネットワーク設定を **ifcfg** 形式で **/etc/sysconfig/network-scripts/** に保存していました。RHEL 9.0 以降では、RHEL は新しいネットワーク設定を鍵ファイル形式で **/etc/NetworkManager/system-connections/** に保存します。以前の形式で **/etc/sysconfig/network-scripts/** に保存された接続設定は、引き続き中断されることなく動作します。既存のプロファイルに変更を加えると、そのまま以前のファイルが更新されます。

## 1.1. MACHINE CONFIG OPERATOR について

OpenShift Container Platform 4.16 は、オペレーティングシステムとクラスター管理を統合します。クラスターは、クラスターノードでの Red Hat Enterprise Linux CoreOS (RHCOS) への更新を含め、独自の更新を管理するので、OpenShift Container Platform では事前に設定されたライフサイクル管理が実行され、ノードのアップグレードのオーケストレーションが単純化されます。

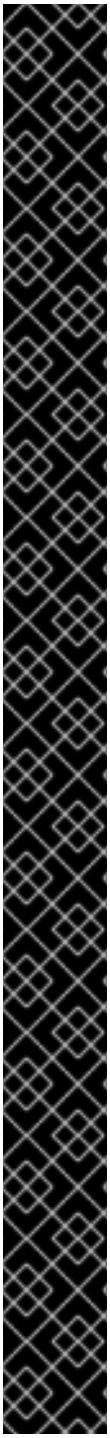
OpenShift Container Platform は、ノードの管理を単純化するために 3 つのデーモンセットとコントローラーを採用しています。これらのデーモンセットは、Kubernetes 形式のコンストラクトを使用してオペレーティングシステムの更新とホストの設定変更をオーケストレーションします。これには、以下が含まれます。

- **machine-config-controller** : コントロールプレーンからマシンのアップグレードを調整します。すべてのクラスターノードを監視し、その設定の更新をオーケストレーションします。
- **machine-config-daemon** デーモンセット: クラスターの各ノードで実行され、machine config で定義された設定で、MachineConfigController の指示通りにマシンを更新します。 ノードは、変更を検知すると Pod からドレイン (解放) され、更新を適用して再起動します。これらの変更は、指定されたマシン設定を適用し、kubelet 設定を制御する Ignition 設定ファイルの形式で実行されます。更新自体はコンテナーで行われます。このプロセスは、OpenShift Container Platform と RHCOS の更新を同時に管理する際に不可欠です。
- **machine-config-server** デーモンセット: コントロールプレーンノードがクラスターに参加する際に Ignition 設定ファイルをコントロールプレーンノードに提供します。

このマシン設定は Ignition 設定のサブセットです。**machine-config-daemon** はマシン設定を読み取り、OSTree の更新を行う必要があるか、一連の systemd kubelet ファイルの変更、設定の変更、オペレーティングシステムまたは OpenShift Container Platform 設定などへのその他の変更を適用する必要

があるかを確認します。

ノード管理操作の実行時に、**KubeletConfig** カスタムリソース (CR) を作成または変更します。



**重要**

マシン設定への変更が行われると、Machine Config Operator (MCO) は変更を有効にするために、対応するすべてのノードを自動的に再起動します。

ノード停止ポリシーを使用すると、一部のマシン設定の変更によって発生する停止を軽減できます。**マシン設定の変更後のノードの再起動動作について** を参照してください。

あるいは、変更を加える前に、マシン設定の変更後にノードが自動的に再起動しないようにすることもできます。対応するマシン設定プールで **spec.paused** フィールドを **true** に設定して、自動再起動プロセスを一時停止します。一時停止すると、**spec.paused** フィールドを **false** に設定し、ノードが新しい設定で再起動されるまで、マシン設定の変更は適用されません。

以下の変更は、ノードの再起動をトリガーしません。

- MCO が以下の変更のいずれかを検出すると、ノードのドレインまたは再起動を行わずに更新を適用します。
  - マシン設定の **spec.config.passwd.users.sshAuthorizedKeys** パラメーターの SSH キーの変更。
  - **openshift-config** namespace でのグローバルプルシークレットまたはプルシークレットへの変更
  - Kubernetes API Server Operator による **/etc/kubernetes/kubelet-ca.crt** 認証局 (CA) の自動ローテーション。
- MCO は、**/etc/containers/registries.conf** ファイルへの変更 (**ImageDigestMirrorSet**、**ImageTagMirrorSet**、または **ImageContentSourcePolicy** オブジェクトの追加や編集など) を検出すると、対応するノードをドレインし、変更を適用し、ノードの分離を解除します。次の変更ではノードドレインは発生しません。
  - **pull-from-mirror = "digest-only"** パラメーターがミラーごとに設定されたレジストリーの追加。
  - **pull-from-mirror = "digest-only"** パラメーターがレジストリーに設定されたミラーの追加。
  - **unqualified-search-registries** へのアイテムの追加。

ノードの設定が、現在適用されているマシン設定で指定されているものと完全に一致しない場合があります。この状態は **設定ドリフト** と呼ばれます。Machine Config Daemon (MCD) は、ノードの設定ドリフトを定期的にチェックします。MCD が設定のドリフトを検出した場合、管理者がノード設定を修正するまで、MCO はノードを **degraded** とマークします。degraded 状態のノードは、オンラインであり動作中ですが、更新することはできません。

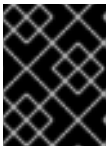
**関連情報**

- OpenShift SDN ネットワークプラグインについて

## 1.2. マシン設定の概要

Machine Config Operator (MCO) は systemd、CRI-O、Kubelet、カーネル、ネットワークマネージャーその他のシステム機能への更新を管理します。また、これはホストに設定ファイルを書き込むことができる **MachineConfig** CRD を提供します (machine-config-operator を参照)。MCO の機能や、これが他のコンポーネントとどのように対話するかを理解することは、詳細なシステムレベルの変更を OpenShift Container Platform クラスターに加える上で重要です。以下は、MCO、マシン設定、およびそれらの使用方法について知っておく必要のある点です。

- マシン設定は、名前のアルファベット順、辞書編集上の昇順に処理されます。レンダーコントローラーは、リストの最初のマシン設定をベースとして使用し、残りをベースマシン設定に追加します。
- マシン設定は、OpenShift Container Platform ノードのプールを表す各システムのオペレーティングシステムのファイルまたはサービスに特定の変更を加えることができます。
- MCO はマシンのプールのオペレーティングシステムに変更を適用します。すべての OpenShift Container Platform クラスターについては、ワーカーおよびコントロールプレーンノードプールから始まります。ロールラベルを追加することで、ノードのカスタムプールを設定できます。たとえば、アプリケーションが必要とする特定のハードウェア機能が含まれるワーカーノードのカスタムプールを設定できます。ただし、本セクションの例では、デフォルトのプールタイプの変更に重点を置いています。



**重要**

ノードには、**master** または **worker** などの複数のラベルを適用できますが、ノードを **単一の** マシン設定プールのメンバーにすることもできます。

- Machine Config Operator(MCO) は **topology.kubernetes.io/zone** ラベルに基づいて、ゾーンによってアルファベット順に影響を受けるノードを更新するようになりました。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、ノードは年齢別にアップグレードされ、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。
- 一部のマシン設定は、OpenShift Container Platform がディスクにインストールされる前に行われる必要があります。ほとんどの場合、これは、インストール後のマシン設定として実行されるのではなく、OpenShift Container Platform インストーラープロセスに直接挿入されるマシン設定を作成して実行できます。他の場合に、ノードごとの個別 IP アドレスの設定や高度なディスクのパーティション設定などを行うには、OpenShift Container Platform インストーラーの起動時にカーネル引数を渡すベアメタルのインストールを実行する必要がある場合があります。
- MCO はマシン設定で設定される項目を管理します。MCO が競合するファイルを管理することを明示的に指示されない限り、システムに手動で行う変更は MCO によって上書きされることはありません。つまり、MCO は要求される特定の更新のみを行い、ノード全体に対する制御を要求しません。
- ノードの手動による変更は推奨されません。ノードの使用を中止して新規ノードを起動する必要がある場合は、それらの直接的な変更は失われます。
- MCO は、**/etc** および **/var** ディレクトリーのファイルに書き込みを行う場合にのみサポートされます。ただし、これらの領域のいずれかにシンボリックリンクを指定して書き込み可能になるディレクトリーへのシンボリックリンクもあります。**/opt** および **/usr/local** ディレクトリーが例になります。

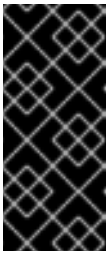
- Ignition は MachineConfig で使用される設定形式です。詳細は、Ignition 設定仕様 v3.2.0 を参照してください。
- Ignition 設定は OpenShift Container Platform のインストール時に直接提供でき、MCO が Ignition 設定を提供するのと同じ方法でフォーマットできますが、MCO では元の Ignition 設定を確認する方法がありません。そのため、それらをデプロイする前に Ignition 設定をマシン設定にラップする必要があります。
- MCO で管理されるファイルが MCO 外で変更されると、Machine Config Daemon (MCD) はノードを **degraded** として設定します。これは問題のあるファイルを上書きしませんが、継続して **degraded** 状態で動作します。
- マシン設定を使用する主な理由として、これは OpenShift Container Platform クラスターのプールに対して新規ノードを起動する際に適用されます。**machine-api-operator** は新規マシンをプロビジョニングし、MCO がこれを設定します。

MCO は Ignition を設定形式として使用します。OpenShift Container Platform バージョン 4.6 では、Ignition 設定仕様のバージョン 2 から 3 に移行しています。

### 1.2.1. マシン設定で変更できる設定

MCO で変更できるコンポーネントの種類には、以下が含まれます。

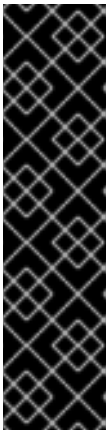
- **config**: ignition 設定オブジェクト (Ignition 設定仕様 を参照してください) を作成し、以下を含む OpenShift Container Platform マシン上でのファイル、systemd サービスおよびその他の機能の変更などを実行できます。
  - **Configuration files**: **/var** または **/etc** ディレクトリーでファイルを作成するか、上書きします。
  - **systemd units**: systemd サービスを作成し、そのステータスを設定するか、追加設定により既存の systemd サービスに追加します。
  - **users and groups**: インストール後に passwd セクションで SSH キーを変更します。



**重要**

- マシン設定を使用した SSH キーの変更は、**core** ユーザーにのみサポートされています。
- マシン設定を使用した新しいユーザーの追加はサポートされていません。

- **kernelArguments**: OpenShift Container Platform ノードの起動時に、引数をカーネルコマンドラインに追加します。
- **kernelType**: オプションで、標準カーネルの代わりに使用する標準以外のカーネルを特定します。(RAN の) RT カーネルを使用するには、**realtime** を使用します。これは一部のプラットフォームでのみサポートされます。**64k-pages** パラメーターを使用して、64k ページサイズのカーネルを有効にします。この設定は、64 ビット ARM アーキテクチャーを使用するマシンに限定されます。
- **fips**: FIPS モードを有効にします。FIPS は、インストール後の手順ではなく、インストール時に設定する必要があります。



**重要**

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、FIPS モードでのシステムのインストール を参照してください。

FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、x86_64、ppc64le、および s390x アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

- **extensions**: 事前にパッケージ化されたソフトウェアを追加して RHCOS 機能を拡張します。この機能については、利用可能な拡張機能には usbguard およびカーネルモジュールが含まれます。

- **カスタムリソース (ContainerRuntime および Kubelet 用)**: マシン設定外で、MCO は CRI-O コンテナーランタイムの設定 (**ContainerRuntime** CR) および Kubelet サービス ( **Kubelet** CR) を変更するために 2 つの特殊なカスタムリソースを管理します。

MCO は、OpenShift Container Platform ノードでオペレーティングシステムコンポーネントを変更できる唯一の Operator という訳ではありません。他の Operator もオペレーティングシステムレベルの機能を変更できます。1 つの例として、Node Tuning Operator を使用して、Tuned デーモンプロファイルを使用したノードレベルのチューニングを実行できます。

インストール後に実行できる MCO 設定のタスクは、次の手順に記載されています。OpenShift Container Platform のインストール時またはインストール前に実行する必要のあるシステム設定タスクについては、RHCOS ベアメタルのインストールについての説明を参照してください。デフォルトでは、MCO で行う多くの変更には再起動が必要です。

以下の変更は、ノードの再起動をトリガーしません。

- MCO が以下の変更のいずれかを検出すると、ノードのドレインまたは再起動を行わずに更新を適用します。
    - マシン設定の **spec.config.passwd.users.sshAuthorizedKeys** パラメーターの SSH キーの変更。
    - **openshift-config** namespace でのグローバルプルシークレットまたはプルシークレットへの変更
    - Kubernetes API Server Operator による **/etc/kubernetes/kubelet-ca.crt** 認証局 (CA) の自動ローテーション。
- MCO は、**/etc/containers/registries.conf** ファイルへの変更 (**ImageDigestMirrorSet**、**ImageTagMirrorSet**、または **ImageContentSourcePolicy** オブジェクトの追加や編集など) を検出すると、対応するノードをドレインし、変更を適用し、ノードの分離を解除します。次の変更ではノードドレインは発生しません。
    - **pull-from-mirror = "digest-only"** パラメーターがミラーごとに設定されたレジストリーの追加。
    - **pull-from-mirror = "digest-only"** パラメーターがレジストリーに設定されたミラーの追加。
    - **unqualified-search-registries** へのアイテムの追加。

その他のケースでは、**ノード停止ポリシー** を使用することで、MCO が変更を加えたときにワークロードの停止を軽減できます。詳細は、**マシン設定の変更後のノード再起動動作について** を参照してください。

ノードの設定が、現在適用されているマシン設定で指定されているものと完全に一致しない場合があります。この状態は **設定ドリフト** と呼ばれます。Machine Config Daemon (MCD) は、ノードの設定ドラフトを定期的にチェックします。MCD が設定のドリフトを検出した場合、管理者がノード設定を修正するまで、MCO はノードを **degraded** とマークします。degraded 状態のノードは、オンラインであり動作中ですが、更新することはできません。設定ドリフトの詳細は、**Understanding configuration drift detection** を参照してください。

## 1.3. MACHINE CONFIG OPERATOR ノードのドレイン動作について

マシン設定を使用して、新しい設定ファイルの追加、systemd ユニットまたはカーネル引数の変更、SSH キーの更新などのシステム機能を変更すると、Machine Config Operator (MCO) がそれらの変更を適用し、各ノードが目的の設定状態にあることを確認します。

変更を加えると、MCO は新しくレンダリングされたマシン設定を生成します。ほとんどの場合、MCO は、新しくレンダリングされたマシン設定を適用するときに、影響を受けるすべてのノードの設定が更新されるまで、影響を受ける各ノードで次の手順を実行します。

1. 遮断。MCO は、ノードを追加のワークロードに対してスケジュール不可としてマークします。
2. ドレイン。MCO は、ノード上で実行中のすべてのワークロードを終了します。その結果、ワークロードが他のノードに再スケジュールされます。
3. 適用。MCO は、必要に応じて新しい設定をノードに書き込みます。
4. 再起動します。MCO はノードを再起動します。
5. 遮断解除。MCO は、ノードをワークロードに対してスケジュール可能としてマークします。

このプロセス全体を通じて、MCO はマシン設定プールで設定された **MaxUnavailable** 値に基づいて必要な数の Pod を維持します。

MCO がマスターノード上の Pod をドレインする場合は、次の条件に注意してください。

- シングルノードの OpenShift クラスターでは、MCO はドレイン操作をスキップします。
- MCO は、etcd などのサービスへの干渉を防ぐために、静的 Pod をドレインしません。

**注記**

場合によっては、ノードがドレインされないことがあります。詳細は、「Machine Config Operator について」を参照してください。

ノード停止ポリシーを使用するか、コントロールプレーンの再起動を無効にすることで、ドレインと再起動のサイクルによって引き起こされる停止を軽減できます。詳細は、「マシン設定の変更後のノード再起動動作について」および「Machine Config Operator の自動再起動の無効化」を参照してください。

**関連情報**

- Machine Config Operator について

- ノード停止ポリシーを使用してマシン設定の変更による停止を最小限に抑える
- Machine Config Operator の自動再起動の無効化

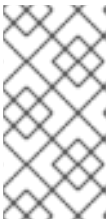
## 1.4. 設定ドリフト検出について

ノードのディスク上の状態がマシン設定で設定される内容と異なる場合があります。これは、**設定ドリフト** と呼ばれます。たとえば、クラスター管理者は、マシン設定で設定されたファイル、systemd ユニットファイル、またはファイルパーミッションを手動で変更する場合があります。これにより、設定のドリフトが発生します。設定ドリフトにより、Machine Config Pool のノード間で問題が発生したり、マシン設定が更新されると問題が発生したりする可能性があります。

Machine Config Operator (MCO) は Machine Config Daemon (MCD) を使用して、設定ドリフトがないかノードを定期的に確認します。検出されると、MCO はノードおよびマシン設定プール (MCP) を **Degraded** に設定し、エラーを報告します。degraded 状態のノードは、オンラインであり動作中ですが、更新することはできません。

MCD は、以下の状況の時に設定ドリフトの検出を実行します。

- ノードがブートする時。
- マシン設定で指定されたファイル (Ignition ファイルと systemd ドロップインユニット) がマシン設定以外で変更された時。
- 新しいマシン設定が適用される前。



**注記**

新しいマシン設定をノードに適用すると、MCD は設定ドリフトの検出を一時的に停止します。新しいマシン設定はノード上のマシン設定とは必ず異なるため、この停止処理が必要です。新しいマシン設定が適用された後に、MCD は新しいマシン設定を使用して設定ドリフトの検出を再開します。

設定ドリフトの検出を実行する際に、MCD はファイルの内容とパーミッションが、現在適用されているマシン設定で指定されるものに完全に一致することを確認します。通常、MCD は検出がトリガーされてから 2 秒未満で設定ドリフトを検出します。

MCD が設定ドリフトを検出すると、MCD は以下のタスクを実行します。

- コンソールログにエラーを出力する
- Kubernetes イベントを生成する
- ノードでのそれ以上の検出を停止する
- ノードおよび MCP を **degraded**に設定する

MCP をリスト表示して、パフォーマンスが低下したノードがあるかどうかを確認できます。

```
$ oc get mcp worker
```

パフォーマンスが低下した MCP がある場合、以下の出力のように**DEGRADEDMACHINECOUNT** フィールドの値がゼロ以外になります。

**出力例**

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
worker   rendered-worker-404caf3180818d8ac1f50c32f14b57c3   False     True       True       2
1                 1                   1                   5h51m
```

マシン設定プールを調べることで、設定ドリフトによって問題が発生しているかどうかを判別できます。

```
$ oc describe mcp worker
```

**出力例**

```
 ...
    Last Transition Time:  2021-12-20T18:54:00Z
    Message:               Node ci-ln-j4h8nkb-72292-pxqxz-worker-a-fjks4 is reporting: "content mismatch
for file \"/etc/mco-test-file\"" 1
    Reason:                1 nodes are reporting degraded status on sync
    Status:                True
    Type:                  NodeDegraded 2
 ...
```

1 このメッセージは、マシン設定によって追加されたノードの **/etc/mco-test-file** ファイルが、マシン設定外で変更されていることを示しています。

2 ノードの状態は **NodeDegraded** です。

あるいは、パフォーマンスが低下しているノードが分かっている場合は、そのノードを確認します。

```
$ oc describe node/ci-ln-j4h8nkb-72292-pxqxz-worker-a-fjks4
```

**出力例**

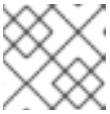
```
 ...

