



# OpenShift Container Platform 4.10

## Windows Container Support for OpenShift

Red Hat OpenShift for Windows Containers ガイド



# OpenShift Container Platform 4.10 Windows Container Support for OpenShift

---

Red Hat OpenShift for Windows Containers ガイド

## 法律上の通知

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

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

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

All other trademarks are the property of their respective owners.

## 概要

Red Hat OpenShift for Windows Containers は、OpenShift Container Platform で Microsoft Windows Server コンテナを実行するための組み込みサポートを提供します。本書では、すべての詳細情報を提供します。

## 目次

<b>第1章 RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS の概要</b>	<b>3</b>
<b>第2章 RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS リリースノート</b>	<b>4</b>
2.1. RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS について	4
2.2. サポート	4
2.3. RED HAT WINDOWS MACHINE CONFIG OPERATOR 5.1.1 リリースノート	4
2.4. RED HAT WINDOWS MACHINE CONFIG OPERATOR 5.1.0 リリースノート	5
2.5. RED HAT WINDOWS MACHINE CONFIG OPERATOR 5.0.0 リリースノート	5
2.6. WINDOWS MACHINE CONFIG OPERATOR の前提条件	6
2.7. 既知の制限	8
<b>第3章 WINDOWS コンテナワークロードについて</b>	<b>10</b>
3.1. WINDOWS MACHINE CONFIG OPERATOR の前提条件	10
3.2. WINDOWS ワークロード管理	12
3.3. WINDOWS ノードサービス	14
3.4. 既知の制限	15
<b>第4章 WINDOWS コンテナワークロードの有効化</b>	<b>17</b>
前提条件	17
4.1. WINDOWS MACHINE CONFIG OPERATOR のインストール	17
4.2. WINDOWS MACHINE CONFIG OPERATOR のシークレットの設定	20
4.3. 関連情報	21
<b>第5章 WINDOWS マシンセットの作成</b>	<b>22</b>
5.1. AWS 上での WINDOWS マシンセットの作成	22
5.2. VSPHERE 上での WINDOWS マシンセットの作成	27
<b>第6章 WINDOWS コンテナワークロードのスケジューリング</b>	<b>38</b>
前提条件	38
6.1. WINDOWS POD の配置	38
6.2. スケジューリングメカニズムをカプセル化するための RUNTIMECLASS オブジェクトの作成	39
6.3. WINDOWS コンテナワークロードのデプロイメント例	40
6.4. マシンセットの手動によるスケーリング	41
<b>第7章 WINDOWS ノードのアップグレード</b>	<b>44</b>
7.1. WINDOWS MACHINE CONFIG OPERATOR のアップグレード	44
<b>第8章 BYOH (BRING-YOUR-OWN-HOST) WINDOWS インスタンスをノードとして使用</b>	<b>45</b>
8.1. BYOH WINDOWS インスタンスの設定	45
8.2. BYOH WINDOWS インスタンスの削除	46
<b>第9章 WINDOWS ノードの削除</b>	<b>47</b>
9.1. 特定マシンの削除	47
<b>第10章 WINDOWS コンテナワークロードの無効化</b>	<b>48</b>
10.1. WINDOWS MACHINE CONFIG OPERATOR のアンインストール	48
10.2. WINDOWS MACHINE CONFIG OPERATOR NAMESPACE の削除	48



## 第1章 RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS の概要

Red Hat OpenShift support for Windows Containers は、OpenShift Container Platform クラスターで Windows コンピュートノードを実行する機能を提供します。これは、Red Hat Windows Machine Config Operator (WMCO) を使用して Windows ノードをインストールし、管理することで実行できます。Red Hat Subscription を使用すると、OpenShift Container Platform での Windows ワークロードの実行をサポートできます。詳細は [リリースノート](#) を参照してください。

Linux と Windows の両方を含むワークロードの場合、OpenShift Container Platform を使用すると、Windows Server コンテナで実行される Windows ワークロードをデプロイすると同時に、Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) でホストされる従来の Linux ワークロードを提供できます。詳細は、[Windows コンテナワークロードのスタートガイド](#) を参照してください。

クラスターで Windows ワークロードを実行するには、WMCO が必要です。WMCO は、クラスター上で Windows ワークロードをデプロイし、管理するプロセスをオーケストレーションします。詳細は、[Windows コンテナワークロードを有効にする方法](#) を参照してください。

Windows **MachineSet** オブジェクトを作成して、インフラストラクチャー Windows マシンセットおよび関連マシンを作成し、サポートされている Windows ワークロードを新しい Windows マシンに移動できます。複数のプラットフォームで Windows **MachineSet** オブジェクトを作成できます。

Windows コンピュートノードに [Windows ワークロードをスケジュール](#) できます。

[Windows Machine Config Operator のアップグレードを実行](#) して、Windows ノードに最新の更新が適用されていることを確認できます。

特定のマシンを削除することで、[Windows ノードを削除](#) できます。

[Bring-Your-Own-Host \(BYOH\) Windows インスタンスを使用](#) して、Windows Server VM を再利用し、OpenShift Container Platform に移動できます。BYOH Windows インスタンスは、Windows サーバーがオフラインになった場合に、主要な中断を軽減するのに役立ちます。BYOH Windows インスタンスは、OpenShift Container Platform 4.8 以降のバージョンのノードとして使用できます。

次の手順を実行して、[Windows コンテナのワークロードを無効化](#) できます。

- Windows Machine Config Operator のアンインストール
- Windows Machine Config Operator namespace の削除

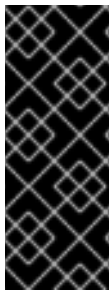
## 第2章 RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS リリースノート

### 2.1. RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS について

Red Hat OpenShift support for Windows Containers は、OpenShift Container Platform クラスターでの Windows コンピュートノードの実行を可能にします。Windows ワークロードは、Red Hat Windows Machine Config Operator (WMCO) を使用して Windows ノードをインストールし、管理することで実行できます。Windows ノードが利用可能になると、Windows コンテナワークロードを OpenShift Container Platform で実行できます。