Annotations:        cloud.network.openshift.io/egress-ipconfig: [{"interface":"nic0","ifaddr":
{"ipv4":"10.0.128.0/17"},"capacity":{"ip":10}}]
                    csi.volume.kubernetes.io/nodeid:
                      {"pd.csi.storage.gke.io":"projects/openshift-gce-devel-ci/zones/us-central1-
a/instances/ci-ln-j4h8nkb-72292-pxqxz-worker-a-fjks4"}
                    machine.openshift.io/machine: openshift-machine-api/ci-ln-j4h8nkb-72292-pxqxz-worker-
a-fjks4
                    machineconfiguration.openshift.io/controlPlaneTopology: HighlyAvailable
                    machineconfiguration.openshift.io/currentConfig: rendered-worker-
67bd55d0b02b0f659aef33680693a9f9
                    machineconfiguration.openshift.io/desiredConfig: rendered-worker-
67bd55d0b02b0f659aef33680693a9f9
                    machineconfiguration.openshift.io/reason: content mismatch for file "/etc/mco-test-file"
1
                    machineconfiguration.openshift.io/state: Degraded 2
 ...
```

1 エラーメッセージは、ノードとリスト表示されたマシン設定の間で設定ドリフトが検出されたことを示しています。このエラーメッセージは、マシン設定によって追加された **/etc/mco-test-file** の

2 ノードの状態は **Degraded** です。

以下の修復策のいずれかを実行して、設定ドリフトを修正し、ノードを **Ready** の状態に戻すことができます。

- ノード上のファイルの内容とパーミッションがマシン設定で設定される内容と一致するようにします。手動でファイルの内容を書き換えたり、ファイルパーミッション変更したりすることができます。
- パフォーマンスが低下したノードで 強制ファイル を生成します。強制ファイルにより、MCD は通常の設定ドリフトの検出をバイパスし、現在のマシン設定を再度適用します。



**注記**

ノード上で強制ファイルを生成すると、そのノードが再起動されます。

## 1.5. マシン設定プールのステータスの確認

Machine Config Operator (MCO)、そのサブコンポーネント、およびこれが管理するリソースのステータスを表示するには、以下の **oc** コマンドを使用します。

**手順**

1. 各マシン設定プール (MCP) のクラスターで使用可能な MCO 管理ノードの数を確認するには、次のコマンドを実行します。

```
$ oc get machineconfigpool
```

**出力例**

```
NAME     CONFIG                 UPDATED  UPDATING   DEGRADED  MACHINECOUNT
READYMACHINECOUNT  UPDATEDMACHINECOUNT DEGRADEDMACHINECOUNT
AGE
master   rendered-master-06c9c4…  True     False      False     3          3              3
0                  4h42m
worker   rendered-worker-f4b64…   False    True       False     3          2              2
0                  4h42m
```

ここでは、以下のようになります。

UPDATED

**True** ステータスは、MCO が現在のマシン設定をその MCP のノードに適用したことを示します。現在のマシン設定は、**oc get mcp** 出力の **STATUS** フィールドに指定されています。**False** ステータスは、MCP 内のノードが更新中であることを示します。

UPDATING

**True** ステータスは、MCO が、**MachineConfigPool** カスタムリソースで指定された目的のマシン設定を、その MCP 内の少なくとも 1 つのノードに適用していることを示します。目的のマシン設定は、新しく編集されたマシン設定です。更新中のノードは、スケジューリン

グに使用できない場合があります。**False** ステータスは、MCP 内のすべてのノードが更新されたことを示します。

DEGRADED

**True** ステータスは、MCO がその MCP 内の少なくとも 1 つのノードに現在のまたは目的のマシン設定を適用することをブロックされているか、設定が失敗していることを示します。機能が低下したノードは、スケジューリングに使用できない場合があります。**False** ステータスは、MCP 内のすべてのノードの準備ができていることを示します。

MACHINECOUNT

その MCP 内のマシンの総数を示します。

READYMACHINECOUNT

スケジューリングの準備ができているその MCP 内のマシンの総数を示します。

UPDATEDMACHINECOUNT

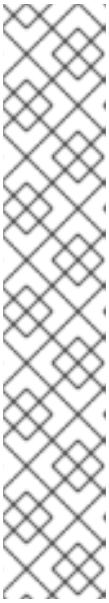
現在のマシン設定を持つ MCP 内のマシンの総数を示します。

DEGRADEDMACHINECOUNT

機能低下または調整不能としてマークされている、その MCP 内のマシンの総数を示します。

前の出力では、3 つのコントロールプレーン (マスター) ノードと 3 つのワーカーノードがあります。コントロールプレーン MCP と関連するノードは、現在のマシン設定に更新されます。ワーカー MCP のノードは、目的のマシン設定に更新されています。**UPDATEDMACHINECOUNT** が **2** であることからわかるように、ワーカー MCP 内の 2 つのノードが更新され、1 つがまだ更新中です。**DEGRADEDDMACHINECOUNT** が **0** で、**DEGRADE** が **False** であることからわかるように、問題はありません。

MCP のノードが更新されている間、**CONFIG** の下にリストされているマシン設定は、MCP の更新元である現在のマシン設定です。更新が完了すると、リストされたマシン設定は、MCP が更新された目的のマシン設定になります。



**注記**

ノードが遮断されている場合、そのノードは **READYMACHINECOUNT** には含まれませんが、**MACHINECOUNT** には含まれます。また、MCP ステータスは **UPDATING** に設定されます。ノードには現在のマシン設定があるため、**UPDATEDMACHINECOUNT** の合計にカウントされます。

**出力例**

```
NAME     CONFIG                 UPDATED  UPDATING   DEGRADED
MACHINECOUNT  READYMACHINECOUNT  UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT  AGE
master   rendered-master-06c9c4…  True     False      False    3          3
3                0                4h42m
worker   rendered-worker-c1b41a…  False    True       False    3          2
3                0                4h42m
```

2. **MachineConfigPool** カスタムリソースを調べて MCP 内の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 describe mcp worker
```

**出力例**

```
...
  Degraded Machine Count:     0
  Machine Count:              3
  Observed Generation:        2
  Ready Machine Count:        3
  Unavailable Machine Count:  0
  Updated Machine Count:      3
Events:                       <none>
```

**注記**

ノードが遮断されている場合、そのノードは **Ready Machine Count** に含まれません。**Unavailable Machine Count** に含まれます。

**出力例**

```
...
  Degraded Machine Count:     0
  Machine Count:              3
  Observed Generation:        2
  Ready Machine Count:        2
  Unavailable Machine Count:  1
  Updated Machine Count:      3
```

3. 既存の各 **MachineConfig** オブジェクトを表示するには、次のコマンドを実行します。

```
$ oc get machineconfigs
```

**出力例**

```
NAME                        GENERATEDBYCONTROLLER          IGNITIONVERSION  AGE
00-master                   2c9371fbb673b97a6fe8b1c52...   3.2.0            5h18m
00-worker                   2c9371fbb673b97a6fe8b1c52...   3.2.0            5h18m
01-master-container-runtime      2c9371fbb673b97a6fe8b1c52...   3.2.0            5h18m
01-master-kubelet                2c9371fbb673b97a6fe8b1c52…     3.2.0            5h18m
...
rendered-master-dde...           2c9371fbb673b97a6fe8b1c52...   3.2.0            5h18m
rendered-worker-fde...           2c9371fbb673b97a6fe8b1c52...   3.2.0            5h18m
```

**rendered** として一覧表示された **MachineConfig** オブジェクトが変更されたり、削除されたりすることが意図されていないことに注意してください。

4. 特定のマシン設定 (この場合は **01-master-kubelet**) の内容を表示するには、次のコマンドを実行します。

```
$ oc describe machineconfigs 01-master-kubelet
```

コマンドの出力は、この **MachineConfig** オブジェクトに設定ファイル (**cloud.conf** および **kubelet.conf**) と systemd サービス (Kubernetes Kubelet) の両方が含まれていることを示しています。

**出力例**

```
Name:         01-master-kubelet
...
Spec:
  Config:
    Ignition:
      Version:  3.2.0
    Storage:
      Files:
        Contents:
          Source:   data:,
        Mode:       420
        Overwrite:  true
        Path:       /etc/kubernetes/cloud.conf
        Contents:
          Source:
data:,kind%3A%20KubeletConfiguration%0AapiVersion%3A%20kubelet.config.k8s.io%2Fv1beta1%0Aauthentication%3A%0A%20%20x509%3A%0A%20%20%20%20clientCAFile%3A%20%2Fetc%2Fkubernetes%2Fkubelet-ca.crt%0A%20%20anonymous...
        Mode:       420
        Overwrite:  true
        Path:       /etc/kubernetes/kubelet.conf
    Systemd:
      Units:
        Contents:  [Unit]
Description=Kubernetes Kubelet
Wants=rpc-statd.service network-online.target crio.service
After=network-online.target crio.service

ExecStart=/usr/bin/hyperkube \
    kubelet \
      --config=/etc/kubernetes/kubelet.conf \ ...
```

適用するマシン設定で問題が発生した場合は、この変更を常に取り消すことができます。たとえば、**oc create -f ./myconfig.yaml** を実行してマシン設定を適用した場合、次のコマンドを実行してそのマシン設定を削除できます。

```
$ oc delete -f ./myconfig.yaml
```

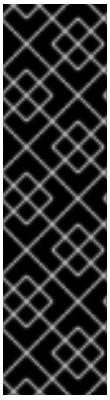
これが唯一の問題である場合、影響を受けるプールのノードは動作が低下していない状態に戻るはずです。これにより、レンダリングされた設定は、直前のレンダリングされた状態にロールバックされます。

独自のマシン設定をクラスターに追加する場合、直前の例に示されたコマンドを使用して、それらのステータスと、それらが適用されるプールの関連するステータスを確認できます。

## 1.6. マシン設定ノードのステータスの確認

更新中は、問題が発生してノードのトラブルシューティングが必要になる場合に備えて、個々のノードの進行状況を監視することをお勧めします。

クラスターに対する Machine Config Operator (MCO) 更新のステータスを確認するには、次の **oc** コマンドを使用します。



**重要**

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

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

**手順**

1. 次のコマンドを実行して、すべてのマシン設定プールに含まれる全ノードの更新ステータスの概要を取得します。

```
$ oc get machineconfignodes
```

**出力例**

```
NAME                         UPDATED   UPDATEPREPARED   UPDATEEXECUTED   UPDATEPOSTACTIONCOMPLETED   UPDATECOMPLETED   RESUMED
ip-10-0-12-194.ec2.internal   True      False            False            False                       False             False
ip-10-0-17-102.ec2.internal   False     True             False            False                       False             False
ip-10-0-2-232.ec2.internal    False     False            True             False                       False             False
ip-10-0-59-251.ec2.internal   False     False            False            True                        False             False
ip-10-0-59-56.ec2.internal    False     False            False            False                       True              True
ip-10-0-6-214.ec2.internal    False     False            Unknown          False                       False             False
```

ここでは、以下のようになります。

UPDATED

**True** ステータスは、MCO が現在のマシン設定をその特定のノードに適用したことを示します。**False** ステータスは、ノードが現在更新中であることを示します。**Unknown** ステータスは、操作が処理中であることを表します。

UPDATEPREPARED

**False** ステータスは、配布される新しいマシン設定の調整を MCO が開始していないことを示します。**True** ステータスは、MCO が更新のこのフェーズを完了したことを示します。**Unknown** ステータスは、操作が処理中であることを表します。

UPDATEEXECUTED

**False** ステータスは、MCO がノードの遮断とドレインを開始していないことを示します。また、ディスクの状態とオペレーティングシステムの更新が開始されていないことも示します。**True** ステータスは、MCO が更新のこのフェーズを完了したことを示します。**Unknown** ステータスは、操作が処理中であることを表します。

UPDATEPOSTACTIONCOMPLETED

**False** ステータスは、MCO がノードの再起動またはデーモンの終了を開始していないこと

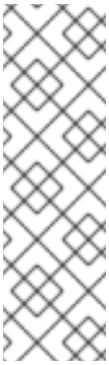
を示します。**True** ステータスは、MCO が再起動とノードステータスの更新を完了したことを示します。**Unknown** ステータスは、このフェーズの更新プロセス中にエラーが発生したか、MCO が現在更新を適用していることを示します。

UPDATECOMPLETED

**False** ステータスは、MCO がノードの遮断解除とノードの状態およびメトリクスの更新を開始していないことを示します。**True** ステータスは、MCO がノードの状態と利用可能なメトリクスの更新を完了したことを示します。

RESUMED

**False** ステータスは、MCO が設定ドリフトモニターを起動していないことを示します。**True** ステータスは、ノードが動作を再開したことを示します。**Unknown** ステータスは、操作が処理中であることを表します。



**注記**

上記の 1 次フェーズには、2 次フェーズを含めることができます。2 次フェーズを使用すると、更新の進行状況をより詳細に確認できます。前述のコマンドの **-o Wide** オプションを使用すると、更新の 2 次フェーズを含む詳細情報を取得できます。これにより、**UPDATECOMPATIBLE**、**UPDATEFILESANDOS**、**DRAINEDNODE**、**CORDONEDNODE**、**REBOOTNODE**、**RELOADEDCRIO**、および **UNCORDONED** 列が追加されます。これらの 2 次フェーズは常に発生するわけではなく、適用する更新の種類によって異なります。

2. 次のコマンドを実行して、特定のマシン設定プールに含まれるノードの更新ステータスを確認します。

```
$ oc get machineconfignodes $(oc get machineconfignodes -o json | jq -r
'.items[]|select(.spec.pool.name=="<pool_name>")|.metadata.name') 1
```

1 プールの名前は **MachineConfigPool** オブジェクト名です。

**出力例**

```
NAME                        UPDATED   UPDATEPREPARED   UPDATEEXECUTED
UPDATEPOSTACTIONCOMPLETE   UPDATECOMPLETE   RESUMED
ip-10-0-48-226.ec2.internal   True      False          False          False                 False
False
ip-10-0-5-241.ec2.internal    True      False          False          False                 False
False
ip-10-0-74-108.ec2.internal   True      False          False          False                 False
False
```

3. 次のコマンドを実行して、個々のノードの更新ステータスを確認します。

```
$ oc describe machineconfignode/<node_name> 1
```

1 ノードの名前は **MachineConfigNode** オブジェクト名です。

**出力例**

-

```
Name:         <node_name>
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  machineconfiguration.openshift.io/v1alpha1
Kind:         MachineConfigNode
Metadata:
  Creation Timestamp:  2023-10-17T13:08:58Z
  Generation:          1
  Resource Version:    49443
  UID:                 4bd758ab-2187-413c-ac42-882e61761b1d
Spec:
  Node Ref:
    Name:        <node_name>
  Pool:
    Name:        master
  ConfigVersion:
    Desired: rendered-worker-823ff8dc2b33bf444709ed7cd2b9855b (1)
Status:
  Conditions:
    Last Transition Time:  2023-10-17T13:09:02Z
    Message:               Node has completed update to config rendered-master-
cf99e619747ab19165f11e3546c71f1e
    Reason:                NodeUpgradeComplete
    Status:                True
    Type:                  Updated
    Last Transition Time:  2023-10-17T13:09:02Z
    Message:               This node has not yet entered the UpdatePreparing phase
    Reason:                NotYetOccured
    Status:                False
  Config Version:
    Current:           rendered-worker-823ff8dc2b33bf444709ed7cd2b9855b
    Desired:           rendered-worker-823ff8dc2b33bf444709ed7cd2b9855b (2)
  Health:              Healthy
  Most Recent Error:
  Observed Generation:  3
```

(1) **spec.configversion.desired** フィールドで指定し必要な設定は、ノード上で新しい設定が検出されるとすぐに更新されます。

(2) **status.configversion.desired** フィールドに指定した必要な設定は、新しい設定が Machine Config Daemon (MCD) によって検証された場合にのみ更新されます。MCD は、更新の現在のフェーズをチェックすることによって検証を実行します。更新が **UPDATEPREPARED** フェーズを正常に通過すると、ステータスに新しい設定が追加されます。

## 1.7. MACHINE CONFIG OPERATOR 証明書について

Machine Config Operator 証明書は、Red Hat Enterprise Linux CoreOS (RHCOS) ノードと Machine Config Server 間の接続を保護するために使用されます。詳細は、Machine Config Operator 証明書 を参照してください。

### 1.7.1. 証明書の表示と操作

次の証明書はクラスター内で Machine Config Controller (MCC) によって処理され、**ControllerConfig** リソースにあります。

- **/etc/kubernetes/kubelet-ca.crt**
- **/etc/kubernetes/static-pod-resources/configmaps/cloud-config/ca-bundle.pem**
- **/etc/pki/ca-trust/source/anchors/openshift-config-user-ca-bundle.crt**

MCC は、イメージレジストリー証明書とそれに関連するユーザーバンドル証明書も処理します。

証明書の元となる基礎バンドル、署名データとサブジェクトデータなど、リストされた証明書に関する情報を取得できます。

**手順**

- 次のコマンドを実行して、詳細な証明書情報を取得します。

```
$ oc get controllerconfig/machine-config-controller -o yaml | yq -y
'.status.controllerCertificates'
```

**出力例**

```
"controllerCertificates": [
                {
                    "bundleFile": "KubeAPIServerServingCAData",
                    "signer": "<signer_data1>",
                    "subject": "CN=openshift-kube-apiserver-operator_node-system-admin-
signer@168909215"
                },
                {
                    "bundleFile": "RootCAData",
                    "signer": "<signer_data2>",
                    "subject": "CN=root-ca,OU=openshift"
                }
            ]
```

- 次のコマンドを使用してマシン設定プールのステータスを確認し、ControllerConfig で見つかった情報のより単純なバージョンを取得します。

```
$ oc get mcp master -o yaml | yq -y '.status.certExpirys'
```