これらのリリースノートは、OpenShift Container Platform のすべての Windows コンテナワークロード機能を提供する WMCO の開発を追跡します。

WMCO のバージョン 5.x は OpenShift Container Platform 4.10 とのみ互換性があります。



#### 重要

Microsoft が Docker を使用した Windows Server 2019 イメージの公開を停止したため、Red Hat はバージョン 6.0.0 より前の WMCO リリースの Windows Azure をサポートしなくなりました。WMCO 5.y.z 以前の場合、Windows Server 2019 イメージには Docker がプリインストールされている必要があります。WMCO 6.0.0 以降では、containerd がランタイムとして使用されます。WMCO 6.0.0 を使用する OpenShift Container Platform 4.11 にアップグレードできます。

### 2.2. サポート

Red Hat OpenShift support for Windows Containers は、オプションのインストール可能なコンポーネントとして提供されており、その使用が可能です。Windows Container Support for Red Hat OpenShift は OpenShift Container Platform サブスクリプションの一部ではありません。追加の Red Hat サブスクリプションが必要で、[対象範囲](#) および [サービスレベルアグリーメント](#) に従ってサポートされます。

Windows Container Support for Red Hat OpenShift のサポートを受けるには、この別のサブスクリプションが必要です。この追加の Red Hat サブスクリプションがないと、実稼働クラスターへの Windows コンテナワークロードのデプロイはサポートされません。[Red Hat カスタマーポータル](#) からサポートをリクエストできます。

詳細は、[Red Hat OpenShift support for Windows Containers](#) の Red Hat OpenShift Container Platform ライフサイクルポリシーのドキュメントを参照してください。

この追加の Red Hat サブスクリプションがない場合は、正式なサポートのないディストリビューションである Community Windows Machine Config Operator を使用できます。

### 2.3. RED HAT WINDOWS MACHINE CONFIG OPERATOR 5.1.1 リリースノート

WMCO のこのリリースでは、バグ修正といくつかの改善が行われています。WMCO 5.1.1 のコンポーネントは [RHBA-2023:4487](#) で利用できるようになりました。<https://errata.devel.redhat.com/advisory/101759>

#### 2.3.1. バグ修正



- 以前のバージョンでは、エンドポイントオブジェクトに必要な情報がないため、WMCO Pod は起動時に失敗していました。今回の修正により、WMCO はエンドポイントオブジェクトが必須フィールドに存在することを検証します。その結果、WMCO は無効または誤って設定されたエンドポイントオブジェクトを開始および調整できます。(OCPBUGS-5131)

## 2.3.2. 削除された機能

### 2.3.2.1. Microsoft Azure のサポートが削除される

Microsoft Azure のサポートが削除されました。Microsoft は、Docker が事前にインストールされている Azure レジストリーからイメージを削除します。これは、Microsoft Azure で WCMO 5.x を使用するための前提条件です。

## 2.4. RED HAT WINDOWS MACHINE CONFIG OPERATOR 5.1.0 リリースノート

WMCO のこのリリースでは、バグ修正といくつかの改善が行われています。WMCO 5.1.0 のコンポーネントは、[RHBA-2022:4989-01](#) で利用できるようになりました。

### 2.4.1. バグ修正

以前は、ノードの外部 IP がポインターレコード (PTR) なしで存在していた場合、Windows の Bring-Your-Own-Host (BYOH) インスタンスのリバース DNS ルックアップが失敗していました。このリリースでは、最初のノード IP アドレスに PTR レコードが存在しない場合、WMCO はリバースルックアップレコードが見つかるまで他のノードアドレスを調べます。その結果、ノードの外部 IP アドレスが PTR レコードなしで存在する場合、Windows BYOH インスタンスのリバース設定は成功します。  
([BZ#2081825](#))

### 2.4.2. 既知の問題

Microsoft Azure の **machineSets** で **publicIP** パラメーターが **false** に設定されている場合、Windows マシンセットはスケールアップできません。この問題は ([BZ#2091642](#)) で追跡されています。

### 2.4.3. 新機能および改善点

#### 2.4.3.1. Windows ノード証明書が更新される

このリリースにより、WMCO は kubelet クライアント認証局 (CA) 証明書のローテーション時に Windows ノードの証明書を更新するようになりました。

#### 2.4.3.2. Windows Server 2022 のサポート

このリリースでは、Windows Server 2022 が VMware vSphere とベアメタルをサポートするようになりました。

## 2.5. RED HAT WINDOWS MACHINE CONFIG OPERATOR 5.0.0 リリースノート

本 WMCO リリースノートは、Windows コンピュートノードを OpenShift Container Platform クラスターで実行するためのバグ修正を提供します。WMCO 5.0.0 のコンポーネントは [RHSA-2022:0577](#) でリリースされています。

- 以前のリリースでは、Windows ノード上の Windows コンテナに誤った DNS サーバー IP が割り当てられる可能性があります。これにより、DNS 解決が失敗することがありました。今回の修正では、ハードコードされたクラスターの DNS 情報が削除され、DNS サーバーの IP がコマンドラインの引数として渡されるようになりました。その結果、Windows ノード上の Windows コンテナには有効な DNS サーバー IP が割り当てられ、DNS 解決は Windows ワークロードに対して機能します。(BZ#1994859)
- 以前のリリースでは、PowerShell をデフォルトの SSH シェルとして使用する Windows VM に対して WMCO によって実行される特定のコマンドが正しく解析されませんでした。その結果、これらの VM をノードとしてクラスターに追加できませんでした。この修正により、WMCO は VM のデフォルトの SSH シェルを識別し、それに応じてコマンドを実行します。その結果、デフォルトの SSH シェルとして PowerShell を使用する VM を、ノードとしてクラスターに追加できるようになりました。(BZ#2000772)
- 以前のリリースでは、Bring-Your-Own-Host (BYOH) VM が DNS オブジェクトで指定されていた場合に、WMCO で VM がそのノードオブジェクトに適切に関連付けられませんでした。これにより、WMCO はすでに完全に設定されている VM の設定を試みました。今回の修正により、WMCO は、関連付けられたノードの検索時に VM の DNS アドレスを正しく解決します。その結果、BYOH VM は必要な場合にのみ設定されるようになりました。(BZ#2005360)
- 以前のリリースでは、**windows-exporter** メトリックエンドポイントオブジェクトに削除されたマシンへの参照が含まれていた場合に、WMCO はそれらのマシンの **Deleting** フェーズ通知イベントを無視していました。今回の修正により、マシンオブジェクトの検証がイベントフィルターリングから削除されます。その結果、マシンがまだ削除中である場合でも、**windows-exporter** メトリックエンドポイントオブジェクトが正しく更新されます。(BZ#2008601)
- 以前のリリースでは、WMCO 以外のエンティティで BYOH ノードに関連付けられた証明書署名要求 (CSR) が変更された場合に、WMCO の CSR への参照が古くなり、それを承認できませんでした。今回の修正により、更新の競合が検出された場合に、WMCO は指定されたタイムアウトまで CSR 承認を再試行します。その結果、CSR 処理は期待どおりに完了します。(BZ#2032048)

## 2.6. WINDOWS MACHINE CONFIG OPERATOR の前提条件

以下では、Windows Machine Config Operator のサポート対象のプラットフォームバージョン、Windows Server バージョン、およびネットワーク設定について詳しく説明します。対象のプラットフォームのみに関連する情報については、vSphere のドキュメントを参照してください。



### 重要

Microsoft が [Docker を使用した Windows Server 2019 イメージの公開を停止した](#) ため、Red Hat はバージョン 6.0.0 より前の WMCO リリースの Windows Azure をサポートしなくなりました。WMCO 5.y.z 以前の場合、Windows Server 2019 イメージには Docker がプリインストールされている必要があります。WMCO 6.0.0 以降では、containerd がランタイムとして使用されます。WMCO 6.0.0 を使用する OpenShift Container Platform 4.11 にアップグレードできます。

### 2.6.1. WMCO 5.1.x でサポートされるプラットフォームおよび Windows Server バージョン

以下の表は、該当するプラットフォームに基づいて WMCO 5.1.1 および 5.1.0 でサポートされている [Windows Server のバージョン](#) を示しています。一覧にない Windows Server バージョンはサポート対象外で、使用しようとするエラーが発生します。これらのエラーを防ぐには、プラットフォームに適したバージョンのみを使用してください。

プラットフォーム	サポート対象の Windows Server バージョン
Amazon Web Services (AWS)	Windows Server 2019 (version 1809)
Microsoft Azure	Windows Server 2019 (version 1809)
VMware vSphere	Windows Server 長期サービスチャネル (LTSC): Windows Server 2022 (OS ビルド <a href="#">20348.681</a> 以降)
ベアメタルまたはプロバイダーの指定なし	<ul style="list-style-type: none"> <li>Windows Server 2022 長期サービスチャネル (LTSC) OS ビルド <a href="#">20348.681</a> 以降。</li> <li>Windows Server 2019 (version 1809)</li> </ul>

### 2.6.2. WMCO 5.0.0 でサポートされるプラットフォームと Windows Server バージョン

次の表に、該当するプラットフォームに基づいて、WMCO 5.0.0 でサポートされている [Windows Server のバージョン](#) を示します。一覧にない Windows Server バージョンはサポート対象外で、使用しようするとエラーが発生します。これらのエラーを防ぐには、プラットフォームに適したバージョンのみを使用してください。

プラットフォーム	サポート対象の Windows Server バージョン
Amazon Web Services (AWS)	Windows Server 2019 (version 1809)
VMware vSphere	Windows Server 2022 長期サービスチャネル (LTSC) OS ビルド <a href="#">20348.681</a> 以降。
ベアメタルまたはプロバイダーの指定なし	Windows Server 2019 (version 1809)

### 2.6.3. サポート対象のネットワーク

サポート対象のネットワーク設定は、OVN-Kubernetes を使用したハイブリッドネットワークのみです。この機能の詳細は、以下の追加リソースを参照してください。以下の表は、プラットフォームで使用するネットワーク設定の種類と Windows Server バージョンの概要を示しています。クラスターのインストール時にネットワーク設定を指定する必要があります。OpenShift SDN ネットワークは OpenShift Container Platform クラスターのデフォルトのネットワークであることに注意してください。ただし、OpenShift SDN は WMCO ではサポートされません。

表2.1 プラットフォームネットワーキングサポート

プラットフォーム	サポート対象のネットワーク
Amazon Web Services (AWS)	OVN-Kubernetes を使用したハイブリッドネットワーク

プラットフォーム	サポート対象のネットワーク
Microsoft Azure	OVN-Kubernetes を使用したハイブリッドネットワーク
VMware vSphere	カスタム VXLAN ポートを備えた OVN-Kubernetes でのハイブリッドネットワーク
ベアメタル	OVN-Kubernetes を使用したハイブリッドネットワーク

表2.2 WMCO 5.1.0 Hybrid OVN-Kubernetes Windows Server のサポート

OVN-Kubernetes を使用したハイブリッドネットワーク	サポート対象の Windows Server バージョン
デフォルトの VXLAN ポート	Windows Server 2019 (version 1809)
カスタム VXLAN ポート	Windows Server 2022 長期サービスチャネル (LTSC) OS ビルド <a href="#">20348.681</a> 以降