**出力例**

```
status:
  certExpirys:
  - bundle: KubeAPIServerServingCAData
    subject: CN=admin-kubeconfig-signer,OU=openshift
  - bundle: KubeAPIServerServingCAData
    subject: CN=kube-csr-signer_@1689585558
  - bundle: KubeAPIServerServingCAData
    subject: CN=kubelet-signer,OU=openshift
  - bundle: KubeAPIServerServingCAData
```

```
    subject: CN=kube-apiserver-to-kubelet-signer,OU=openshift
  - bundle: KubeAPIServerServingCAData
    subject: CN=kube-control-plane-signer,OU=openshift
```

このメソッドは、マシン設定プール情報をすでに使用している OpenShift Container Platform アプリケーションを対象としています。

- **/etc/docker/cert.d** ディレクトリーの内容を調べて、どのイメージレジストリー証明書がノード上にあるかを確認します。

```
# ls /etc/docker/certs.d
```

**出力例**

```
image-registry.openshift-image-registry.svc.cluster.local:5000 image-registry.openshift-image-registry.svc:5000
```

# 第2章 マシン設定オブジェクトを使用したノードの設定

このセクションのタスクを使用して、**MachineConfig** オブジェクトを作成し、OpenShift Container Platform ノードで実行されているファイル、systemd ユニットファイルその他のオペレーティングシステム機能を変更することができます。マシン設定の使用に関する詳細は、SSH 認証キーの 更新、イメージ署名の検証、SCTP の有効化、および OpenShift Container Platform の iSCSI イニシエーター名の設定 に関するコンテンツを参照してください。

OpenShift Container Platform は Ignition 仕様バージョン 3.2 をサポートします。今後作成する新規のマシン設定はすべて Ignition 仕様バージョン 3.2 をベースとする必要があります。OpenShift Container Platform クラスターをアップグレードする場合、既存の Ignition 仕様バージョン 2.x マシン設定は仕様バージョン 3.2 に自動的に変換されます。

ノードの設定が、現在適用されているマシン設定で指定されているものと完全に一致しない場合があります。この状態は **設定ドリフト** と呼ばれます。Machine Config Daemon (MCD) は、ノードの設定ドラフトを定期的にチェックします。MCD が設定のドリフトを検出した場合、管理者がノード設定を修正するまで、MCO はノードを **degraded** とマークします。degraded 状態のノードは、オンラインであり動作中ですが、更新することはできません。設定ドリフトの詳細は、Understanding configuration drift detection を参照してください。

### ヒント

他の設定ファイルを OpenShift Container Platform ノードに追加する方法については、以下の「chrony タイムサービスの設定」の手順をモデルとして使用します。

## 2.1. CHRONY タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定できます。

### 手順

1. **chrony.conf** ファイルのコンテンツを含む Butane 設定を作成します。たとえば、ワーカーノードで chrony を設定するには、**99-worker-chrony.bu** ファイルを作成します。

**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。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
```

```
    inline: |
      pool 0.rhel.pool.ntp.org iburst 4
      driftfile /var/lib/chrony/drift
      makestep 1.0 3
      rtcsync
      logdir /var/log/chrony
```

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。

3 マシン設定ファイルの **mode** フィールドに 8 進数の値でモードを指定します。ファイルを作成し、変更を適用すると、**mode** は 10 進数の値に変換されます。コマンド **oc get mc <mc-name> -o yaml** で YAML ファイルを確認できます。

4 DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。または、NTP サーバーの **1.rhel.pool.ntp.org**、**2.rhel.pool.ntp.org**、または **3.rhel.pool.ntp.org** のいずれかを指定できます。

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-chrony.yaml**) を生成します。

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.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-chrony.yaml
```

**関連情報**

- Butane でのマシン設定の作成

## 2.2. CHRONY タイムサービスの無効化

**MachineConfig** カスタムリソース (CR) を使用して、特定のロールを持つノードの chrony タイムサービス (**chronyd**) を無効にすることができます。

**前提条件**

- OpenShift 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。

**手順**

1. 指定されたノードロールの **chronyd** を無効にする **MachineConfig** CR を作成します。
   a. 以下の YAML を **disable-chronyd.yaml** ファイルに保存します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: <node_role> (1)
  name: disable-chronyd
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - contents: |
            [Unit]
            Description=NTP client/server
            Documentation=man:chronyd(8) man:chrony.conf(5)
            After=ntpdate.service sntp.service ntpd.service
            Conflicts=ntpd.service systemd-timesyncd.service
            ConditionCapability=CAP_SYS_TIME
            [Service]
            Type=forking
            PIDFile=/run/chrony/chronyd.pid
            EnvironmentFile=-/etc/sysconfig/chronyd
            ExecStart=/usr/sbin/chronyd $OPTIONS
            ExecStartPost=/usr/libexec/chrony-helper update-daemon
            PrivateTmp=yes
            ProtectHome=yes
            ProtectSystem=full
            [Install]
            WantedBy=multi-user.target
          enabled: false
          name: "chronyd.service"
```

(1) **chronyd** を無効にするノードロール (例: **master**)。

b. 以下のコマンドを実行して **MachineConfig** CR を作成します。

```
$ oc create -f disable-chronyd.yaml
```

## 2.3. カーネル引数のノードへの追加

特殊なケースとして、クラスターのノードセットにカーネル引数を追加する必要がある場合があります。これは十分に注意して実行する必要があり、設定する引数による影響を十分に理解している必要があります。

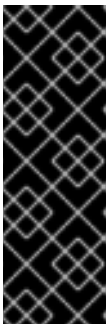
**警告**

カーネル引数を正しく使用しないと、システムが起動不可能になる可能性があります。

設定可能なカーネル引数の例には、以下が含まれます。

- **nosmt**: カーネルの対称マルチスレッド (SMT) を無効にします。マルチスレッドは、各 CPU の複数の論理スレッドを許可します。潜在的なクロススレッド攻撃に関連するリスクを減らすために、マルチテナント環境での **nosmt** の使用を検討できます。SMT を無効にすることは、基本的にパフォーマンスよりもセキュリティーを重視する選択をしていることになります。

- **systemd.unified_cgroup_hierarchy** : Linux コントロールグループバージョン 2 (cgroup v2) を有効にします。cgroup v2 は、カーネル コントロールグループ の次のバージョンであり、複数の改善を提供します。



**重要**

cgroup v1 は非推奨の機能です。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

OpenShift Container Platform で非推奨となったか、削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

- **enforcing=0**: SELinux (Security Enhanced Linux) を Permissive モードで実行するように設定します。Permissive モードでは、システムは、SELinux が読み込んだセキュリティーポリシーを実行しているかのように動作します。これには、オブジェクトのラベル付けや、アクセスを拒否したエントリーをログに出力するなどの動作が含まれますが、いずれの操作も拒否される訳ではありません。Permissive モードは、実稼働システムでの使用はサポートされませんが、デバッグには役に立ちます。

**警告**

実稼働環境の RHCOS での SELinux の無効化はサポートされていません。ノード上で SELinux が無効になったら、再プロビジョニングしてから実稼働クラスターに再び追加する必要があります。

カーネル引数の一覧と説明については、Kernel.org カーネルパラメーター を参照してください。

次の手順では、以下を特定する **MachineConfig** オブジェクトを作成します。

- カーネル引数を追加する一連のマシン。この場合、ワーカーロールを持つマシン。
- 既存のカーネル引数の最後に追加されるカーネル引数。
- マシン設定のリストで変更が適用される場所を示すラベル。

**前提条件**

- 作業用の OpenShift Container Platform クラスターに対する管理者権限が必要です。

**手順**

1. OpenShift Container Platform クラスターの既存の **MachineConfig** をリスト表示し、マシン設定にラベルを付ける方法を判別します。

```
$ oc get MachineConfig
```

**出力例**

```
NAME                                               GENERATEDBYCONTROLLER
IGNITIONVERSION   AGE
00-master                                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0
33m
00-worker                                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0
33m
01-master-container-runtime                              52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-master-kubelet                                        52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-worker-container-runtime                              52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-worker-kubelet                                        52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-master-generated-registries                           52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-master-ssh                                                                3.2.0             40m
99-worker-generated-registries                           52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-worker-ssh                                                                3.2.0             40m
rendered-master-23e785de7587df95a4b517e0647e5ab7
52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0             33m
rendered-worker-5d596d9293ca3ea80c896a1191735bb1
52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0             33m
```

2. カーネル引数を識別する **MachineConfig** オブジェクトファイルを作成します (例: **05-worker-kernelarg-selinuxpermissive.yaml**)。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker 1
  name: 05-worker-kernelarg-selinuxpermissive 2
spec:
  kernelArguments:
    - enforcing=0 3
```

1. 新しいカーネル引数をワーカーノードのみに適用します。
2. マシン設定 (05) 内の適切な場所を特定するための名前が指定されます (SELinux permissive モードを設定するためにカーネル引数を追加します)。
3. 正確なカーネル引数を **enforcing=0** として特定します。

3. 新規のマシン設定を作成します。

```
$ oc create -f 05-worker-kernelarg-selinuxpermissive.yaml
```

4. マシン設定で新規の追加内容を確認します。

```
$ oc get MachineConfig
```

**出力例**

```
NAME                                               GENERATEDBYCONTROLLER
IGNITIONVERSION   AGE
00-master                                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0
33m
00-worker                                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0
33m
01-master-container-runtime                              52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-master-kubelet                                  52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-worker-container-runtime                              52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-worker-kubelet                                  52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
05-worker-kernelarg-selinuxpermissive                                                3.2.0             105s
99-master-generated-registries                     52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-master-ssh                                                                        3.2.0             40m
99-worker-generated-registries                     52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-worker-ssh                                                                        3.2.0             40m
rendered-master-23e785de7587df95a4b517e0647e5ab7
52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0             33m
rendered-worker-5d596d9293ca3ea80c896a1191735bb1
52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0             33m
```

5. ノードを確認します。

```
$ oc get nodes
```

**出力例**

```
NAME                         STATUS                     ROLES    AGE   VERSION
ip-10-0-136-161.ec2.internal   Ready                      worker   28m   v1.29.4
ip-10-0-136-243.ec2.internal   Ready                      master   34m   v1.29.4
ip-10-0-141-105.ec2.internal   Ready,SchedulingDisabled   worker   28m   v1.29.4
ip-10-0-142-249.ec2.internal   Ready                      master   34m   v1.29.4
ip-10-0-153-11.ec2.internal    Ready                      worker   28m   v1.29.4
ip-10-0-153-150.ec2.internal   Ready                      master   34m   v1.29.4
```

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

6. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) をリスト表示して、カーネル引数が機能することを確認します。

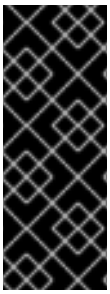
```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

**出力例**

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
BOOT_IMAGE=/ostree/rhcos-... console=tty0 console=ttyS0,115200n8
rootflags=defaults,prjquota rw root=UUID=fd0... ostree=/ostree/boot.0/rhcos/16...
coreos.oem.id=qemu coreos.oem.id=ec2 ignition.platform.id=ec2 enforcing=0

sh-4.2# exit
```

**enforcing=0** 引数が他のカーネル引数に追加されていることを確認できるはずです。

## 2.4. RHCOS のカーネル引数でのマルチパスの有効化



**重要**

インストール時のマルチパスの有効化が、OpenShift Container Platform 4.8 以降のバージョンでプロビジョニングされるノードでサポートおよび推奨されるようになりました。非最適化パスに対して I/O があると、I/O システムエラーが発生するように設定するには、インストール時にマルチパスを有効にする必要があります。インストール時にマルチパスを有効にする方法の詳細は、**ベアメタルへのインストール** ドキュメントの「インストール後のマルチパスの有効化」を参照してください。

Red Hat Enterprise Linux CoreOS (RHCOS) はプライマリーディスクでのマルチパスをサポートするようになり、ハードウェア障害に対する対障害性が強化され、ホストの可用性を強化できるようになりました。インストール後のサポートは、マシン設定を使用してマルチパスをアクティベートすることで利用できます。

**重要**

IBM Z® および IBM® LinuxONE では、インストール時にクラスターを設定した場合のみマルチパスを有効にできます。詳細は、**IBM Z® および IBM® LinuxONE への z/VM を使用したクラスターのインストール** の RHCOS の「インストールおよび OpenShift Container Platform ブートストラッププロセスの開始」を参照してください。

**重要**

OpenShift Container Platform 4.16 クラスターが、マルチパスが設定された IBM Power® の "vSCSI" ストレージを持つ単一の VIOS ホストでインストール後のアクティビティーとしてインストールまたは設定されている場合、マルチパスが有効になっている CoreOS ノードは起動に失敗します。ノードに使用できるパスは 1 つだけなので、これは想定内の動作です。

**前提条件**

- バージョン 4.7 以降を使用する OpenShift Container Platform クラスターが実行中である。
- 管理者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。

- ディスクでマルチパスが有効になっていることを確認しました。マルチパスは、HBA アダプターを介して SAN に接続されているホストでのみサポートされます。

**手順**

1. インストール後にコントロールプレーンノードでマルチパスを有効にするには、以下を実行します。

   - 以下の例のように、**master** ラベルを追加し、マルチパスカーネル引数を特定するようクラスターに指示する **99-master-kargs-mpath.yaml** などのマシン設定ファイルを作成します。

   ```
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     labels:
       machineconfiguration.openshift.io/role: "master"
     name: 99-master-kargs-mpath
   spec:
     kernelArguments:
       - 'rd.multipath=default'
       - 'root=/dev/disk/by-label/dm-mpath-root'
   ```

2. インストール後にワーカーノードでマルチパスを有効にするには、以下を実行します。

   - **worker** ラベルを追加し、マルチパスカーネル引数などを特定するようクラスターに指示する **99-worker-kargs-mpath.yaml** などのマシン設定ファイルを作成します。

   ```
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     labels:
       machineconfiguration.openshift.io/role: "worker"
     name: 99-worker-kargs-mpath
   spec:
     kernelArguments:
       - 'rd.multipath=default'
       - 'root=/dev/disk/by-label/dm-mpath-root'
   ```

3. 以前に作成したマスターまたはワーカー YAML ファイルのいずれかを使用して新規のマシン設定を作成します。

   ```
   $ oc create -f ./99-worker-kargs-mpath.yaml
   ```

4. マシン設定で新規の追加内容を確認します。

   ```
   $ oc get MachineConfig
   ```

   **出力例**

   ```
   NAME                                               GENERATEDBYCONTROLLER
   IGNITIONVERSION   AGE
   00-master                                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0
   33m
   ```

```
00-worker                                  52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0
33m
01-master-container-runtime                52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-master-kubelet                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-worker-container-runtime                52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
01-worker-kubelet                          52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-master-generated-registries             52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-master-ssh                                                       3.2.0             40m
99-worker-generated-registries             52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             33m
99-worker-kargs-mpath                      52dd3ba6a9a527fc3ab42afac8d12b693534c8c9
3.2.0             105s
99-worker-ssh                                                       3.2.0             40m
rendered-master-23e785de7587df95a4b517e0647e5ab7
52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0             33m
rendered-worker-5d596d9293ca3ea80c896a1191735bb1
52dd3ba6a9a527fc3ab42afac8d12b693534c8c9   3.2.0             33m
```

5. ノードを確認します。

```
$ oc get nodes
```

**出力例**

```
NAME                         STATUS                     ROLES    AGE   VERSION
ip-10-0-136-161.ec2.internal   Ready                      worker   28m   v1.29.4
ip-10-0-136-243.ec2.internal   Ready                      master   34m   v1.29.4
ip-10-0-141-105.ec2.internal   Ready,SchedulingDisabled   worker   28m   v1.29.4
ip-10-0-142-249.ec2.internal   Ready                      master   34m   v1.29.4
ip-10-0-153-11.ec2.internal    Ready                      worker   28m   v1.29.4
ip-10-0-153-150.ec2.internal   Ready                      master   34m   v1.29.4
```

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

6. ワーカーノードのいずれかに移動し、カーネルコマンドライン引数 (ホストの **/proc/cmdline** 内) をリスト表示して、カーネル引数が機能することを確認します。

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

**出力例**

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
```

```
...

sh-4.2# exit
```

追加したカーネル引数が表示されるはずです。

**関連情報**

- インストール時 のマルチパスの有効化の詳細は、RHCOS のカーネル引数で のマルチパスの有効化 を参照してください。

## 2.5. リアルタイムカーネルのノードへの追加

一部の OpenShift Container Platform ワークロードには、高度な決定論的アプローチが必要になります。Linux はリアルタイムのオペレーティングシステムではありませんが、Linux のリアルタイムカーネルには、リアルタイムの特性を持つオペレーティングシステムを提供するプリエンプティブなスケジューラーが含まれます。

OpenShift Container Platform ワークロードでこれらのリアルタイムの特性が必要な場合、マシンを Linux のリアルタイムカーネルに切り替えることができます。OpenShift Container Platform 4.16 の場合、**MachineConfig** オブジェクトを使用してこの切り替えを行うことができます。変更はマシン設定の **kernelType** 設定を **realtime** に変更するだけで簡単に行えますが、この変更を行う前に他のいくつかの点を考慮する必要があります。

- 現在、リアルタイムカーネルはワーカーノードでのみサポートされ、使用できるのはラジオアクセスネットワーク (RAN) のみになります。
- 以下の手順は、Red Hat Enterprise Linux for Real Time 8 で認定されているシステムを使用したベアメタルのインストールで完全にサポートされます。
- OpenShift Container Platform でのリアルタイムサポートは、特定のサブスクリプションに制限されます。
- 以下の手順は、Google Cloud Platform での使用についてもサポートされます。

**前提条件**

- OpenShift Container Platform クラスター (バージョン 4.4 以降) が実行中である。
- 管理者権限を持つユーザーとしてクラスターにログインしている。

**手順**

1. リアルタイムカーネルのマシン設定を作成します。**realtime** カーネルタイプの **MachineConfig** オブジェクトが含まれる YAML ファイル (**99-worker-realtime.yaml** など) を作成します。以下の例では、すべてのワーカーノードにリアルタイムカーネルを使用するようにクラスターに指示します。

```
$ cat << EOF > 99-worker-realtime.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: "worker"
  name: 99-worker-realtime
```

```
spec:
  kernelType: realtime
EOF
```

2. マシン設定をクラスターに追加します。以下を入力してマシン設定をクラスターに追加します。

```
$ oc create -f 99-worker-realtime.yaml
```

3. リアルタイムカーネルを確認します。影響を受けるそれぞれのノードの再起動後に、クラスターにログインして以下のコマンドを実行し、リアルタイムカーネルが設定されたノードのセットの通常のカーネルを置き換えていることを確認します。