表2.3 WMCO 5.0.0 Hybrid OVN-Kubernetes Windows Server のサポート

OVN-Kubernetes を使用したハイブリッドネットワーク	サポート対象の Windows Server バージョン
デフォルトの VXLAN ポート	Windows Server 2019 (version 1809)
カスタム VXLAN ポート	Windows Server 2022 長期サービスチャネル (LTSC) OS ビルド <a href="#">20348.681</a> 以降

## 2.7. 既知の制限

WMCO によって管理される Windows ノード (Windows ノード) を使用する場合は、次の制限に注意してください。

- 以下の OpenShift Container Platform 機能は、Windows ノードではサポートされていません。
  - イメージビルド
  - OpenShift Pipeline
  - OpenShift Service Mesh
  - ユーザー定義プロジェクトの OpenShift モニターリング
  - OpenShift Serverless
  - Horizontal Pod Autoscaling

- Vertical Pod Autoscaling
- 次の Red Hat 機能は、Windows ノードではサポートされていません。
  - [Red Hat のコスト管理](#)
  - [Red Hat OpenShift Local](#)
- Windows ノードは、プライベートレジストリーからのコンテナイメージのプルをサポートしていません。パブリックレジストリーのイメージを使用するか、イメージを事前にプルすることができます。
- Windows ノードは、デプロイ設定を使用して作成されたワークロードをサポートしていません。デプロイまたはその他の方法を使用して、ワークロードをデプロイできます。
- Windows ノードは、クラスター全体のプロキシを使用するクラスターではサポートされていません。これは、WMCO がワークロードのプロキシ接続を介してトラフィックをルーティングできないためです。
- Windows ノードは、切断された環境にあるクラスターではサポートされていません。
- Red Hat OpenShift による Windows コンテナのサポートでは、トランクポートを介したクラスターへの Windows ノードの追加はサポートされていません。Windows ノードを追加するためにサポートされている唯一のネットワーク設定は、VLAN のトラフィックを伝送するアクセスポート経由です。
- Windows コンテナの Red Hat OpenShift サポートは、すべてのクラウドプロバイダーのツリー内ストレージドライバーのみをサポートします。
- Kubernetes は、次の [ノード機能の制限](#) を特定しました。
  - Windows コンテナでは Huge Page はサポートされていません。
  - 特権コンテナは、Windows コンテナではサポートされていません。
  - Pod 終了の猶予期間では、containerd コンテナランタイムを Windows ノードにインストールする必要があります。
- Kubernetes は、[いくつかの API 互換性の問題](#) を特定しました。

## 第3章 WINDOWS コンテナワークロードについて

Red Hat OpenShift support for Windows Containers は、OpenShift Container Platform で Microsoft Windows Server コンテナを実行するための組み込みサポートを提供します。Linux と Windows ワークロードの組み合わせを使用して異種環境を管理する場合、OpenShift Container Platform では、Windows Server コンテナで実行されている Windows ワークロードをデプロイできますが、Red Hat Enterprise Linux CoreOS (RHCOS) または Red Hat Enterprise Linux (RHEL) でホストされる従来の Linux ワークロードも提供できます。



### 注記

Windows ノードを持つクラスターのマルチテナンシーはサポートされません。マルチテナントの悪意のある使用により、すべての Kubernetes 環境でのセキュリティ上のリスクが導入されます。[Pod セキュリティポリシー](#)、またはノードのより詳細なロールベースアクセス制御 (RBAC) などの追加のセキュリティ機能により、悪用がより困難になります。ただし、悪意のあるマルチテナントワークロードの実行を選択する場合、ハイパーバイザーは使用する必要のある唯一のセキュリティオプションになります。Kubernetes のセキュリティドメインは、個別のノードではなく、クラスター全体に対応します。これらのタイプの悪意のあるマルチテナントワークロードには、物理的に分離されたクラスターを使用する必要があります。

Windows Server コンテナは共有カーネルを使用してリソース分離を行います。悪意のあるマルチテナンシーのシナリオで使用することは意図されていません。悪意のあるマルチテナンシーを使用するシナリオの場合、テナントを確実に分離するために Hyper-V 分離コンテナを使用する必要があります。

### 3.1. WINDOWS MACHINE CONFIG OPERATOR の前提条件

以下では、Windows Machine Config Operator のサポート対象のプラットフォームバージョン、Windows Server バージョン、およびネットワーク設定について詳しく説明します。対象のプラットフォームのみに関連する情報については、vSphere のドキュメントを参照してください。



### 重要

Microsoft が [Docker を使用した Windows Server 2019 イメージの公開を停止した](#) ため、Red Hat はバージョン 6.0.0 より前の WMCO リリースの Windows Azure をサポートしなくなりました。WMCO 5.y.z 以前の場合、Windows Server 2019 イメージには Docker がプリインストールされている必要があります。WMCO 6.0.0 以降では、containerd がランタイムとして使用されます。WMCO 6.0.0 を使用する OpenShift Container Platform 4.11 にアップグレードできます。

#### 3.1.1. WMCO 5.1.x でサポートされるプラットフォームおよび Windows Server バージョン

以下の表は、該当するプラットフォームに基づいて WMCO 5.1.1 および 5.1.0 でサポートされている [Windows Server のバージョン](#) を示しています。一覧にない Windows Server バージョンはサポート対象外で、使用しようとするとエラーが発生します。これらのエラーを防ぐには、プラットフォームに適したバージョンのみを使用してください。

プラットフォーム	サポート対象の Windows Server バージョン
Amazon Web Services (AWS)	Windows Server 2019 (version 1809)



プラットフォーム	サポート対象の Windows Server バージョン
Microsoft Azure	Windows Server 2019 (version 1809)
VMware vSphere	Windows Server 長期サービスチャネル (LTSC): Windows Server 2022 (OS ビルド <a href="#">20348.681</a> 以降)
ベアメタルまたはプロバイダーの指定なし	<ul style="list-style-type: none"> <li>Windows Server 2022 長期サービスチャネル (LTSC) OS ビルド <a href="#">20348.681</a> 以降。</li> <li>Windows Server 2019 (version 1809)</li> </ul>