```
$ oc get nodes
```

**出力例**

```
NAME                                         STATUS  ROLES    AGE   VERSION
ip-10-0-143-147.us-east-2.compute.internal  Ready   worker   103m  v1.29.4
ip-10-0-146-92.us-east-2.compute.internal   Ready   worker   101m  v1.29.4
ip-10-0-169-2.us-east-2.compute.internal    Ready   worker   102m  v1.29.4
```

```
$ oc debug node/ip-10-0-143-147.us-east-2.compute.internal
```

**出力例**

```
Starting pod/ip-10-0-143-147us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`

sh-4.4# uname -a
Linux <worker_node> 4.18.0-147.3.1.rt24.96.el8_1.x86_64 #1 SMP PREEMPT RT
        Wed Nov 27 18:29:55 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

カーネル名には **rt** が含まれ、"PREEMPT RT" のテキストは、これがリアルタイムカーネルであることを示します。

4. 通常のカーネルに戻るには、**MachineConfig** オブジェクトを削除します。

```
$ oc delete -f 99-worker-realtime.yaml
```

## 2.6. JOURNALD の設定

OpenShift Container Platform ノードで **journald** サービスの設定が必要な場合は、適切な設定ファイルを変更し、そのファイルをマシン設定としてノードの適切なプールに渡すことで実行できます。

この手順では、**/etc/systemd/journald.conf** ファイルの **journald** 速度制限の設定を変更し、それらをワーカーノードに適用する方法について説明します。このファイルの使用方法についての情報は、**journald.conf** man ページを参照してください。

**前提条件**

- OpenShift Container Platform クラスターが実行中である。

- 管理者権限を持つユーザーとしてクラスターにログインしている。

**手順**

1. 必要な設定で **/etc/systemd/journald.conf** ファイルが含まれる Butane 設定ファイル **40-worker-custom -journald.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。

```
variant: openshift
version: 4.16.0
metadata:
  name: 40-worker-custom-journald
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  files:
  - path: /etc/systemd/journald.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        # Disable rate limiting
        RateLimitInterval=1s
        RateLimitBurst=10000
        Storage=volatile
        Compress=no
        MaxRetentionSec=30s
```

2. Butane を使用して、ワーカー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 (**40-worker-custom-journald.yaml**) を生成します。

```
$ butane 40-worker-custom-journald.bu -o 40-worker-custom-journald.yaml
```

3. マシン設定をプールに適用します。

```
$ oc apply -f 40-worker-custom-journald.yaml
```

4. 新規マシン設定が適用され、ノードの状態が低下した状態にないことを確認します。これには数分の時間がかかる場合があります。各ノードで新規マシン設定が正常に適用されるため、ワーカープールには更新が進行中であることが表示されます。

```
$ oc get machineconfigpool
NAME   CONFIG              UPDATED UPDATING DEGRADED MACHINECOUNT
READYMACHINECOUNT UPDATEDMACHINECOUNT DEGRADEDMACHINECOUNT
AGE
master rendered-master-35 True    False    False    3          3              3              0
34m
worker rendered-worker-d8 False   True     False    3          1              1              0
34m
```

5. 変更が適用されたことを確認するには、ワーカーノードにログインします。

```
$ oc get node | grep worker
ip-10-0-0-1.us-east-2.compute.internal   Ready    worker   39m   v0.0.0-master+$Format:%h$
$ oc debug node/ip-10-0-0-1.us-east-2.compute.internal
Starting pod/ip-10-0-141-142us-east-2computeinternal-debug ...
...
sh-4.2# chroot /host
sh-4.4# cat /etc/systemd/journald.conf
# Disable rate limiting
RateLimitInterval=1s
RateLimitBurst=10000
Storage=volatile
Compress=no
MaxRetentionSec=30s
sh-4.4# exit
```

**関連情報**

- Butane でのマシン設定の作成

## 2.7. 拡張機能の RHCOS への追加

RHCOS はコンテナー指向の最小限の RHEL オペレーティングシステムであり、すべてのプラットフォームで OpenShift Container Platform クラスターに共通の機能セットを提供するように設計されています。ソフトウェアパッケージを RHCOS システムに追加することは一般的に推奨されていませんが、MCO は RHCOS ノードに最小限の機能セットを追加するために使用できる **extensions** 機能を提供します。

現時点で、以下の拡張機能が利用可能です。

- **usbguard**: **usbguard** 拡張機能を追加すると、RHCOS システムを割り込みの USB デバイスから保護します。詳細は、USBGuard を参照してください。
- **kerberos**: **kerberos** 拡張機能を追加すると、ユーザーとマシンの両方がネットワークに対して自分自身を識別し、管理者が設定したエリアとサービスへの定義済みの制限付きアクセスを取得できるメカニズムが提供されます。Kerberos クライアントのセットアップ方法や Kerberos 化された NFS 共有のマウント方法などの詳細は、Kerberos の使用 を参照してください。

以下の手順では、マシン設定を使用して 1 つ以上の拡張機能を RHCOS ノードに追加する方法を説明します。

**前提条件**

- OpenShift Container Platform クラスター (バージョン 4.6 以降) が実行中である。
- 管理者権限を持つユーザーとしてクラスターにログインしている。

**手順**

1. 拡張機能のマシン設定を作成します。**MachineConfig extensions** オブジェクトが含まれる YAML ファイル (例: **80-extensions.yaml**) を作成します。この例では、クラスターに対して **usbguard** 拡張機能を追加するように指示します。

```
$ cat << EOF > 80-extensions.yaml
```

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 80-worker-extensions
spec:
  config:
    ignition:
      version: 3.2.0
  extensions:
    - usbguard
EOF
```

2. マシン設定をクラスターに追加します。以下を入力してマシン設定をクラスターに追加します。

```
$ oc create -f 80-extensions.yaml
```

これにより、すべてのワーカーノードで **usbguard** の rpm パッケージがインストールされるように設定できます。

3. 拡張機能が適用されていることを確認します。

```
$ oc get machineconfig 80-worker-extensions
```

**出力例**

```
NAME                 GENERATEDBYCONTROLLER IGNITIONVERSION AGE
80-worker-extensions                       3.2.0           57s
```

4. 新規マシン設定が適用され、ノードの状態が低下した状態にないことを確認します。これには数分の時間がかかる場合があります。各マシンで新規マシン設定が正常に適用されるため、ワーカープールには更新が進行中であることが表示されます。

```
$ oc get machineconfigpool
```

**出力例**

```
NAME   CONFIG              UPDATED UPDATING DEGRADED MACHINECOUNT
READYMACHINECOUNT UPDATEDMACHINECOUNT DEGRADEDMACHINECOUNT
AGE
master rendered-master-35 True    False    False    3          3              3                0
34m
worker rendered-worker-d8 False   True     False    3          1              1                0
34m
```

5. 拡張機能を確認します。拡張機能が適用されたことを確認するには、以下を実行します。

```
$ oc get node | grep worker
```

**出力例**

```
NAME                                         STATUS  ROLES    AGE   VERSION
ip-10-0-169-2.us-east-2.compute.internal     Ready   worker   102m  v1.29.4
```

```
$ oc debug node/ip-10-0-169-2.us-east-2.compute.internal
```

**出力例**

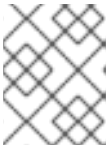
```
...
To use host binaries, run `chroot /host`
sh-4.4# chroot /host
sh-4.4# rpm -q usbguard
usbguard-0.7.4-4.el8.x86_64.rpm
```

## 2.8. マシン設定マニフェストでのカスタムファームウェアブロブの読み込み

**/usr/lib** 内のファームウェアブロブのデフォルトの場所は読み取り専用であるため、検索パスを更新して、カスタムファームウェアブロブを特定できます。これにより、ブロブが RHCOS によって管理されない場合に、マシン設定マニフェストでローカルファームウェアブロブを読み込むことができます。

**手順**

1. Butane 設定ファイル **98-worker-firmware-blob.bu** を作成します。このファイルは、root 所有でローカルストレージに書き込みできるように、検索パスを更新します。以下の例では、カスタムブロブファイルをローカルワークステーションからノードの **/var/lib/firmware** 下に配置しています。



**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。

**カスタムファームウェアブロブ用の Butane 設定ファイル**

```
variant: openshift
version: 4.16.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-worker-firmware-blob
storage:
  files:
  - path: /var/lib/firmware/<package_name> 1
    contents:
      local: <package_name> 2
    mode: 0644 3
openshift:
  kernel_arguments:
    - 'firmware_class.path=/var/lib/firmware' 4
```

1 ファームウェアパッケージのコピー先となるノードのパスを設定します。

2 Butane を実行しているシステムのローカルファイルディレクトリーから読み取るコンテンツを含むファイルを指定します。ローカルファイルのパスは **files-dir** ディレクトリーからの相対パスで、以下の手順の Butane で **--files-dir** オプションを使用して指定する必要があります。

3 RHCOS ノードのファイルのパーミッションを設定します。**0644** パーミッションを設定することが推奨されます。

4 **firmware_class.path** パラメーターは、ローカルワークステーションからノードのルートファイルシステムにコピーされたカスタムファームウェアブロブを検索するカーネルの検索パスをカスタマイズします。この例では、**/var/lib/firmware** をカスタマイズされたパスとして使用します。

2. Butane を実行して、ローカルワークステーション上の**98-worker-firmware-blob.yaml** という名前のファームウェアブロブのコピーを使用する **MachineConfig** オブジェクトファイルを生成します。ファームウェアブロブには、ノードに配信される設定が含まれます。次の例では、**--files-dir** オプションを使用して、ローカルファイルが配置されるワークステーション上のディレクトリーを指定します。

```
$ butane 98-worker-firmware-blob.bu -o 98-worker-firmware-blob.yaml --files-dir
<directory_including_package_name>
```

3. 以下の 2 つの方法のいずれかで、設定をノードに適用します。

   - 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 98-worker-firmware-blob.yaml
     ```

     **MachineConfig** オブジェクト YAML ファイルは、マシンの設定を終了するために作成されます。

4. 将来的に **MachineConfig** オブジェクトを更新する必要がある場合に備えて、Butane 設定を保存します。

**関連情報**

- Butane でのマシン設定の作成

## 2.9. ノードアクセス用のコアユーザーパスワードの変更

デフォルトでは、Red Hat Enterprise Linux CoreOS (RHCOS) はクラスター内のノードに **core** という名前のユーザーを作成します。**core** ユーザーを使用して、クラウドプロバイダーのシリアルコンソールまたはベアメタルベースボードコントローラーマネージャー (BMC) を介してノードにアクセスできます。これは、たとえば、ノードがダウンしていて、SSH または **oc debug node** コマンドを使用して、そのノードにアクセスできない場合に役立ちます。ただし、デフォルトでは、このユーザーにはパスワードがないため、パスワードを作成しないとログインできません。

マシン設定を使用して、**core** ユーザーのパスワードを作成できます。Machine Config Operator (MCO) がパスワードを割り当て、そのパスワードを **/etc/shadow** ファイルに挿入して、**core** ユーザーでログインできるようにします。MCO はパスワードハッシュを調べません。そのため、パスワードに問題が

ある場合、MCO は報告できません。

**注記**

- パスワードは、クラウドプロバイダーのシリアルコンソールまたは BMC を介してのみ機能します。SSH では動作しません。
- **/etc/shadow** ファイルまたはパスワードを設定する systemd ユニットを含むマシン設定がある場合、パスワードハッシュよりも優先されます。

必要に応じて、パスワードの作成に使用したマシン設定を編集して、パスワードを変更できます。また、マシン設定を削除することでパスワードを削除できます。マシン設定を削除しても、ユーザーアカウントは削除されません。

**手順**

1. オペレーティングシステムでサポートされているツールを使用して、ハッシュ化されたパスワードを作成します。たとえば次のコマンドを実行し、**mkpasswd** を使用してハッシュ化されたパスワードを作成します。

```
$ mkpasswd -m SHA-512 testpass
```

**出力例**

```
$
$6$CBZwA6s6AVFOtiZe$aUKDWpthhJEyR3nnhM02NM1sKCpHn9XN.NPrJNQ3HYewioaorpwL3mKGLxvW0AOb4pJxqoqP4nFX77y0p00.8.
```

2. **core** ユーザー名とハッシュ化されたパスワードを含むマシン設定ファイルを作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: set-core-user-password
spec:
  config:
    ignition:
      version: 3.2.0
    passwd:
      users:
      - name: core (1)
        passwordHash: <password> (2)
```

(1) これは **core** である必要があります。

(2) **core** アカウントで使用するハッシュ化されたパスワード。

3. 次のコマンドを実行して、マシン設定を作成します。

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

ノードは再起動せず、しばらくすると使用可能になります。次の例に示すように、**oc get mcp** を使用して、マシン設定プールが更新されるのを監視できます。

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
master   rendered-master-d686a3ffc8fdec47280afec446fce8dd   True      False      False      3
3              3                   0                   64m
worker   rendered-worker-4605605a5b1f9de1d061e9d350f251e5   False     True       False
3              0                   0                   0                   64m
```

**検証**

1. ノードが **UPDATED=True** 状態に戻ったら、次のコマンドを実行してノードのデバッグセッションを開始します。

```
$ oc debug node/<node_name>
```

2. 次のコマンドを実行して、デバッグシェル内のルートディレクトリーとして **/host** を設定します。

```
sh-4.4# chroot /host
```

3. **/etc/shadow** ファイルの内容を確認します。

**出力例**

```
...
core:$6$2sE/010goDuRSxxv$o18K52wor.wlwZp:19418:0:99999:7:::
...
```

ハッシュ化されたパスワードは、**core** ユーザーに割り当てられます。

# 第3章 ノード停止ポリシーを使用してマシン設定の変更による停止を最小限に抑える

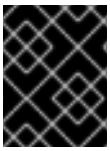
デフォルトでは、**MachineConfig** オブジェクトのフィールドに何らかの変更を加えると、Machine Config Operator (MCO) がそのマシン設定に関連付けられているノードをドレインして再起動します。ただし、**ノード停止ポリシー** を作成すると、一部の Ignition 設定オブジェクトに対して、ワークロードの停止をほとんどまたはまったく引き起こさない一連の変更を定義できます。

ノード停止ポリシーを使用すると、クラスターに停止を引き起こす設定変更と、引き起こさない設定変更を定義できます。これにより、クラスター内で小さなマシン設定の変更を行うときに、ノードのダウンタイムを短縮できます。ポリシーを設定するには、**openshift-machine-config-Operator** namespace にある **MachineConfiguration** オブジェクトを変更します。後述する **MachineConfiguration** オブジェクトのノード停止ポリシーの例を参照してください。

**注記**

ノード停止ポリシーに関係なく、常に再起動が必要となるマシン設定の変更があります。詳細は、**Machine Config Operator について** を参照してください。

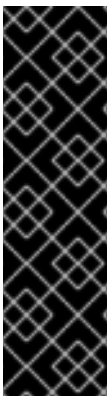
ノード停止ポリシーを作成すると、MCO がポリシーを検証し、フォーマットの問題など、ファイル内の潜在的な問題を検索します。次に、MCO はポリシーをクラスターのデフォルト設定とマージし、マシン設定の **status.nodeDisruptionPolicyStatus** フィールドに、マシン設定が将来変更されたときに実行されるアクションを入力します。クラスターのデフォルト設定は、常にポリシー内の設定で上書きされます。



**重要**

MCO は、ノード停止ポリシーによって変更が正常に適用できるかどうかを検証しません。したがって、ノード停止ポリシーの正確性を確認する責任はお客様にあります。

たとえば、sudo 設定によるノードのドレインと再起動が必要ないように、ノード停止ポリシーを設定できます。また、**sshd** への更新を適用する際に sshd サービスだけをリロードするようにクラスターを設定することもできます。



**重要**

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

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

次の Ignition 設定オブジェクトに変更を加えるときに、MCO の動作を制御できます。

- **設定ファイル**: **/var** または **/etc** ディレクトリー内のファイルを追加または更新します。
- **systemd ユニット**: systemd サービスを作成してステータスを設定するか、既存の systemd サービスを変更します。

- **ユーザーとグループ**: インストール後に **passwd** セクションで SSH キーを変更します。
- ICSP、ITMS、IDMS オブジェクト: **ImageContentSourcePolicy** (ICSP)、**ImageTagMirrorSet** (ITMS)、および **ImageDigestMirrorSet** (IDMS) オブジェクトからミラーリングルールを削除できます。

これらの変更のいずれかを行うと、ノード停止ポリシーによって、MCO が変更を実装するときに必要なアクションが次の中から決定されます。

- **Reboot**: MCO はノードをドレインして再起動します。これがデフォルトの動作です。
- **None**: MCO はノードのドレインも再起動も実行しません。MCO は、それ以上のアクションなしで変更を適用します。
- **Drain**: MCO はノードのワークロードを遮断してドレインします。ワークロードは新しい設定で再起動します。
- **Reload**: サービスの場合、MCO はサービスを再起動せずに指定されたサービスをリロードします。
- **Restart**: サービスの場合、MCO は指定されたサービスを完全に再起動します。
- **DaemonReload**: MCO は systemd マネージャー設定をリロードします。
- **Special**: これは MCO 専用の内部アクションであり、ユーザーが設定することはできません。

**注記**

- **Reboot** および **None** アクションを他のアクションと一緒に使用することはできません。**Reboot** および **None** アクションは、他のアクションをオーバーライドするためです。
- アクションは、ノード停止ポリシーのリストに設定されている順序で適用されます。
- ノードの再起動やその他の停止が必要となるその他のマシン設定の変更を行った場合、その再起動はノード停止ポリシーのアクションよりも優先されます。

## 3.1. ノード停止ポリシーの例

次の例の **MachineConfiguration** オブジェクトには、ノード停止ポリシーが含まれています。

**ヒント**

**MachineConfiguration** オブジェクトと **MachineConfig** オブジェクトは別々のオブジェクトです。**MachineConfiguration** オブジェクトは、MCO Operator の設定パラメーターを含む MCO namespace 内のシングルトンオブジェクトです。**MachineConfig** オブジェクトは、マシン設定プールに適用される変更を定義します。

次の例の **MachineConfiguration** オブジェクトでは、ユーザー定義のポリシーは示していません。デフォルトのノード停止ポリシーの値を **status** スタンザに示します。

**デフォルトのノード停止ポリシー**

```
apiVersion: operator.openshift.io/v1
```

```
kind: MachineConfiguration
  name: cluster
spec:
  logLevel: Normal
  managementState: Managed
  operatorLogLevel: Normal
status:
  nodeDisruptionPolicyStatus:
    clusterPolicies:
      files:
      - actions:
        - type: None
        path: /etc/mco/internal-registry-pull-secret.json
      - actions:
        - type: None
        path: /var/lib/kubelet/config.json
      - actions:
        - reload:
            serviceName: crio.service
          type: Reload
        path: /etc/machine-config-daemon/no-reboot/containers-gpg.pub
      - actions:
        - reload:
            serviceName: crio.service
          type: Reload
        path: /etc/containers/policy.json
      - actions:
        - type: Special
        path: /etc/containers/registries.conf
      sshkey:
        actions:
        - type: None
  readyReplicas: 0
```

次の例では、SSH キーが変更されたときに、MCO はクラスターノードのドレイン、**crio.service** のリロード、systemd 設定のリロード、**crio-service** の再起動を実行します。

**SSH キーの変更に対するノード停止ポリシーの例**

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
# ...
spec:
  nodeDisruptionPolicy:
    sshkey:
      actions:
      - type: Drain
      - reload:
          serviceName: crio.service
        type: Reload
      - type: DaemonReload
      - restart:
```

```
        serviceName: crio.service
      type: Restart
# ...
```

次の例では、**/etc/chrony.conf** ディレクトリー内のファイルが変更されたときに、MCO はクラスターノード上の **chronyd.service** をリロードします。

**設定ファイルの変更に対するノード停止ポリシーの例**

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
# ...
spec:
  nodeDisruptionPolicy:
    files:
    - actions:
      - reload:
          serviceName: chronyd.service
        type: Reload
      path: /etc/chrony.conf
```

次の例では、**auditd.service** systemd ユニットが変更されたときに、MCO はクラスターノードのドレイン、**crio.service** のリロード、systemd マネージャー設定のリロード、**crio.service** の再起動を実行します。

**設定ファイルの変更に対するノード停止ポリシーの例**

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
# ...
spec:
  nodeDisruptionPolicy:
    units:
      - name: auditd.service
        actions:
          - type: Drain
          - type: Reload
            reload:
              serviceName: crio.service
          - type: DaemonReload
          - type: Restart
            restart:
              serviceName: crio.service
```

次の例では、**registries.conf** ディレクトリー内のファイルが変更されたときに、MCO はノードをドレインまたは再起動せず、それ以上のアクションなしで変更を適用します。

**設定ファイルの変更に対するノード停止ポリシーの例**

-

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
# ...
spec:
  nodeDisruptionPolicy:
      - actions:
        - type: None
        path: /etc/containers/registries.conf
```

## 3.2. マシン設定の変更時にノードを再起動する動作を設定する

ノード停止ポリシーを作成すると、クラスターの停止を引き起こすマシン設定の変更と、引き起こさない変更を定義できます。

**/var** または **/etc** ディレクトリー内のファイル、systemd ユニット、SSH キー、および **registries.conf** ファイルの変更に対してノードがどのように反応するかを制御できます。

これらの変更のいずれかを行うと、ノード停止ポリシーによって、MCO が変更を実装するときに必要なアクションが次の中から決定されます。

- **Reboot**: MCO はノードをドレインして再起動します。これがデフォルトの動作です。
- **None**: MCO はノードのドレインも再起動も実行しません。MCO は、それ以上のアクションなしで変更を適用します。
- **Drain**: MCO はノードのワークロードを遮断してドレインします。ワークロードは新しい設定で再起動します。
- **Reload**: サービスの場合、MCO はサービスを再起動せずに指定されたサービスをリロードします。
- **Restart**: サービスの場合、MCO は指定されたサービスを完全に再起動します。
- **DaemonReload**: MCO は systemd マネージャー設定をリロードします。
- **Special**: これは MCO 専用の内部アクションであり、ユーザーが設定することはできません。

**注記**

- **Reboot** および **None** アクションを他のアクションと一緒に使用することはできません。**Reboot** および **None** アクションは、他のアクションをオーバーライドするためです。
- アクションは、ノード停止ポリシーのリストに設定されている順序で適用されます。
- ノードの再起動やその他の停止が必要となるその他のマシン設定の変更を行った場合、その再起動はノード停止ポリシーのアクションよりも優先されます。

**前提条件**

- フィーチャーゲートを使用して **TechPreviewNoUpgrade** 機能セットを有効にしている。詳細は、「フィーチャーゲートを使用した機能の有効化」を参照してください。

**警告**

クラスターで **TechPreviewNoUpgrade** 機能セットを有効にすると、マイナーバージョンの更新ができなくなります。**TechPreviewNoUpgrade** 機能セットは無効にできません。実稼働クラスターではこの機能セットを有効にしないでください。

**手順**

1. **machineconfigurations.operator.openshift.io** オブジェクトを編集して、ノード停止ポリシーを定義します。

```
$ oc edit MachineConfiguration cluster -n openshift-machine-config-operator
```

2. 次のようなノード停止ポリシーを追加します。

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
# ...
spec:
  nodeDisruptionPolicy: (1)
    files: (2)
    - actions: (3)
      - reload: (4)
          serviceName: chronyd.service (5)
        type: Reload
      path: /etc/chrony.conf (6)
    sshkey: (7)
      actions:
      - type: Drain
      - reload:
          serviceName: crio.service
        type: Reload
      - type: DaemonReload
      - restart:
          serviceName: crio.service
        type: Restart
    units: (8)
    - actions:
      - type: Drain
      - reload:
          serviceName: crio.service
        type: Reload
      - type: DaemonReload
```

```
    - restart:
        serviceName: crio.service
      type: Restart
    name: test.service
```

1. ノード停止ポリシーを指定します。
2. マシン設定ファイルの定義と、それらのパスの変更に対して実行するアクションのリストを指定します。このリストは最大 50 個のエントリーをサポートします。
3. 指定したファイルが変更されたときに実行する一連のアクションを指定します。アクションは、このリストに設定されている順序で適用されます。このリストは最大 10 個のエントリーをサポートします。
4. 指定したファイルが変更されたときに、リストしたサービスをリロードすることを指定します。
5. アクションの対象となるサービスの完全な名前を指定します。
6. マシン設定によって管理されるファイルの場所を指定します。ポリシー内のアクションは、**path** 内のファイルが変更されたときに適用されます。
7. クラスター内の SSH キーが変更されたときに実行するサービス名とアクションのリストを指定します。
8. systemd ユニット名のリストと、これらのユニットが変更されたときに実行するアクションを指定します。

**検証**

- 作成した **MachineConfiguration** オブジェクトファイルを表示します。

```
$ oc get MachineConfiguration/cluster -o yaml
```

**出力例**

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: cluster
# ...
status:
  nodeDisruptionPolicyStatus: 1
    clusterPolicies:
      files:
# ...
      - actions:
        - reload:
            serviceName: chronyd.service
          type: Reload
        path: /etc/chrony.conf
      sshkey:
```

```
      actions:
      - type: Drain
      - reload:
          serviceName: crio.service
        type: Reload
      - type: DaemonReload
      - restart:
          serviceName: crio.service
        type: Restart
    units:
    - actions:
      - type: Drain
      - reload:
          serviceName: crio.service
        type: Reload
      - type: DaemonReload
      - restart:
          serviceName: crio.service
        type: Restart
      name: test.se
# ...
```

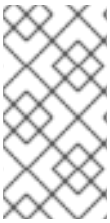
1 現在のクラスター検証済みのポリシーを示します。

# 第4章 MCO 関連のカスタムリソースの設定

MCO は **MachineConfig** オブジェクトを管理する以外にも、2 つのカスタムリソース (CR) (**KubeletConfig** および **ContainerRuntimeConfig**) を管理します。これらの CR を使用すると、kubelet および CRI-O コンテナーランタイムサービスの動作に影響を与えるノードレベルの設定を変更できます。

## 4.1. KUBELET パラメーターを編集するための KUBELETCONFIG CRD の作成

kubelet 設定は、現時点で Ignition 設定としてシリアル化されているため、直接編集することができます。ただし、新規の **kubelet-config-controller** も Machine Config Controller (MCC) に追加されます。これにより、**KubeletConfig** カスタムリソース (CR) を使用して kubelet パラメーターを編集できます。

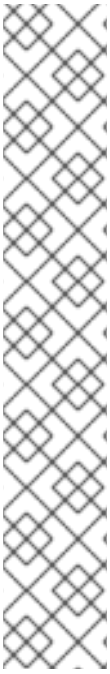


**注記**

**kubeletConfig** オブジェクトのフィールドはアップストリーム Kubernetes から kubelet に直接渡されるため、kubelet はそれらの値を直接検証します。**kubeletConfig** オブジェクトに無効な値により、クラスターノードが利用できなくなります。有効な値は、Kubernetes ドキュメント を参照してください。

以下のガイダンスを参照してください。

- 既存の **KubeletConfig** CR を編集して既存の設定を編集するか、変更ごとに新規 CR を作成する代わりに新規の設定を追加する必要があります。CR を作成するのは、別のマシン設定プールを変更する場合、または一時的な変更を目的とした変更の場合のみにして、変更を元に戻すことができるようにすることを推奨します。
- マシン設定プールごとに、そのプールに加える設定変更をすべて含めて、**KubeletConfig** CR を 1 つ作成します。
- 必要に応じて、クラスターごとに 10 を制限し、複数の **KubeletConfig** CR を作成します。最初の **KubeletConfig** CR について、Machine Config Operator (MCO) は **kubelet** で追加されたマシン設定を作成します。それぞれの後続の CR で、コントローラーは数字の接尾辞が付いた別の **kubelet** マシン設定を作成します。たとえば、**kubelet** マシン設定があり、その接尾辞が **-2** の場合に、次の **kubelet** マシン設定には **-3** が付けられます。



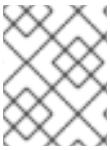
**注記**

kubelet またはコンテナーのランタイム設定をカスタムマシン設定プールに適用する場合、**machineConfigSelector** のカスタムロールは、カスタムマシン設定プールの名前と一致する必要があります。

たとえば、次のカスタムマシン設定プールの名前は **infra** であるため、カスタムロールも **infra** にする必要があります。

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

マシン設定を削除する場合は、制限を超えないようにそれらを逆の順序で削除する必要があります。たとえば、**kubelet-3** マシン設定を、**kubelet-2** マシン設定を削除する前に削除する必要があります。



**注記**

接尾辞が **kubelet-9** のマシン設定があり、別の **KubeletConfig** CR を作成する場合には、**kubelet** マシン設定が 10 未満の場合でも新規マシン設定は作成されません。

**KubeletConfig CR の例**

```
$ oc get kubeletconfig
```

```
NAME                AGE
set-max-pods        15m
```

**KubeletConfig マシン設定を示す例**

```
$ oc get mc | grep kubelet
```

```
...
99-worker-generated-kubelet-1                  b5c5119de007945b6fe6fb215db3b8e2ceb12511    3.2.0
26m
...
```

以下の手順は、ワーカーノードでノードあたりの Pod の最大数を設定する方法を示しています。

**前提条件**

1. 設定するノードタイプの静的な **MachineConfigPool** CR に関連付けられたラベルを取得します。以下のいずれかの手順を実行します。

    a. マシン設定プールを表示します。

```
$ oc describe machineconfigpool <name>
```

以下に例を示します。

```
$ oc describe machineconfigpool worker
```

**出力例**

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  creationTimestamp: 2019-02-08T14:52:39Z
  generation: 1
  labels:
    custom-kubelet: set-max-pods 1
```

1 ラベルが追加されると、**labels** の下に表示されます。

b. ラベルが存在しない場合は、キー/値のペアを追加します。

```
$ oc label machineconfigpool worker custom-kubelet=set-max-pods
```

**手順**

1. 選択可能なマシン設定オブジェクトを表示します。

```
$ oc get machineconfig
```

デフォルトで、2 つの kubelet 関連の設定である **01-master-kubelet** および **01-worker-kubelet** を選択できます。

2. ノードあたりの最大 Pod の現在の値を確認します。

```
$ oc describe node <node_name>
```

以下に例を示します。

```
$ oc describe node ci-ln-5grqprb-f76d1-ncnqq-worker-a-mdv94
```

**Allocatable** スタンザで **value: pods: <value>** を検索します。

**出力例**

```
Allocatable:
 attachable-volumes-aws-ebs:  25
 cpu:                         3500m
 hugepages-1Gi:               0
 hugepages-2Mi:               0
 memory:                      15341844Ki
 pods:                        250
```

3. ワーカーノードでノードあたりの最大の Pod を設定するには、kubelet 設定を含むカスタムリソースファイルを作成します。

**重要**

特定のマシン設定プールをターゲットとする kubelet 設定は、依存するプールにも影響します。たとえば、ワーカーノードを含むプール用の kubelet 設定を作成すると、インフラストラクチャーノードを含むプールを含むすべてのサブセットプールにも設定が適用されます。これを回避するには、ワーカーノードのみを含む選択式を使用して新しいマシン設定プールを作成し、kubelet 設定でこの新しいプールをターゲットにする必要があります。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: set-max-pods
spec:
  machineConfigPoolSelector:
    matchLabels:
      custom-kubelet: set-max-pods 1
  kubeletConfig:
    maxPods: 500 2
```

1 Machine Config Pool からラベルを入力します。

2 kubelet 設定を追加します。この例では、**maxPods** を使用してノードあたりの最大 Pod を設定します。

**注記**

kubelet が API サーバーと通信する速度は、1秒あたりのクエリー (QPS) およびバースト値により異なります。デフォルト値の **50** (**kubeAPIQPS** の場合) および **100** (**kubeAPIBurst** の場合) は、各ノードで制限された Pod が実行されている場合には十分な値です。ノード上に CPU およびメモリーリソースが十分にある場合には、kubelet QPS およびバーストレートを更新することが推奨されます。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: set-max-pods
spec:
  machineConfigPoolSelector:
    matchLabels:
      custom-kubelet: set-max-pods
  kubeletConfig:
    maxPods: <pod_count>
    kubeAPIBurst: <burst_rate>
    kubeAPIQPS: <QPS>
```

a. ラベルを使用してワーカーのマシン設定プールを更新します。

```
$ oc label machineconfigpool worker custom-kubelet=set-max-pods
```

b. **KubeletConfig** オブジェクトを作成します。

```
$ oc create -f change-maxPods-cr.yaml
```

c. **KubeletConfig** オブジェクトが作成されていることを確認します。

```
$ oc get kubeletconfig
```

**出力例**

```
NAME              AGE
set-max-pods      15m
```

クラスター内のワーカーノードの数によっては、ワーカーノードが 1 つずつ再起動されるのを待機します。3 つのワーカーノードを持つクラスターの場合は、10 分から 15 分程度かかる可能性があります。

4. 変更がノードに適用されていることを確認します。

a. **maxPods** 値が変更されたワーカーノードで確認します。

```
$ oc describe node <node_name>
```

b. **Allocatable** スタンザを見つけます。

```
 ...
Allocatable:
  attachable-volumes-gce-pd:  127
  cpu:                        3500m
  ephemeral-storage:          123201474766
  hugepages-1Gi:              0
  hugepages-2Mi:              0
  memory:                     14225400Ki
  pods:                       500 (1)
 ...
```

(1) この例では、**pods** パラメーターは **KubeletConfig** オブジェクトに設定した値を報告するはずです。

5. **KubeletConfig** オブジェクトの変更を確認します。

```
$ oc get kubeletconfigs set-max-pods -o yaml
```

これは、以下の例のように **True** および **type:Success** のステータスを表示します。

```
spec:
  kubeletConfig:
    maxPods: 500
  machineConfigPoolSelector:
    matchLabels:
      custom-kubelet: set-max-pods
status:
  conditions:
```

```
- lastTransitionTime: "2021-06-30T17:04:07Z"
  message: Success
  status: "True"
  type: Success
```

## 4.2. CRI-O パラメーターを編集するための CONTAINERRUNTIMECONFIG CR の作成

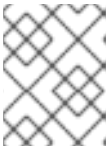
特定のマシン設定プール (MCP) に関連付けられたノードの OpenShift Container Platform CRI-O ランタイムに関連付けられる設定の一部を変更することができます。**ContainerRuntimeConfig** カスタムリソース (CR) を使用して、設定値を設定し、MCP に一致するラベルを追加します。次に、MCO は関連付けられたノードで **crio.conf** および **storage.conf** 設定ファイルを更新された値を使用して再ビルドします。

**注記**

**ContainerRuntimeConfig** CR を使用して実装された変更を元に戻すには、CR を削除する必要があります。マシン設定プールからラベルを削除しても、変更は元に戻されません。

**ContainerRuntimeConfig** CR を使用して以下の設定を変更することができます。

- **PID 制限**: **ContainerRuntimeConfig** での PID 制限の設定は非推奨になる予定です。PID 制限が必要な場合は、代わりに **KubeletConfig** CR の **podPidsLimit** フィールドを使用することを推奨します。**podPidsLimit** フィールドのデフォルト値は **4096** です。

  **注記**

  CRI-O フラグはコンテナーの cgroup に適用され、Kubelet フラグは Pod の cgroup に設定されます。それに応じて PID 制限を調整してください。

- Log level: **logLevel** パラメーターは CRI-O **log_level** パラメーターを設定します。これはログメッセージの詳細レベルです。デフォルトは **info** (**log_level = info**) です。他のオプションには、**fatal**、**panic**、**error**、**warn**、**debug**、および **trace** が含まれます。

- Overlay size: **overlaySize** パラメーターは、コンテナーイメージの最大サイズである CRI-O Overlay ストレージドライバーの **size** パラメーターを設定します。

- **最大ログサイズ**: **ContainerRuntimeConfig** での最大ログサイズの設定は非推奨になる予定です。最大ログサイズが必要な場合は、代わりに **KubeletConfig** CR の **containerLogMaxSize** フィールドを使用することを推奨します。

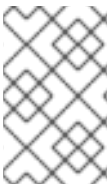
- **コンテナーランタイム**: **defaultRuntime** パラメーターは、コンテナーランタイムを **runc** または **crun** に設定します。デフォルトは **runc** です。

マシン設定プールごとに、そのプールに加える設定変更をすべて含めて、**ContainerRuntimeConfig** CR を 1 つ割り当てる必要があります。同じコンテンツをすべてのプールに適用している場合には、すべてのプールに必要となるのは **ContainerRuntimeConfig** CR 1 つだけです。

既存の **ContainerRuntimeConfig** CR を編集して既存の設定を編集するか、変更ごとに新規 CR を作成する代わりに新規の設定を追加する必要があります。異なるマシン設定プールを変更する場合や、変更が一時的で元に戻すことができる場合のみ、新しい **ContainerRuntimeConfig** CR の作成を推奨しています。

必要に応じて複数の **ContainerRuntimeConfig** CR を作成できます。この場合、制限はクラスターごとに 10 個となっています。最初の **ContainerRuntimeConfig** CR について、MCO は **containerruntime** で追加されたマシン設定を作成します。それぞれの後続の CR で、コントローラーは数字の接尾辞が付いた新規の **containerruntime** マシン設定を作成します。たとえば、**containerruntime** マシン設定に **-2** 接尾辞がある場合、次の **containerruntime** マシン設定が **-3** を付けて追加されます。