### 3.1.2. WMCO 5.0.0 でサポートされるプラットフォームと Windows Server バージョン

次の表に、該当するプラットフォームに基づいて、WMCO 5.0.0 でサポートされている [Windows Server のバージョン](#) を示します。一覧にない Windows Server バージョンはサポート対象外で、使用しようするとエラーが発生します。これらのエラーを防ぐには、プラットフォームに適したバージョンのみを使用してください。

プラットフォーム	サポート対象の Windows Server バージョン
Amazon Web Services (AWS)	Windows Server 2019 (version 1809)
VMware vSphere	Windows Server 2022 長期サービスチャネル (LTSC) OS ビルド <a href="#">20348.681</a> 以降。
ベアメタルまたはプロバイダーの指定なし	Windows Server 2019 (version 1809)

### 3.1.3. サポート対象のネットワーク

サポート対象のネットワーク設定は、OVN-Kubernetes を使用したハイブリッドネットワークのみです。この機能の詳細は、以下の追加リソースを参照してください。以下の表は、プラットフォームで使用するネットワーク設定の種類と Windows Server バージョンの概要を示しています。クラスタのインストール時にネットワーク設定を指定する必要があります。OpenShift SDN ネットワークは OpenShift Container Platform クラスタのデフォルトのネットワークであることに注意してください。ただし、OpenShift SDN は WMCO ではサポートされません。

表3.1 プラットフォームネットワーキングサポート

プラットフォーム	サポート対象のネットワーク
Amazon Web Services (AWS)	OVN-Kubernetes を使用したハイブリッドネットワーク
Microsoft Azure	OVN-Kubernetes を使用したハイブリッドネットワーク

プラットフォーム	サポート対象のネットワーク
VMware vSphere	カスタム VXLAN ポートを備えた OVN-Kubernetes でのハイブリッドネットワーク
ベアメタル	OVN-Kubernetes を使用したハイブリッドネットワーク

表3.2 WMCO 5.1.0 Hybrid OVN-Kubernetes Windows Server のサポート

OVN-Kubernetes を使用したハイブリッドネットワーク	サポート対象の Windows Server バージョン
デフォルトの VXLAN ポート	Windows Server 2019 (version 1809)
カスタム VXLAN ポート	Windows Server 2022 長期サービスチャネル (LTSC)OS ビルド <a href="#">20348.681</a> 以降

表3.3 WMCO 5.0.0 Hybrid OVN-Kubernetes Windows Server のサポート

OVN-Kubernetes を使用したハイブリッドネットワーク	サポート対象の Windows Server バージョン
デフォルトの VXLAN ポート	Windows Server 2019 (version 1809)
カスタム VXLAN ポート	Windows Server 2022 長期サービスチャネル (LTSC)OS ビルド <a href="#">20348.681</a> 以降

## 関連情報

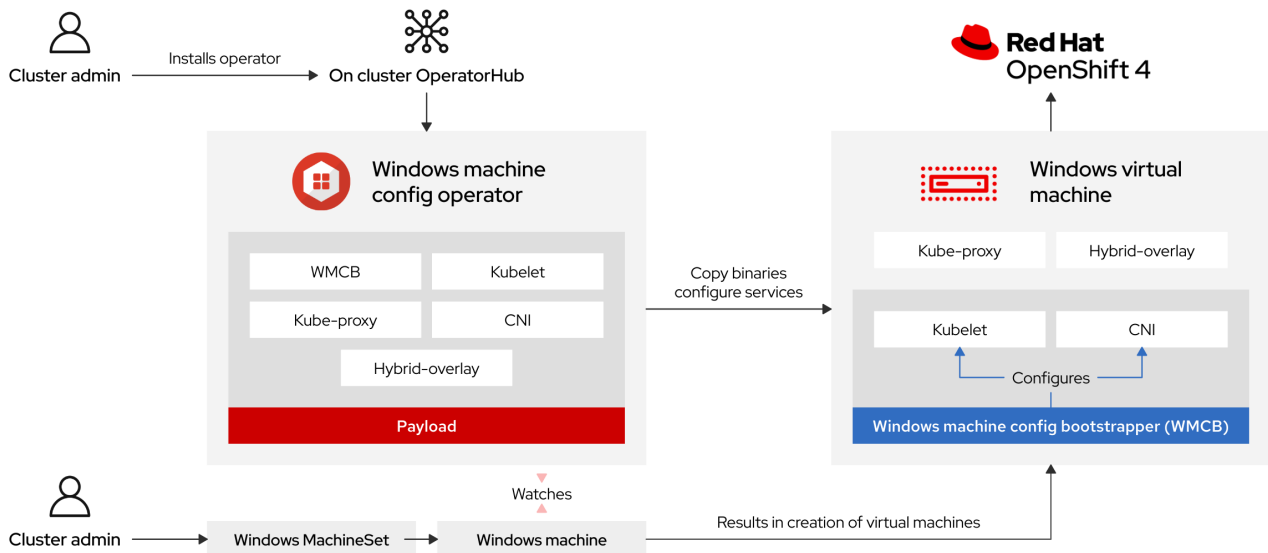
- [OVN-Kubernetes を使用したハイブリッドネットワークの設定](#) を参照してください。

## 3.2. WINDOWS ワークロード管理

クラスターで Windows ワークロードを実行するには、まず Windows Machine Config Operator (WMCO) をインストールする必要があります。WMCO は Linux ベースのコントロールプレーンおよびコンピュートノードで実行される Linux ベースの Operator です。WMCO は、クラスター上で Windows ワークロードをデプロイし、管理するプロセスをオーケストレーションします。