マシン設定を削除する場合、制限を超えないようにそれらを逆の順序で削除する必要があります。たとえば、**containerruntime-3** マシン設定を、**containerruntime-2** マシン設定を削除する前に削除する必要があります。



**注記**

接尾辞が **containerruntime-9** のマシン設定があり、別の **ContainerRuntimeConfig** CR を作成する場合には、**containerruntime** マシン設定が 10 未満の場合でも新規マシン設定は作成されません。

**複数の ContainerRuntimeConfig CR を示す例**

```
$ oc get ctrcfg
```

**出力例**

```
NAME         AGE
ctr-overlay  15m
ctr-level    5m45s
```

**複数の containerruntime マシン設定を示す例**

```
$ oc get mc | grep container
```

**出力例**

```
...
01-master-container-runtime                 b5c5119de007945b6fe6fb215db3b8e2ceb12511    3.2.0
57m
...
01-worker-container-runtime                 b5c5119de007945b6fe6fb215db3b8e2ceb12511    3.2.0
57m
...
99-worker-generated-containerruntime            b5c5119de007945b6fe6fb215db3b8e2ceb12511
3.2.0            26m
99-worker-generated-containerruntime-1          b5c5119de007945b6fe6fb215db3b8e2ceb12511
3.2.0            17m
99-worker-generated-containerruntime-2          b5c5119de007945b6fe6fb215db3b8e2ceb12511
3.2.0            7m26s
...
```

次の例では、**log_level** フィールドを **debug** に設定し、オーバーレイサイズを 8 GB に設定します。

**ContainerRuntimeConfig CR の例**

```
apiVersion: machineconfiguration.openshift.io/v1
```

```
kind: ContainerRuntimeConfig
metadata:
 name: overlay-size
spec:
 machineConfigPoolSelector:
   matchLabels:
     pools.operator.machineconfiguration.openshift.io/worker: " 1
 containerRuntimeConfig:
   logLevel: debug 2
   overlaySize: 8G 3
   defaultRuntime: "crun" 4
```

1. マシン設定プールのラベルを指定します。コンテナーのランタイム設定の場合、ロールは関連付けられているマシン設定プールの名前と一致する必要があります。
2. オプション: ログメッセージの詳細レベルを指定します。
3. オプション: コンテナーイメージの最大サイズを指定します。
4. オプション：新規コンテナーにデプロイするコンテナーランタイムを指定します。デフォルト値は **runc** です。

**手順**

**ContainerRuntimeConfig** CR を使用して CRI-O 設定を変更するには、以下を実行します。

1. **ContainerRuntimeConfig** CR の YAML ファイルを作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: ContainerRuntimeConfig
metadata:
 name: overlay-size
spec:
 machineConfigPoolSelector:
   matchLabels:
     pools.operator.machineconfiguration.openshift.io/worker: " 1
 containerRuntimeConfig: 2
   logLevel: debug
   overlaySize: 8G
```

   1. 変更する必要のあるマシン設定プールのラベルを指定します。
   2. 必要に応じてパラメーターを設定します。

2. **ContainerRuntimeConfig** CR を作成します。

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

3. CR が作成されたことを確認します。

```
$ oc get ContainerRuntimeConfig
```

**出力例**

```
NAME          AGE
overlay-size  3m19s
```

4. 新規の **containerruntime** マシン設定が作成されていることを確認します。

```
$ oc get machineconfigs | grep containerrun
```

**出力例**

```
99-worker-generated-containerruntime  2c9371fbb673b97a6fe8b1c52691999ed3a1bfc2
3.2.0  31s
```

5. すべてが準備状態にあるものとして表示されるまでマシン設定プールをモニターします。

```
$ oc get mcp worker
```

**出力例**

```
NAME    CONFIG              UPDATED  UPDATING  DEGRADED  MACHINECOUNT
READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT
AGE
worker  rendered-worker-169  False    True      False     3          1                1                 0
9h
```

6. 設定が CRI-O で適用されたことを確認します。

   a. マシン設定プールのノードに対して **oc debug** セッションを開き、**chroot /host** を実行します。

```
$ oc debug node/<node_name>
```

```
sh-4.4# chroot /host
```

   b. **crio.conf** ファイルの変更を確認します。

```
sh-4.4# crio config | grep 'log_level'
```

**出力例**

```
log_level = "debug"
```

   c. `storage.conf` ファイルの変更を確認します。

```
sh-4.4# head -n 7 /etc/containers/storage.conf
```

**出力例**

```
[storage]
  driver = "overlay"
  runroot = "/var/run/containers/storage"
  graphroot = "/var/lib/containers/storage"
```

```
[storage.options]
  additionalimagestores = []
  size = "8G"
```

## 4.3. CRI-O を使用した OVERLAY のデフォルトのコンテナールートパーティションの最大サイズの設定

各コンテナーのルートパーティションには、基礎となるホストの利用可能なディスク領域がすべて表示されます。以下のガイダンスに従って、すべてのコンテナーのルートディスクの最大サイズを設定します。

Overlay の最大サイズや、ログレベルなどの他の CRI-O オプションを設定するには、以下の **ContainerRuntimeConfig** カスタムリソース定義 (CRD) を作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: ContainerRuntimeConfig
metadata:
 name: overlay-size
spec:
 machineConfigPoolSelector:
   matchLabels:
     custom-crio: overlay-size
 containerRuntimeConfig:
   logLevel: debug
   overlaySize: 8G
```

**手順**

1. 設定オブジェクトを作成します。

```
$ oc apply -f overlaysize.yml
```

2. 新規の CRI-O 設定をワーカーノードに適用するには、ワーカーのマシン設定プールを編集します。

```
$ oc edit machineconfigpool worker
```

3. **ContainerRuntimeConfig** CRD に設定した **matchLabels** 名に基づいて **custom-crio** ラベルを追加します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  creationTimestamp: "2020-07-09T15:46:34Z"
  generation: 3
  labels:
    custom-crio: overlay-size
    machineconfiguration.openshift.io/mco-built-in: ""
```

4. 変更を保存して、マシン設定を表示します。

```
$ oc get machineconfigs
```

新規の **99-worker-generated-containerruntime** および **rendered-worker-xyz** オブジェクトが作成されます。

**出力例**

```
99-worker-generated-containerruntime  4173030d89fbf4a7a0976d1665491a4d9a6e54f1  3.2.0             7m42s
rendered-worker-xyz                   4173030d89fbf4a7a0976d1665491a4d9a6e54f1  3.2.0             7m36s
```

5. これらのオブジェクトの作成後に、変更が適用されるようにマシン設定プールを監視します。

```
$ oc get mcp worker
```

ワーカーノードには、マシン数、更新数およびその他の詳細と共に **UPDATING** が **True** として表示されます。

**出力例**

```
NAME   CONFIG             UPDATED   UPDATING   DEGRADED   MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
worker rendered-worker-xyz False True False     3           2                   2                     0                   20h
```

完了すると、ワーカーノードは **UPDATING** を **False** に戻し、**UPDATEDMACHINECOUNT** 数は **MACHINECOUNT** に一致します。

**出力例**

```
NAME   CONFIG             UPDATED   UPDATING   DEGRADED   MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
worker   rendered-worker-xyz   True      False      False      3       3          3          0                   20h
```

ワーカーマシンを見ると、新規の 8 GB の最大サイズの設定がすべてのワーカーに適用されていることを確認できます。

**出力例**

```
head -n 7 /etc/containers/storage.conf
[storage]
  driver = "overlay"
  runroot = "/var/run/containers/storage"
  graphroot = "/var/lib/containers/storage"
  [storage.options]
    additionalimagestores = []
    size = "8G"
```

コンテナー内では、ルートパーティションが 8 GB であることを確認できます。

**出力例**

```
~ $ df -h
Filesystem              Size      Used Available Use% Mounted on
overlay                 8.0G      8.0K      8.0G   0% /
```

# 第5章 ブートイメージ更新

Machine Config Operator (MCO) は、ブートイメージを使用して Red Hat Enterprise Linux CoreOS (RHCOS) ノードを起動します。デフォルトでは、ブートイメージは OpenShift Container Platform によって管理されません。

そのため、クラスター内のブートイメージはクラスターとともに更新されません。たとえば、クラスターが元々 OpenShift Container Platform 4.12 で作成されていた場合、クラスターがノードを作成するために使用するブートイメージは、クラスターがそれ以降のバージョンであっても、同じ 4.12 バージョンになります。クラスターが後で 4.13 以降にアップグレードされた場合、新しいノードは同じ 4.12 イメージを使用してスケーリングを継続します。

このプロセスにより、以下の問題が発生する可能性があります。

- ノードの起動に余分に時間がかかる
- 証明書の有効期限の問題が発生する
- バージョンスキューの問題が発生する

これらの問題を回避するには、クラスターを更新するたびにブートイメージも更新するようにクラスターを設定できます。**MachineConfiguration** オブジェクトを変更することで、この機能を有効にできます。現在、ブートイメージを更新する機能は、Google Cloud Platform (GCP) クラスターでのみ使用できます。Cluster API によって管理されるクラスターではサポートされていません。

**重要**

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

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

クラスターで使用されている現在のブートイメージを表示するには、マシンセットを調べます。

### ブートイメージ参照を含むマシンセット例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: ci-ln-hmy310k-72292-5f87z-worker-a
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    spec:
# ...
      providerSpec:
# ...
        value:
```

```
        disks:
        - autoDelete: true
          boot: true
          image: projects/rhcos-cloud/global/images/rhcos-412-85-202203181601-0-gcp-x86-64 (1)
# ...
```

(1) このブートイメージは、クラスターの現在のバージョンに関係なく、最初にインストールされた OpenShift Container Platform バージョン (この例では OpenShift Container Platform 4.12) と同じです。**providerSpec** フィールドの構造はプラットフォームごとに異なるため、マシンセット内でブートイメージが表現される方法はプラットフォームによって異なります。

ブートイメージを更新するようにクラスターを設定すると、マシンセットで参照されるブートイメージはクラスターの現在のバージョンと一致します。

## 5.1. ブートイメージ更新の設定

デフォルトでは、ブートイメージは OpenShift Container Platform によって管理されません。**MachineConfiguration** オブジェクトを変更することで、クラスターを更新するたびにブートイメージが更新されるようにクラスターを設定できます。

### 前提条件

- フィーチャーゲートを使用して **TechPreviewNoUpgrade** 機能セットを有効にしている。詳細は、**関連情報** セクションの「フィーチャーゲートを使用した機能の有効化」を参照してください。

### 手順

1. 次のコマンドを実行して、**cluster** という名前の **MachineConfiguration** オブジェクトを編集し、ブートイメージの更新を有効にします。

```
$ oc edit MachineConfiguration cluster
```

   a. オプション: すべてのマシンセットのブートイメージ更新機能を設定します。

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
spec:
# ...
  managedBootImages: (1)
    machineManagers:
    - resource: machinesets
      apiGroup: machine.openshift.io
      selection:
        mode: All (2)
```

(1) ブートイメージ更新機能を有効にします。

(2) クラスター内のすべてのマシンセットを更新することを指定します。

b. オプション: 特定のマシンセットのブートイメージ更新機能を設定します。

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
spec:
# ...
  managedBootImages: 1
    machineManagers:
    - resource: machinesets
      apiGroup: machine.openshift.io
      selection:
        mode: Partial
          partial:
            machineResourceSelector:
              matchLabels:
                update-boot-image: "true" 2
```

1 ブートイメージ更新機能を有効にします。

2 このラベルが設定されたマシンすべてを更新するように指定します。

**ヒント**

マシンセットに適切なラベルが存在しない場合は、次のようなコマンドを実行してキー/値のペアを追加します。

```
$ oc label machineset.machine ci-ln-hmy310k-72292-5f87z-worker-a update-boot-image=true -n openshift-machine-api
```

**検証**

1. 次のコマンドを実行してブートイメージのバージョンを取得します。

```
$ oc get machinesets <machineset_name> -n openshift-machine-api -o yaml
```

**ブートイメージ参照を含むマシンセット例**

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: ci-ln-77hmkpt-72292-d4pxp
    update-boot-image: "true"
  name: ci-ln-77hmkpt-72292-d4pxp-worker-a
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    spec:
```

```
# ...
    providerSpec:
# ...
      value:
        disks:
        - autoDelete: true
          boot: true
          image: projects/rhcos-cloud/global/images/rhcos-416-92-202402201450-0-gcp-x86-
64 (1)
# ...
```

(1) このブートイメージは、現在の OpenShift Container Platform バージョンと同じです。

**関連情報**

- フィーチャーゲートを使用した機能の有効化

## 5.2. ブートイメージ更新の無効化

ブートイメージ更新機能を無効にするには、**MachineConfiguration** オブジェクトを編集して **managedBootImages** スタンザを削除します。

新しいブートイメージバージョンでいくつかのノードが作成された後にこの機能を無効にすると、既存のノードは現在のブートイメージを保持します。この機能をオフにしても、ノードまたはマシンセットは元々インストールされたブートイメージにロールバックされません。マシンセットは、機能が有効になったときに存在していたブートイメージバージョンを保持し、今後クラスターが新しい OpenShift Container Platform バージョンにアップグレードされても再度更新されません。

**手順**

1. **MachineConfiguration** オブジェクトを編集して、ブートイメージ更新を無効にします。

```
$ oc edit MachineConfiguration cluster
```

2. **managedBootImages** スタンザを削除します。

```
apiVersion: operator.openshift.io/v1
kind: MachineConfiguration
metadata:
  name: cluster
  namespace: openshift-machine-config-operator
spec:
# ...
  managedBootImages: (1)
    machineManagers:
    - resource: machinesets
      apiGroup: machine.openshift.io
      selection:
        mode: All
```

(1) ブートイメージ更新を無効にするには、スタンザ全体を削除します。

# 第6章 未使用のレンダリング済みマシン設定の管理

Machine Config Operator (MCO) は、ガベージコレクションアクティビティーを実行しません。つまり、レンダリングされたすべてのマシン設定はクラスター内に残ります。ユーザーまたはコントローラーが新しいマシン設定を適用するたびに、MCO は影響を受けるマシン設定プールごとに新しいレンダリングされた設定を作成します。時間が経つにつれて、レンダリング済みマシン設定の数が多くなり、マシン設定での使用が混乱する可能性があります。レンダリング済みマシン設定が多すぎると、ディスク容量の問題や etcd のパフォーマンスの問題が発生する可能性もあります。

**--confirm** フラグを指定した **oc adm prune renderedmachineconfigs** コマンドを使用すると、古い未使用のレンダリング済みマシン設定を削除できます。このコマンドを使用すると、未使用のレンダリング済みマシン設定をすべて削除することも、特定のマシン設定プール内の未使用のレンダリング済みマシン設定のみを削除することもできます。また、古い設定を確認したい場合に備えて、古いマシン設定を保持するために、指定された数の未使用のレンダリング済みマシン設定を削除することもできます。

**--confirm** フラグなしで **oc adm prune renderedmachineconfigs** コマンドを使用すると、削除されるレンダリング済みマシン設定を確認できます。

**list** サブコマンドを使用して、クラスター内または特定のマシン設定プール内のすべてのレンダリング済みマシン設定を表示します。

**注記**

**oc adm prune renderedmachineconfigs** コマンドは、使用されていないレンダリング済みマシン設定のみを削除します。レンダリング済みマシン設定がマシン設定プールによって使用されている場合、レンダリング済みマシン設定は削除されません。この場合、コマンドは、レンダリング済みマシン設定が削除されなかった理由を詳細に出力します。

## 6.1. レンダリング済みマシン設定の表示

**list** サブコマンドを指定した **oc adm prune renderedmachineconfigs** コマンドを使用すると、レンダリング済みマシン設定のリストを表示できます。

たとえば、次の手順のコマンドは、**worker** マシン設定プールのすべてのレンダリング済みマシン設定をリスト表示します。

**手順**

- オプション: 次のコマンドを使用して、レンダリング済みマシン設定をリスト表示します。

```
$ oc adm prune renderedmachineconfigs list --in-use=false --pool-name=worker
```

ここでは、以下のようになります。

**list**
: クラスター内のレンダリング済みマシン設定のリストを表示します。

**--in-use**
: オプション: 指定されたプールから、使用済みのマシン設定のみを表示するか、すべてのマシン設定を表示するかを指定します。**true** の場合、出力には、マシン設定プールによって使用されているレンダリング済みマシン設定がリスト表示されます。**false** の場合、出力にはクラスター内のすべてのレンダリング済みマシン設定がリスト表示されます。デフォルト値は **false** です。

**--pool-name**

オプション: マシン設定を表示するマシン設定プールを指定します。

**出力例**

```
worker
status: rendered-worker-ae115e2b5e6ae05e0e6e5d62c7d0dd81
spec: rendered-worker-ae115e2b5e6ae05e0e6e5d62c7d0dd81
```

- 次のコマンドを実行して、自動的に削除できるレンダリング済みマシン設定をリスト表示します。コマンド出力で **as it's currently in use** とマークされているレンダリング済みマシン設定は削除されません。

```
$ oc adm prune renderedmachineconfigs --pool-name=worker
```

コマンドはドライランモードで実行され、マシン設定はどれも削除されません。

ここでは、以下のようになります。

**--pool-name**

オプション: 指定されたマシン設定プール内のマシン設定を表示します。

**出力例**

```
Dry run enabled - no modifications will be made. Add --confirm to remove rendered machine
configs.
DRY RUN: Deleted rendered MachineConfig rendered-worker-
23d7322831a57f02998e7e1600a0865f
DRY RUN: Deleted rendered MachineConfig rendered-worker-
fc94397dc7c43808c7014683c208956e
DRY RUN: Skipping deletion of rendered MachineConfig rendered-worker-
ad5a3cad36303c363cf458ab0524e7c0 as it's currently in use
```

## 6.2. 未使用のレンダリング済みマシン設定の削除

**--confirm** コマンドを指定した **oc adm prune renderedmachineconfigs** コマンドを使用すると、未使用のレンダリング済みマシン設定を削除できます。削除されていないレンダリング済みマシン設定がある場合、コマンド出力には削除されなかった設定が示され、削除されなかった理由がリスト表示されます。

**手順**

1. オプション: 次のコマンドを実行して、自動的に削除できるレンダリング済みマシン設定をリスト表示します。コマンド出力で **as it's currently in use** とマークされているレンダリング済みマシン設定は削除されません。

```
$ oc adm prune renderedmachineconfigs --pool-name=worker
```

**出力例**

```
Dry run enabled - no modifications will be made. Add --confirm to remove rendered machine
configs.
```