図3.1 WMCO の設計



Windows ワークロードをデプロイする前に、Windows コンピュートノードを作成し、これをクラスターに参加させる必要があります。Windows ノードは、クラスター内の Windows ワークロードをホストし、他の Linux ベースのコンピュートノードと共に実行できます。Windows Server コンピュートマシンをホストする Windows マシンセットを作成して、Windows コンピュートノードを作成することができます。Docker 形式のコンテナランタイムアドオンが有効にされた Windows OS イメージを指定する Windows 固有のラベルをマシンセットに適用する必要があります。

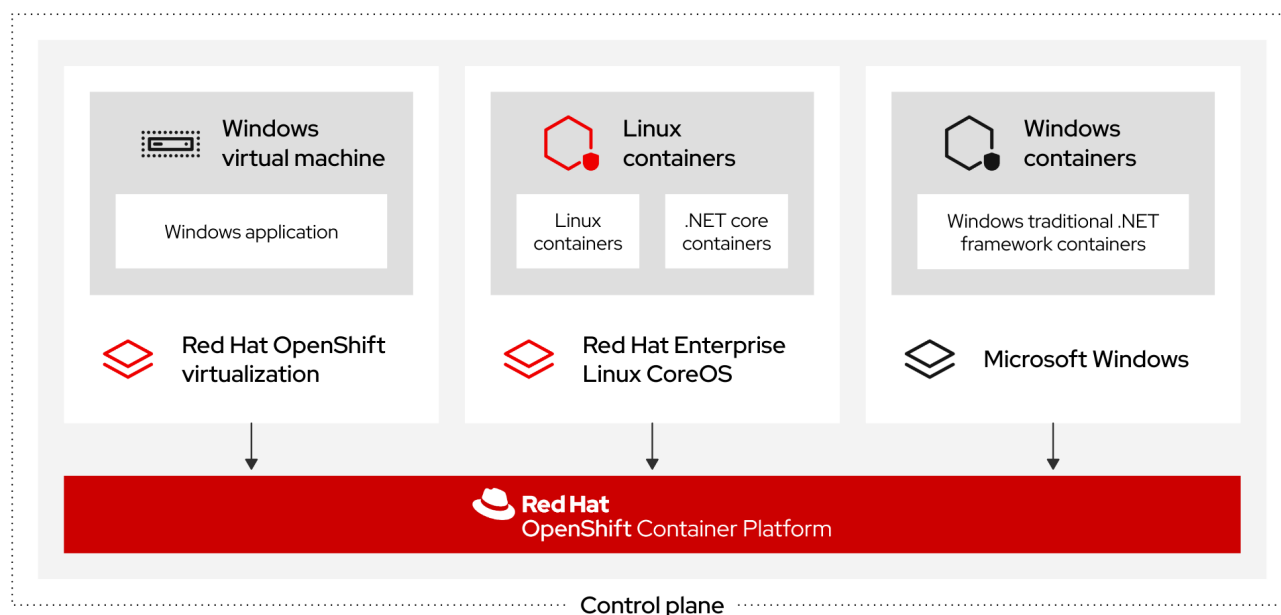


### 重要

現時点で、Docker 形式のコンテナランタイムは Windows ノードで使用されます。Kubernetes では、コンテナランタイムとしての Docker を非推奨としています。詳細は、Kubernetes ドキュメントの [Docker の非推奨](#) について参照してください。Containerd は、今後の Kubernetes リリースで Windows ノードについてサポートされる新しいコンテナランタイムになります。

WMCO は Windows ラベルの付いたマシンを監視します。Windows マシンセットを検出し、そのそれぞれのマシンがプロビジョニングされると、WMCO は基礎となる Windows 仮想マシン (VM) を設定し、それがコンピュートノードとしてクラスターに参加できるようにします。

図3.2 Windows および Linux ワークロードの組み合わせ



WMCO は、Windows インスタンスとの対話に使用されるプライベートキーが含まれる namespace に事前に決定されたシークレットがあることを想定します。WMCO は起動時にこのシークレットの有無を確認し、作成した Windows **MachineSet** オブジェクトで参照する必要があるユーザーデータのシークレットを作成します。次に WMCO は、プライベートキーに対応するパブリックキーでユーザーデータのシークレットを設定します。このデータが適用されると、クラスターは SSH 接続を使用して Windows 仮想マシンに接続できます。

クラスターが Windows 仮想マシンとの接続を確立した後に、Linux ベースのノードの場合と同様の手法を使用して Windows ノードを管理できます。



### 注記

OpenShift Container Platform Web コンソールは、Linux ノードで利用可能な機能と同じモニタリング機能のほとんどを Windows ノードについて提供します。ただし、現時点で Windows ノードで実行されている Pod のワークロードグラフを監視する機能は利用できません。

Windows ワークロードの Windows ノードへのスケジュールは、ティント、容認、ノードセクターなどの一般的な Pod スケジュールの手法を使用して実行できますが、**RuntimeClass** オブジェクトを使用して Windows ワークロードを Linux ワークロードおよび他の Windows 版のワークロードから区別できます。

## 3.3. WINDOWS ノードサービス

以下の Windows 固有のサービスは、各 Windows ノードにインストールされます。

サービス	説明
kubelet	Windows ノードを登録し、そのステータスを管理します。
Container Network Interface (CNI) プラグイン	Windows ノードの <a href="#">ネットワーク</a> を公開します。

サービス	説明
Windows Machine Config Bootstrapper (WMCB)	kubelet および CNI プラグインを設定します。
<a href="#">Windows Exporter</a>	Windows ノードから Prometheus メトリクスをエクスポートします。
<a href="#">Kubernetes Cloud Controller Manager (CCM)</a>	基礎となる Azure クラウドプラットフォームと対話します。
hybrid-overlay	OpenShift Container Platform <a href="#">Host Network Service (HNS)</a> を作成します。
kube-proxy	外部との通信を許可するノードでネットワークルールを維持します。