```
DRY RUN: Deleted rendered MachineConfig rendered-worker-
23d7322831a57f02998e7e1600a0865f
DRY RUN: Deleted rendered MachineConfig rendered-worker-
fc94397dc7c43808c7014683c208956e
DRY RUN: Skipping deletion of rendered MachineConfig rendered-worker-
ad5a3cad36303c363cf458ab0524e7c0 as it's currently in use
```

ここでは、以下のようになります。

pool-name
: オプション: マシン設定を削除するマシン設定プールを指定します。

2. 次のコマンドを実行して、未使用のレンダリング済みマシン設定を削除します。次の手順のコマンドは、**worker** マシン設定プール内の最も古い 2 つの未使用のレンダリング済みマシン設定を削除します。

```
$ oc adm prune renderedmachineconfigs --pool-name=worker --count=2 --confirm
```

ここでは、以下のようになります。

**--count**
: オプション: 最も古いものから順に、削除する未使用のレンダリング済みマシン設定の最大数を指定します。

**--confirm**
: オプション: 最も古いものから順に、削除する未使用のレンダリング済みマシン設定の最大数を指定します。

**--pool-name**
: オプション: マシンを削除するマシン設定プールを指定します。指定しない場合は、すべてのプールが評価されます。

# 第7章 RHCOS イメージのレイヤー化

Red Hat Enterprise Linux CoreOS (RHCOS) イメージのレイヤー化を使用すると、ベースイメージの上に追加のイメージを **重ねる** ことで、ベース RHCOS イメージの機能を簡単に拡張できます。この階層化は、RHCOS のベースイメージを変更しません。代わりに、すべての RHCOS 機能を含む **カスタムレイヤーイメージ** を作成し、クラスター内の特定のノードに追加機能を追加します。

## 7.1. RHCOS イメージのレイヤー化について

イメージのレイヤー化を使用すると、クラスターワーカーノードの基盤となるノードオペレーティングシステムをカスタマイズできます。これにより、ノードのオペレーティングシステムや特殊なソフトウェアなどのカスタマイズの追加を含め、すべてを最新の状態に保つことができます。

Containerfile を使用してカスタムレイヤーイメージを作成し、カスタムオブジェクトを使用してそれをノードに適用します。カスタムレイヤーイメージは、カスタムオブジェクトを削除することでいつでも削除できます。

RHCOS イメージのレイヤー化を使用すると、RPM を基本イメージにインストールでき、カスタムコンテンツが RHCOS と一緒に起動されます。Machine Config Operator (MCO) は、デフォルトの RHCOS イメージの場合と同じ方法で、これらのカスタムレイヤーイメージをロールアウトし、これらのカスタムコンテナーを監視できます。RHCOS イメージのレイヤー化により、RHCOS ノードの管理方法がより柔軟になります。

**重要**

リアルタイムカーネルと拡張機能の RPM をカスタムレイヤードコンテンツとしてインストールすることは推奨しません。これは、これらの RPM が、マシン設定を使用してインストールされた RPM と競合する可能性があるためです。競合がある場合、MCO がマシン設定 RPM をインストールしようとすると、**degraded** 状態になります。続行する前に、競合する拡張機能をマシン設定から削除する必要があります。

カスタムレイヤーイメージをクラスターに適用するとすぐに、カスタムレイヤーイメージとそれらのノードの **所有権** を効率的に取得できます。Red Hat は引き続き標準ノード上の RHCOS の基本イメージの維持と更新を担当しますが、カスタムレイヤーイメージを使用するノードのイメージの維持と更新はお客様の責任となります。カスタムレイヤーイメージで適用したパッケージと、パッケージで発生する可能性のある問題については、お客様が責任を負うものとします。

カスタムレイヤーイメージをノードにデプロイする方法には、次の 2 つのものがあります。

**クラスター上でのレイヤー化**

クラスター上でのレイヤー化 では、**MachineOSConfig** オブジェクトを作成し、そのオブジェクトに Containerfile やその他のパラメーターを追加します。ビルドはクラスター上で実行します。作成されるカスタムレイヤーイメージは、リポジトリーに自動的にプッシュされ、**MachineOSConfig** オブジェクトで指定したマシン設定プールに適用されます。プロセス全体がクラスター内で完全に実行されます。

**重要**

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

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

**クラスター外でのレイヤー化**

クラスター外でのレイヤー化 では、OpenShift Container Platform イメージと適用する RPM を参照する Containerfile を作成して、独自の環境でレイヤーイメージをビルドし、そのイメージをリポジトリーにプッシュします。その後、クラスターで、新しいイメージを参照する対象ノードプールの **MachineConfig** オブジェクトを作成します。Machine Config Operator は、関連付けられたマシン設定の **osImageURL** 値で指定されているように、RHCOS の基本イメージをオーバーライドし、新しいイメージを起動します。

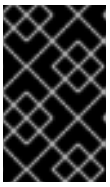
**重要**

どちらの方法でも、クラスターの残りの部分にインストールされている同じベース RHCOS イメージを使用します。クラスターで使用されるベースイメージを取得するには、**oc adm release info --image-for rhel-coreos** コマンドを使用します。

## 7.2. CONTAINERFILE の例

RHCOS イメージのレイヤー化により、次のタイプのイメージを使用してカスタムレイヤーイメージを作成できます。

- **OpenShift Container Platform ホットフィックス**。Customer Experience and Engagement (CEE) を使用して、ホットフィックスパッケージ を取得し、RHCOS イメージに適用することができます。場合によっては、公式の OpenShift Container Platform リリースに含まれる前に、バグ修正または機能強化が必要になることがあります。RHCOS イメージのレイヤー化により、公式にリリースされる前にホットフィックスを簡単に追加し、基になる RHCOS イメージに修正が組み込まれたときにホットフィックスを削除できます。

  **重要**

  一部のホットフィックスは Red Hat Support Exception を必要とし、OpenShift Container Platform のサポート範囲またはライフサイクルポリシーの通常の範囲外です。

  ホットフィックスは Red Hat ホットフィックスポリシー に基づいて提供されます。それを基本イメージ上に適用し、その新しいカスタムレイヤーイメージを非実稼働環境でテストします。カスタムレイヤーイメージが実稼働環境で安全に使用できることを確認したら、独自のスケジュールで特定のノードプールにロールアウトできます。何らかの理由で、カスタムレイヤーイメージを簡単にロールバックして、デフォルトの RHCOS の使用に戻すことができます。

  **ホットフィックスを適用するクラスター上の Containerfile の例**

  ```
  # Using a 4.12.0 image
  ```

```
containerfileArch: noarch
content: |-
  FROM configs AS final
  #Install hotfix rpm
  RUN rpm-ostree override replace https://example.com/myrepo/haproxy-1.0.16-5.el8.src.rpm
&& \
      rpm-ostree cleanup -m && \
      ostree container commit
```

**ホットフィックスを適用するクラスター外の Containerfile の例**

```
# Using a 4.12.0 image
FROM quay.io/openshift-release-dev/ocp-release@sha256...
#Install hotfix rpm
RUN rpm-ostree override replace https://example.com/myrepo/haproxy-1.0.16-5.el8.src.rpm
&& \
    rpm-ostree cleanup -m && \
    ostree container commit
```

- **RHEL パッケージ**。chrony、firewalld、iputils などの Red Hat Enterprise Linux (RHEL) パッケージは、Red Hat Customer Portal からダウンロードできます。

**libreswan ユーティリティーを適用するクラスター外の Containerfile の例**

```
# Get RHCOS base image of target cluster `oc adm release info --image-for rhel-coreos`
# hadolint ignore=DL3006
FROM quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256...

# Install our config file
COPY my-host-to-host.conf /etc/ipsec.d/

# RHEL entitled host is needed here to access RHEL packages
# Install libreswan as extra RHEL package
RUN rpm-ostree install libreswan && \
    systemctl enable ipsec && \
    ostree container commit
```

libreswan には追加の RHEL パッケージが必要なため、イメージはエンタイトルメントのある RHEL ホストでビルドする必要があります。RHEL エンタイトルメントを機能させるには、**etc-pki-entitlement** シークレットを **openshift-machine-api** namespace にコピーする必要があります。

- **サードパーティーのパッケージ**。次のタイプのパッケージなど、サードパーティーから RPM をダウンロードおよびインストールできます。
  - 最先端のドライバーとカーネルの強化により、パフォーマンスを向上させたり、機能を追加したりします。
  - 侵入の可能性と実際の侵入を調査するためのフォレンジッククライアントツール。
  - セキュリティーエージェント。
  - クラスター全体の一貫性のあるビューを提供するインベントリーエージェント。
  - SSH キー管理パッケージ。

### EPEL からのサードパーティーパッケージを適用するクラスター上の Containerfile の例

```
FROM configs AS final

#Enable EPEL (more info at https://docs.fedoraproject.org/en-US/epel/ ) and install htop
RUN rpm-ostree install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
&& \
    rpm-ostree install htop && \
    ostree container commit
```

### EPEL からサードパーティーパッケージを適用するクラスター外の Containerfile の例

```
# Get RHCOS base image of target cluster `oc adm release info --image-for rhel-coreos`
# hadolint ignore=DL3006
FROM quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256...

# Install our config file
COPY my-host-to-host.conf /etc/ipsec.d/

# RHEL entitled host is needed here to access RHEL packages
# Install libreswan as extra RHEL package
RUN rpm-ostree install libreswan && \
    systemctl enable ipsec && \
    ostree container commit
```

この Containerfile は、RHEL fish プログラムをインストールします。fish には追加の RHEL パッケージが必要なため、イメージはエンタイトルメントのある RHEL ホストでビルドする必要があります。RHEL エンタイトルメントを機能させるには、**etc-pki-entitlement** シークレットを **openshift-machine-api** namespace にコピーする必要があります。

### RHEL 依存関係を持つサードパーティーパッケージを適用するクラスター上の Containerfile の例

```
FROM configs AS final

# RHEL entitled host is needed here to access RHEL packages
# Install fish as third party package from EPEL
RUN rpm-ostree install
https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/Packages/f/fish-3.3.1-
3.el9.x86_64.rpm && \
    ostree container commit
```

### RHEL 依存関係を持つサードパーティーパッケージを適用するクラスター外の Containerfile の例

```
# Get RHCOS base image of target cluster `oc adm release info --image-for rhel-coreos`
# hadolint ignore=DL3006
FROM quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256...

# Install our config file
COPY my-host-to-host.conf /etc/ipsec.d/

# RHEL entitled host is needed here to access RHEL packages
```

```
# Install libreswan as extra RHEL package
RUN rpm-ostree install libreswan && \
    systemctl enable ipsec && \
    ostree container commit
```

マシン設定を作成した後、Machine Config Operator (MCO) は次の手順を実行します。

1. 指定された 1 つ以上のプールの新しいマシン設定をレンダリングします。
2. 1 つ以上のプール内のノードでコードンおよびドレイン操作を実行します。
3. 残りのマシン設定パラメーターをノードに書き込みます。
4. カスタムレイヤーイメージをノードに適用します。
5. 新しいイメージを使用してノードを再起動します。

**重要**

クラスターにロールアウトする前に、実稼働環境の外でイメージをテストすることを強く推奨します。

## 7.3. クラスター上でのレイヤー化を使用してカスタムレイヤーイメージを適用する

クラスター上でのビルドプロセスを使用してクラスターにカスタムレイヤーイメージを適用するには、**Containerfile**、マシン設定プールの参照、リポジトリーのプッシュおよびプルシークレット、その他のパラメーターを含む MachineOSConfig カスタムリソースを作成します。これについては、前提条件を参照してください。

オブジェクトを作成すると、Machine Config Operator (MCO) によって **MachineOSBuild** オブジェクトと **machine-os-builder** Pod が作成されます。ビルドプロセスでは、設定マップなどの一時的なオブジェクトも作成されますが、これらはビルドの完了後にクリーンアップされます。

ビルドが完了すると、MCO は新しいカスタムレイヤーイメージをリポジトリーにプッシュし、新しいノードのデプロイ時に使用できるようにします。新しいカスタムレイヤーイメージのダイジェスト付きイメージプル仕様を、**MachineOSBuild** オブジェクトと **machine-os-builder** Pod で確認できます。

これらの新しいオブジェクトや **machine-os-builder** Pod を操作する必要はありません。ただし、これらのリソースは、必要に応じて、すべてトラブルシューティングに使用できます。

カスタムレイヤーイメージを使用するマシン設定プールごとに、個別の **MachineOSConfig** CR が必要です。

**重要**

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

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

**前提条件**

- フィーチャーゲートを使用して **TechPreviewNoUpgrade** 機能セットを有効にしている。詳細は、「フィーチャーゲートを使用した機能の有効化」を参照してください。
- MCO がベースオペレーティングシステムイメージをプルするために必要なプルシークレットが **openshift-machine-config-Operator** namespace にある。
- MCO が新しいカスタムレイヤーイメージをレジストリーにプッシュするために必要なプッシュシークレットがある。
- ノードがレジストリーから新しいカスタムレイヤーイメージをプルするために必要なプルシークレットがある。これは、イメージをリポジトリーにプッシュするために使用するシークレットとは異なるシークレットである必要があります。
- Containerfile を設定する方法をよく理解している。Containerfile の作成方法については、このドキュメントの範囲外です。
- オプション: カスタムレイヤーイメージを適用するノード用に、別のマシン設定プールがある。

**手順**

1. **machineOSconfig** オブジェクトを作成します。
   a. 以下のような YAML ファイルを作成します。

```
apiVersion: machineconfiguration.openshift.io/v1alpha1
kind: MachineOSConfig
metadata:
  name: layered
spec:
  machineConfigPool:
    name: <mcp_name> 1
  buildInputs:
    containerFile: 2
    - containerfileArch: noarch
      content: |-
        FROM configs AS final
        RUN rpm-ostree install cowsay && \
          ostree container commit
    imageBuilder: 3
      imageBuilderType: PodImageBuilder
    baseImagePullSecret: 4
      name: global-pull-secret-copy
    renderedImagePushspec: image-registry.openshift-image-
registry.svc:5000/openshift/os-image:latest 5
    renderedImagePushSecret: 6
      name: builder-dockercfg-7lzwl
  buildOutputs: 7
    currentImagePullSecret:
      name: builder-dockercfg-7lzwl
```

1 カスタムレイヤーイメージをデプロイするノードに関連付けられているマシン設定プールの名前を指定します。

2 カスタムレイヤーイメージを設定するための Containerfile を指定します。

3 使用する Image Builder の名前を指定します。これは **PodImageBuilder** である必要があります。

4 MCO がレジストリーからベースオペレーティングシステムイメージをプルするために必要なプルシークレットの名前を指定します。

5 新しくビルドされたカスタムレイヤーイメージをプッシュするイメージレジストリーを指定します。これは、クラスターがアクセスできる任意のレジストリーにすることができます。この例では、内部 OpenShift Container Platform レジストリーを使用します。

6 MCO が新しくビルドされたカスタムレイヤーイメージをそのレジストリーにプッシュするために必要なプッシュシークレットの名前を指定します。

7 新しくビルドされたカスタムレイヤーイメージをノードがプルするために必要なイメージレジストリーに必要なシークレットを指定します。これは、イメージをリポジトリーにプッシュするために使用するシークレットとは異なるシークレットである必要があります。

b. **MachineOSConfig** オブジェクトを作成します。

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

2. **MachineOSBuild** オブジェクトが作成されて **READY** 状態になったら、必要に応じて、新しいカスタムレイヤーイメージを使用するノードのノード仕様を変更します。

a. **MachineOSBuild** オブジェクトが **READY** であることを確認します。**SUCCEEDED** 値が **True** になったら、ビルドは完了です。

```
$ oc get machineosbuild
```

**MachineOSBuild オブジェクトが準備完了状態であることを示す出力例**

```
NAME                                                                PREPARED   BUILDING   SUCCEEDED   INTERRUPTED   FAILED
layered-rendered-layered-ad5a3cad36303c363cf458ab0524e7c0-builder   False      True       False       False         False
```

b. **MachineOSConfig** オブジェクトで指定したマシン設定プールのラベルを追加して、カスタムレイヤーイメージをデプロイするノードを編集します。

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

ここでは、以下のようになります。

**node-role.kubernetes.io/<mcp_name>=**

カスタムレイヤーイメージをデプロイするノードを特定するノードセレクターを指定します。

変更を保存すると、MCO がノードをドレイン、遮断、再起動します。再起動後、ノードは新しいカスタムレイヤーイメージを使用します。

**検証**

1. 次のコマンドを使用して、新しい Pod が実行されていることを確認します。

```
$ oc get pods -n <machineosbuilds_namespace>
```

```
NAME                                                READY   STATUS    RESTARTS   AGE
build-rendered-layered-ad5a3cad36303c363cf458ab0524e7c0   2/2     Running   0
2m40s (1)
# ...
machine-os-builder-6fb66cfb99-zcpvq                 1/1     Running   0          2m42s (2)
```

(1) これは、カスタムレイヤーイメージがビルドされるビルド Pod です。

(2) この Pod はトラブルシューティングに使用できます。

2. **MachineOSConfig** オブジェクトに新しいカスタムレイヤーイメージへの参照が含まれていることを確認します。

```
$ oc describe MachineOSConfig <object_name>
```

```
apiVersion: machineconfiguration.openshift.io/v1alpha1
kind: MachineOSConfig
metadata:
  name: layered
spec:
  buildInputs:
    baseImagePullSecret:
      name: global-pull-secret-copy
    containerFile:
    - containerfileArch: noarch
      content: ""
    imageBuilder:
      imageBuilderType: PodImageBuilder
    renderedImagePushSecret:
      name: builder-dockercfg-ng82t-canonical
    renderedImagePushspec: image-registry.openshift-image-registry.svc:5000/openshift-
machine-config-operator/os-image:latest
  buildOutputs:
    currentImagePullSecret:
      name: global-pull-secret-copy
  machineConfigPool:
    name: layered
status:
  currentImagePullspec: image-registry.openshift-image-registry.svc:5000/openshift-machine-
config-operator/os-
image@sha256:f636fa5b504e92e6faa22ecd71a60b089dab72200f3d130c68dfec07148d11cd
(1)
```

(1) 新しいカスタムレイヤーイメージのダイジェスト付きイメージプル仕様。

3. **MachineOSBuild** オブジェクトに新しいカスタムレイヤーイメージへの参照が含まれていることを確認します。

```
$ oc describe machineosbuild <object_name>
```

```
apiVersion: machineconfiguration.openshift.io/v1alpha1
kind: MachineOSBuild
metadata:
  name: layered-rendered-layered-ad5a3cad36303c363cf458ab0524e7c0-builder
spec:
  desiredConfig:
    name: rendered-layered-ad5a3cad36303c363cf458ab0524e7c0
  machineOSConfig:
    name: layered
  renderedImagePushspec: image-registry.openshift-image-registry.svc:5000/openshift-machine-config-operator/os-image:latest
# ...
status:
  conditions:
    - lastTransitionTime: "2024-05-21T20:25:06Z"
      message: Build Ready
      reason: Ready
      status: "True"
      type: Succeeded
  finalImagePullspec: image-registry.openshift-image-registry.svc:5000/openshift-machine-config-operator/os-image@sha256:f636fa5b504e92e6faa22ecd71a60b089dab72200f3d130c68dfec07148d11cd (1)
```

(1) 新しいカスタムレイヤーイメージのダイジェスト付きイメージプル仕様。

4. 適切なノードが新しいカスタムレイヤーイメージを使用していることを確認します。
   a. コントロールプレーンノードの root としてデバッグセッションを開始します。

      ```
      $ oc debug node/<node_name>
      ```

   b. **/host** をデバッグシェル内のルートディレクトリーとして設定します。

      ```
      sh-4.4# chroot /host
      ```

   c. **rpm-ostree status** コマンドを実行して、カスタムレイヤーイメージが使用されていることを確認します。

      ```
      sh-5.1# rpm-ostree status
      ```

      **出力例**

      ```
      # ...
      Deployments:
      * ostree-unverified-registry:quay.io/openshift-release-dev/os-image@sha256:f636fa5b504e92e6faa22ecd71a60b089dab72200f3d130c68dfec07148d11cd (1)
                       Digest: sha256:bcea2546295b2a55e0a9bf6dd4789433a9867e378661093b6fdee0031ed1e8a4
                  Version: 416.94.202405141654-0 (2024-05-14T16:58:43Z)
      ```

- 

1 新しいカスタムレイヤーイメージのダイジェスト付きイメージプル仕様。

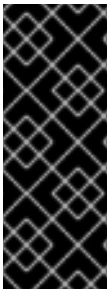
**関連情報**

- フィーチャーゲートを使用した機能の有効化

## 7.4. クラスター外でのレイヤー化を使用してカスタムレイヤーイメージを適用する

特定のマシン設定プール内のノードで、Red Hat Enterprise Linux CoreOS (RHCOS) イメージのレイヤー化を簡単に設定できます。Machine Config Operator (MCO) は、ベースの Red Hat Enterprise Linux CoreOS (RHCOS) イメージを上書きして、新しいカスタムレイヤーイメージでこれらのノードを再起動します。

カスタムレイヤーイメージをクラスターに適用するには、クラスターがアクセスできるリポジトリーにカスタムレイヤーイメージが必要です。次に、カスタムレイヤーイメージを指す **MachineConfig** オブジェクトを作成します。設定するマシン設定プールごとに個別の **MachineConfig** オブジェクトが必要です。



**重要**

カスタムレイヤーイメージを設定すると、OpenShift Container Platform は、カスタムレイヤーイメージを使用するノードを自動的に更新しなくなりました。必要に応じてノードを手動で更新する必要があります。カスタムレイヤーをロールバックすると、OpenShift Container Platform は再びノードを自動的に更新します。カスタムレイヤーイメージを使用するノードの更新に関する重要な情報については、以下の追加リソースセクションを参照してください。

**前提条件**

- タグではなく、OpenShift Container Platform イメージダイジェストに基づくカスタムレイヤーイメージを作成する必要があります。

**注記**

クラスターの残りの部分にインストールされているのと同じ RHCOS の基本イメージを使用する必要があります。**oc adm release info --image-for rhel-coreos** コマンドを使用して、クラスターで使用されている基本イメージを取得します。

たとえば、次の Containerfile は、OpenShift Container Platform 4.16 イメージからカスタムレイヤーイメージを作成し、カーネルパッケージを CentOS 9 Stream のカーネルパッケージでオーバーライドします。

**カスタムレイヤーイメージの Containerfile の例**

```
# Using a 4.16.0 image
FROM quay.io/openshift-release/ocp-release@sha256... 1
#Install hotfix rpm
RUN rpm-ostree override replace http://mirror.stream.centos.org/9-stream/BaseOS/x86_64/os/Packages/kernel-{,core-,modules-,modules-core-,modules-extra-
```

```
}5.14.0-295.el9.x86_64.rpm && \ 2
    rpm-ostree cleanup -m && \
    ostree container commit
```

1 クラスターの RHCOS 基本イメージを指定します。

2 カーネルパッケージを置き換えます。

**注記**

Containerfile の作成方法については、このドキュメントの範囲外です。

- カスタムレイヤーイメージをビルドするプロセスはクラスターの外部で実行されるため、Podman または Buildah で **--authfile/path/to/pull-secret** オプションを使用する必要があります。あるいは、これらのツールでプルシークレットを自動的に読み取るようにするには、デフォルトのファイルの場所のいずれかに追加できます。**~/.docker/config.json**、**$XDG_RUNTIME_DIR/containers/auth.json**、**~/.docker/config.json**、または **~/.dockercfg**。詳細は、**containers-auth.json** のマニュアルページを参照してください。
- カスタムレイヤーイメージを、クラスターがアクセスできるリポジトリーにプッシュする必要があります。

**手順**

1. マシン設定プールを作成します。
   a. 以下のような YAML ファイルを作成します。

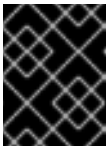
```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker 1
  name: os-layer-custom
spec:
  osImageURL: quay.io/my-registry/custom-image@sha256... 2
```

1 カスタムレイヤーイメージを適用するマシン設定プールを指定します。

2 リポジトリー内のカスタムレイヤーイメージへのパスを指定します。

   b. **MachineConfig** オブジェクトを作成します。

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



**重要**

クラスターにロールアウトする前に、実稼働環境の外でイメージをテストすることを強く推奨します。

### 検証

次のチェックのいずれかを実行することで、カスタムレイヤーイメージが適用されていることを確認できます。

1. ワーカーマシン設定プールが新しいマシン設定でロールアウトされていることを確認します。

   a. 新しいマシン設定が作成されたことを確認します。

   ```
   $ oc get mc
   ```

   **出力例**

   ```
   NAME                                               GENERATEDBYCONTROLLER
   IGNITIONVERSION   AGE
   00-master                                          5bdb57489b720096ef912f738b46330a8f577803
   3.2.0             95m
   00-worker                                          5bdb57489b720096ef912f738b46330a8f577803
   3.2.0             95m
   01-master-container-runtime
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             95m
   01-master-kubelet                                  5bdb57489b720096ef912f738b46330a8f577803
   3.2.0             95m
   01-worker-container-runtime
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             95m
   01-worker-kubelet                                  5bdb57489b720096ef912f738b46330a8f577803
   3.2.0             95m
   99-master-generated-registries
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             95m
   99-master-ssh                                       3.2.0             98m
   99-worker-generated-registries
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             95m
   99-worker-ssh                                       3.2.0             98m
   os-layer-custom                                                       10s  (1)
   rendered-master-15961f1da260f7be141006404d17d39b
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             95m
   rendered-worker-5aff604cb1381a4fe07feaf1595a797e
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             95m
   rendered-worker-5de4837625b1cbc237de6b22bc0bc873
   5bdb57489b720096ef912f738b46330a8f577803   3.2.0             4s  (2)
   ```

   (1) 新しいマシン設定

   (2) 新しいレンダリング済みマシン設定

   b. 新しいマシン設定の **osImageURL** 値が予測されるイメージを指していることを確認します。

   ```
   $ oc describe mc rendered-master-4e8be63aef68b843b546827b6ebe0913
   ```

   **出力例**

   ```
   Name:         rendered-master-4e8be63aef68b843b546827b6ebe0913
   Namespace:
   ```

```
Labels:       <none>
Annotations:  machineconfiguration.openshift.io/generated-by-controller-version:
8276d9c1f574481043d3661a1ace1f36cd8c3b62
              machineconfiguration.openshift.io/release-image-version: 4.16.0-ec.3
API Version:  machineconfiguration.openshift.io/v1
Kind:         MachineConfig
...
  Os Image URL: quay.io/my-registry/custom-image@sha256...
```

c. 関連するマシン設定プールが新しいマシン設定で更新されていることを確認します。

```
$ oc get mcp
```

**出力例**

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
master   rendered-master-6faecdfa1b25c114a58cf178fbaa45e2   True      False      False
3              3                   3                   0                      39m
worker   rendered-worker-6b000dbc31aaee63c6a2d56d04cd4c1b   False     True
False      3              0                   0                   0                      39m  1
```

1 **UPDATING** フィールドが **True** の場合、マシン設定プールは新しいマシン設定で更新されます。フィールドが **False** になると、ワーカーマシン設定プールが新しいマシン設定にロールアウトされます。

d. ノードをチェックして、ノードのスケジューリングが無効になっていることを確認します。これは、変更が適用されていることを示しています。

```
$ oc get nodes
```

**出力例**

```
NAME                                         STATUS                     ROLES                  AGE
VERSION
ip-10-0-148-79.us-west-1.compute.internal    Ready                      worker                 32m
v1.29.4
ip-10-0-155-125.us-west-1.compute.internal   Ready,SchedulingDisabled   worker
35m   v1.29.4
ip-10-0-170-47.us-west-1.compute.internal    Ready                      control-plane,master
42m   v1.29.4
ip-10-0-174-77.us-west-1.compute.internal    Ready                      control-plane,master
42m   v1.29.4
ip-10-0-211-49.us-west-1.compute.internal    Ready                      control-plane,master
42m   v1.29.4
ip-10-0-218-151.us-west-1.compute.internal   Ready                      worker                 31m
v1.29.4
```

2. ノードが **Ready** 状態に戻ったら、ノードがカスタムレイヤーイメージを使用していることを確認します。

a. ノードへの **oc debug** セッションを開きます。以下に例を示します。

```
$ oc debug node/ip-10-0-155-125.us-west-1.compute.internal
```

b. **/host** をデバッグシェル内のルートディレクトリーとして設定します。

```
sh-4.4# chroot /host
```

c. **rpm-ostree status** コマンドを実行して、カスタムレイヤーイメージが使用されていることを確認します。

```
sh-4.4# sudo rpm-ostree status
```

**出力例**

```
State: idle
Deployments:
* ostree-unverified-registry:quay.io/my-registry/...
                   Digest: sha256:...
```

**関連情報**

RHCOS カスタムレイヤーイメージによる更新

## 7.5. RHCOS カスタムレイヤーイメージの削除

特定のマシン設定プール内のノードから、Red Hat Enterprise Linux CoreOS (RHCOS) イメージのレイヤー化を簡単に元に戻すことができます。Machine Config Operator (MCO) は、クラスターベースの Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用してこれらのノードを再起動し、カスタムレイヤーイメージをオーバーライドします。

クラスターから Red Hat Enterprise Linux CoreOS (RHCOS) カスタムレイヤーイメージを削除するには、イメージを適用したマシン設定を削除する必要があります。

**手順**

1. カスタムレイヤーイメージを適用したマシン設定を削除します。

```
$ oc delete mc os-layer-custom
```

マシン設定を削除した後、ノードが再起動します。

**検証**

次のチェックのいずれかを実行することで、カスタムレイヤーイメージが削除されたことを確認できます。

1. ワーカーマシン設定プールが以前のマシン設定で更新されていることを確認します。

```
$ oc get mcp
```

**出力例**

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
```

```
DEGRADEDMACHINECOUNT   AGE
master   rendered-master-6faecdfa1b25c114a58cf178fbaa45e2   True      False      False
3          3                 3                   0                  39m
worker   rendered-worker-6b000dbc31aaee63c6a2d56d04cd4c1b   False     True       False
3          0                 0                   0                  39m (1)
```

1. **UPDATING** フィールドが **True** の場合、マシン設定プールは以前のマシン設定で更新されます。フィールドが **False** になると、ワーカーマシン設定プールが以前のマシン設定にロールアウトされます。

2. ノードをチェックして、ノードのスケジューリングが無効になっていることを確認します。これは、変更が適用されていることを示しています。

```
$ oc get nodes
```

**出力例**

```
NAME                                         STATUS                     ROLES                  AGE   VERSION
ip-10-0-148-79.us-west-1.compute.internal    Ready                      worker                 32m
v1.29.4
ip-10-0-155-125.us-west-1.compute.internal   Ready,SchedulingDisabled   worker
35m   v1.29.4
ip-10-0-170-47.us-west-1.compute.internal    Ready                      control-plane,master   42m
v1.29.4
ip-10-0-174-77.us-west-1.compute.internal    Ready                      control-plane,master   42m
v1.29.4
ip-10-0-211-49.us-west-1.compute.internal    Ready                      control-plane,master   42m
v1.29.4
ip-10-0-218-151.us-west-1.compute.internal   Ready                      worker                 31m
v1.29.4
```

3. ノードが **Ready** 状態に戻ったら、ノードが基本イメージを使用していることを確認します。

   a. ノードへの **oc debug** セッションを開きます。以下に例を示します。

```
$ oc debug node/ip-10-0-155-125.us-west-1.compute.internal
```

   b. **/host** をデバッグシェル内のルートディレクトリーとして設定します。

```
sh-4.4# chroot /host
```

   c. **rpm-ostree status** コマンドを実行して、カスタムレイヤーイメージが使用されていることを確認します。

```
sh-4.4# sudo rpm-ostree status
```

**出力例**

```
State: idle
Deployments:
* ostree-unverified-registry:podman pull quay.io/openshift-release-dev/ocp-
release@sha256:e2044c3cfebe0ff3a99fc207ac5efe6e07878ad59fd4ad5e41f88cb016dacd
```

```
73
                Digest:
sha256:e2044c3cfebe0ff3a99fc207ac5efe6e07878ad59fd4ad5e41f88cb016dacd73
```

## 7.6. RHCOS カスタムレイヤーイメージによる更新

Red Hat Enterprise Linux CoreOS (RHCOS) イメージのレイヤー化を設定すると、OpenShift Container Platform は、カスタムレイヤーイメージを使用するノードプールを自動的に更新しなくなります。必要に応じてノードを手動で更新する必要があります。

カスタムレイヤーイメージを使用するノードを更新するには、次の一般的な手順に従います。

1. カスタムレイヤーイメージを使用するノードを除き、クラスターはバージョン xyz+1 に自動的にアップグレードされます。

2. その後、更新された OpenShift Container Platform イメージと以前に適用した RPM を参照する新しい Containerfile を作成できます。

3. 更新されたカスタムレイヤーイメージを指す新しいマシン設定を作成します。

カスタムレイヤーイメージでノードを更新する必要はありません。ただし、そのノードが現在の OpenShift Container Platform バージョンから大幅に遅れると、予期しない結果が生じる可能性があります。

# 第8章 MACHINE CONFIG DAEMON のメトリクスの概要

Machine Config Daemon は Machine Config Operator の一部です。これはクラスター内のすべてのノードで実行されます。Machine Config Daemon は、各ノードの設定変更および更新を管理します。

## 8.1. MACHINE CONFIG DAEMON のメトリクスについて

OpenShift Container Platform 4.3 以降、Machine Config Daemon はメトリクスのセットを提供します。これらのメトリクスには、Prometheus クラスターモニタリングスタックを使用してアクセスできます。

以下の表では、これらのメトリクスのセットについて説明しています。一部のエントリーには、特定のログを取得するためのコマンドが含まれています。ただし、最も包括的なログのセットは、**oc adm must-gather** コマンドを使用して入手できます。

**注記**

Name 列と Description 列に * が付いているメトリクスは、パフォーマンスの問題を引き起こす可能性のある重大なエラーを表します。このような問題により、更新およびアップグレードが続行されなくなる可能性があります。

**表8.1 MCO メトリクス**

| 名前 | フォーマット | 説明 | 備考 |
|---|---|---|---|
| **mcd_host_os_and_version** | **[]string{"os", "version"}** | RHCOS や RHEL など、MCD が実行されている OS を示します。RHCOS の場合、バージョンは指定されます。 | |
| **mcd_drain_err*** | | ドレイン (解放) の失敗時に受信されるエラーをログに記録します。* | ドレイン (解放) が成功するには、複数回試行する必要がある可能性があり、ターミナルでは、ドレイン (解放) に失敗すると更新を続行できなくなります。ドレイン (解放) にかかる時間を示す **drain_time** メトリクスはトラブルシューティングに役立つ可能性があります。<br><br>詳細な調査を実行するには、以下を実行してログを表示します。<br><br>**$ oc logs -f -n openshift-machine-config-operator machine-config-daemon-<hash> -c machine-config-daemon** |

| 名前 | フォーマット | 説明 | 備考 |
| --- | --- | --- | --- |
| **mcd_pivot_err*** | **[]string{"err", "node", "pivot_target"}** | ピボットで発生するログ。* | ピボットのエラーにより、OS のアップグレードを続行できなくなる可能性があります。<br><br>さらに調査するには、次のコマンドを実行して **machine-config-daemon** コンテナーからのログを確認します。<br><br>**$ oc logs -f -n openshift-machine-config-operator machine-config-daemon-<hash> -c machine-config-daemon** |
| **mcd_state** | **[]string{"state", "reason"}** | 指定ノードの Machine Config Daemon の状態。状態のオプションとして、"Done"、"Working"、および "Degraded" があります。"Degraded" の場合は、理由も含まれます。 | 詳細な調査を実行するには、以下を実行してログを表示します。<br><br>**$ oc logs -f -n openshift-machine-config-operator machine-config-daemon-<hash> -c machine-config-daemon** |
| **mcd_kubelet_state*** | | kubelet の正常性についての失敗をログに記録します。* | これは、失敗数が 0 で空になることが予想されます。失敗数が 2 を超えると、しきい値を超えたことを示すエラーが出されます。これは kubelet の正常性に関連した問題の可能性を示します。<br><br>詳細な調査を行うには、以下のコマンドを実行してノードにアクセスし、そのすべてのログを表示します。<br><br>**$ oc debug node/<node> — chroot /host journalctl -u kubelet** |

| 名前 | フォーマット | 説明 | 備考 |
| --- | --- | --- | --- |
| **mcd_reboot_err*** | **[]string{"message", "err", "node"}** | 再起動の失敗と対応するエラーをログに記録します。* | これは空になることが予想されますが、これは再起動が成功したことを示します。<br><br>詳細な調査を実行するには、以下を実行してログを表示します。<br><br>**$ oc logs -f -n openshift-machine-config-operator machine-config-daemon-<hash> -c machine-config-daemon** |
| **mcd_update_state** | **[]string{"config", "err"}** | 設定更新の成功または失敗、および対応するエラーをログに記録します。 | 予想される値は **rendered-master/rendered-worker-XXXX** です。更新に失敗すると、エラーが表示されます。<br><br>詳細な調査を実行するには、以下を実行してログを表示します。<br><br>**$ oc logs -f -n openshift-machine-config-operator machine-config-daemon-<hash> -c machine-config-daemon** |

**関連情報**

- モニタリングの概要
- クラスターに関するデータの収集