### 3.4. 既知の制限

WMCO によって管理される Windows ノード (Windows ノード) を使用する場合は、次の制限に注意してください。

- 以下の OpenShift Container Platform 機能は、Windows ノードではサポートされていません。
  - イメージビルド
  - OpenShift Pipeline
  - OpenShift Service Mesh
  - ユーザー定義プロジェクトの OpenShift モニターリング
  - OpenShift Serverless
  - Horizontal Pod Autoscaling
  - Vertical Pod Autoscaling
- 次の Red Hat 機能は、Windows ノードではサポートされていません。
  - [Red Hat のコスト管理](#)
  - [Red Hat OpenShift Local](#)
- Windows ノードは、プライベートレジストリーからのコンテナイメージのプルをサポートしていません。パブリックレジストリーのイメージを使用するか、イメージを事前にプルすることができます。
- Windows ノードは、デプロイ設定を使用して作成されたワークロードをサポートしていません。デプロイまたはその他の方法を使用して、ワークロードをデプロイできます。
- Windows ノードは、クラスター全体のプロキシを使用するクラスターではサポートされていません。これは、WMCO がワークロードのプロキシ接続を介してトラフィックをルーティングできないためです。

- Windows ノードは、切断された環境にあるクラスターではサポートされていません。
- Red Hat OpenShift による Windows コンテナのサポートでは、トランクポートを介したクラスターへの Windows ノードの追加はサポートされていません。Windows ノードを追加するためにサポートされている唯一のネットワーク設定は、VLAN のトラフィックを伝送するアクセスポート経由です。
- Windows コンテナの Red Hat OpenShift サポートは、すべてのクラウドプロバイダーのツリー内ストレージドライバーのみをサポートします。
- Kubernetes は、次の [ノード機能の制限](#) を特定しました。
  - Windows コンテナでは Huge Page はサポートされていません。
  - 特権コンテナは、Windows コンテナではサポートされていません。
  - Pod 終了の猶予期間では、containerd コンテナランタイムを Windows ノードにインストールする必要があります。
- Kubernetes は、[いくつかの API 互換性の問題](#) を特定しました。

## 第4章 WINDOWS コンテナワークロードの有効化

Windows ワークロードをクラスターに追加する前に、OpenShift Container Platform OperatorHub で利用可能な Windows Machine Config Operator (WMCO) をインストールする必要があります。WMCO は、クラスター上で Windows ワークロードをデプロイし、管理するプロセスをオーケストレーションします。



### 注記

デュアル NIC は、WMCO が管理する Windows インスタンスではサポートされていません。

### 前提条件

- **cluster-admin** パーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- installer-provisioned infrastructure を使用してクラスターをインストールしたか、または **install-config.yaml** ファイルに **platform: none** フィールドを設定した user-provisioned infrastructure を使用してインストールした場合。
- クラスターに OVN-Kubernetes を使用してハイブリッドネットワークを設定している。これは、クラスターのインストール時に完了する必要があります。詳細は、[ハイブリッドネットワークの設定](#) を参照してください。
- OpenShift Container Platform クラスター (バージョン 4.6.8 以降) を実行している。



### 注記

WMCO はワークロードのプロキシ接続を介してトラフィックをルーティングできないため、[クラスター全体のプロキシ](#) を使用するクラスターではサポートされません。

### 関連情報

- Windows Machine Config Operator の全前提条件については、[Windows コンテナワークロードについて](#) を参照してください。

## 4.1. WINDOWS MACHINE CONFIG OPERATOR のインストール

Web コンソールまたは OpenShift CLI (**oc**) のいずれかを使用して Windows Machine Config Operator をインストールできます。

### 4.1.1. Web コンソールを使用した Windows Machine Config Operator のインストール

OpenShift Container Platform Web コンソールを使って Windows Machine Config Operator (WMCO) をインストールすることができます。



### 注記

デュアル NIC は、WMCO が管理する Windows インスタンスではサポートされていません。

## 手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブから、**Operators → OperatorHub** ページに移動します。
2. **Filter by keyword** ボックスを使用して、カタログで **Windows Machine Config Operator** を検索します。**Windows Machine Config Operator** タイルをクリックします。
3. Operator についての情報を確認してから、**Install** をクリックします。
4. **Install Operator** ページで以下を行います。
  - a. **Update Channel** として **stable** チャンネルを選択します。**stable** チャンネルを使用すると、WMCO の最新の安定したリリースをインストールできます。
  - b. WMCO は単一の namespace でのみ利用できるようにするため、**インストールモード** は事前に設定されます。
  - c. WMCO の **Installed Namespace** を選択します。デフォルトの Operator の推奨される namespace は **openshift-windows-machine-config-operator** です。
  - d. **Enable Operator recommended cluster monitoring on the Namespace** チェックボックスをクリックし、WMCO についてクラスターのモニタリングを有効にします。
  - e. **Approval Strategy** を選択します。
    - **Automatic** ストラテジーにより、Operator Lifecycle Manager (OLM) は新規バージョンが利用可能になると Operator を自動的に更新できます。
    - **Manual** ストラテジーには、Operator の更新を承認するための適切な認証情報を持つユーザーが必要です。
1. **Install** をクリックします。WMCO が **Installed Operators** ページに一覧表示されます。



### 注記

WMCO は、**openshift-windows-machine-config-operator** などのように、定義した namespace に自動的にインストールされます。

2. **Status** に **Succeeded** が表示されていることを検証し、WMCO のインストールが正常に行われたことを確認します。

## 4.1.2. CLI を使用した Windows Machine Config Operator のインストール

OpenShift CLI (**oc**) を使用して、Windows Machine Config Operator (WMCO) をインストールできます。



### 注記

デュアル NIC は、WMCO が管理する Windows インスタンスではサポートされていません。

## 手順

1. WMCO の namespace を作成します。

- a. WMCO の **Namespace** オブジェクト YAML ファイルを作成します。例: **wmco-namespace.yaml**

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-windows-machine-config-operator ❶
  labels:
    openshift.io/cluster-monitoring: "true" ❷
```

- ❶ WMCO を **openshift-windows-machine-config-operator** namespace にデプロイすることが推奨されます。
- ❷ このラベルは、WMCO についてクラスターモニタリングを有効にするために必要です。

- b. namespace を作成します。

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

以下に例を示します。

```
$ oc create -f wmco-namespace.yaml
```

2. WMCO の Operator グループを作成します。

- a. **OperatorGroup** オブジェクト YAML ファイルを作成します。例: **wmco-og.yaml**

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: windows-machine-config-operator
  namespace: openshift-windows-machine-config-operator
spec:
  targetNamespaces:
    - openshift-windows-machine-config-operator
```

- b. Operator グループを作成します。

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

以下に例を示します。

```
$ oc create -f wmco-og.yaml
```

3. namespace を WMCO にサブスクライブします。

- a. **Subscription** オブジェクト YAML ファイルを作成します。例: **wmco-sub.yaml**

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: windows-machine-config-operator
```

```

namespace: openshift-windows-machine-config-operator
spec:
  channel: "stable" ❶
  installPlanApproval: "Automatic" ❷
  name: "windows-machine-config-operator"
  source: "redhat-operators" ❸
  sourceNamespace: "openshift-marketplace" ❹

```

- ❶ **stable** をチャンネルとして指定します。
- ❷ 承認ストラテジーを設定します。 **Automatic** または **Manual** を設定できます。
- ❸ **windows-machine-config-operator** パッケージマニフェストが含まれる、**redhat-operators** カタログソースを指定します。OpenShift Container Platform が、非接続クラスターとも呼ばれる制限されたネットワークにインストールされている場合、Operator Lifecycle Manager (OLM) の設定時に作成した **CatalogSource** オブジェクトの名前を指定します。
- ❹ カタログソースの namespace。デフォルトの OperatorHub カタログソースには **openshift-marketplace** を使用します。

b. サブスクリプションを作成します。

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

以下に例を示します。

```
$ oc create -f wmco-sub.yaml
```

WMCO が **openshift-windows-machine-config-operator** にインストールされるようになりました。

4. WMCO インストールを確認します。

```
$ oc get csv -n openshift-windows-machine-config-operator
```

#### 出力例

```

NAME                                DISPLAY                                VERSION  REPLACES  PHASE
windows-machine-config-operator.2.0.0  Windows Machine Config Operator  2.0.0
Succeeded

```

## 4.2. WINDOWS MACHINE CONFIG OPERATOR のシークレットの設定

Windows Machine Config Operator (WMCO) を実行するには、プライベートキーを含む WMCO namespace でシークレットを作成する必要があります。これは、WMCO が Windows 仮想マシン (VM) と通信できるようにするために必要です。

#### 前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。



- RSA キーが含まれる PEM でエンコードされたファイルを作成します。

#### 手順

- Windows 仮想マシンへのアクセスに必要なシークレットを定義します。

```
$ oc create secret generic cloud-private-key --from-file=private-  
key.pem=${HOME}/.ssh/<key> \  
-n openshift-windows-machine-config-operator ❶
```

- ❶ **openshift-windows-machine-config-operator** などの、WMCO namespace のプライベートキーを作成する必要があります。

クラスターのインストール時に使用されるものとは異なるプライベートキーを使用することが推奨されます。

### 4.3. 関連情報

- [クラスターノードの SSH アクセス用のキーペアの生成](#)
- [Operator のクラスターへの追加](#)

## 第5章 WINDOWS マシンセットの作成

### 5.1. AWS 上での WINDOWS マシンセットの作成

Amazon Web Services (AWS) で OpenShift Container Platform クラスターの特定の機能を果たすように **MachineSet** オブジェクトを作成することができます。たとえば、インフラストラクチャー Windows マシンセットおよび関連マシンを作成して、サポートする Windows ワークロードを新規の Windows マシンに移動できます。

#### 前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Docker フォーマットのコンテナランタイムアドオンが有効な状態で、サポートされている Windows Server をオペレーティングシステムのイメージとして使用している。  
次の **aws** コマンドを使用して、有効な AMI イメージを照会します。

```
$ aws ec2 describe-images --region <aws region name> --filters
"Name=name,Values=Windows_Server-2019*English*Full*Containers*" "Name=is-
public,Values=true" --query "reverse(sort_by(Images, &CreationDate))[*].{name: Name, id:
ImageId}" --output table
```



#### 重要

現時点で、Docker 形式のコンテナランタイムは Windows ノードで使用されます。Kubernetes では、コンテナランタイムとしての Docker を非推奨としています。詳細は、Kubernetes ドキュメントの [Docker の非推奨](#) について参照してください。Containerd は、今後の Kubernetes リリースで Windows ノードについてサポートされる新しいコンテナランタイムになります。

#### 5.1.1. マシン API の概要

マシン API は、アップストリームのクラスター API プロジェクトおよびカスタム OpenShift Container Platform リソースに基づく重要なリソースの組み合わせです。

OpenShift Container Platform 4.10 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.10 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

#### Machines

ノードのホストを記述する基本的なユニットです。マシンには、複数の異なるクラウドプラットフォーム用に提供されるコンピュートノードのタイプを記述する **providerSpec** 仕様があります。たとえば、Amazon Web Services (AWS) 上のワーカーノードのマシンタイプは特定のマシンタイプおよび必要なメタデータを定義する場合があります。

#### マシンセット

**MachineSet** リソースはマシンのグループです。マシンセットとマシンの関係は、レプリカセットと Pod の関係と同様です。マシンを追加する必要がある場合や、マシンの数を縮小したりする必要がある場合、コンピューティングのニーズに応じてマシンセットの **replicas** フィールドを変更します。

**警告**

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

**Machine Autoscaler**

**MachineAutoscaler** リソースは、クラウド内のコンピューティングマシンを自動的にスケーリングします。指定したコンピューティングマシンセット内のノードの最小および最大スケーリング境界を設定でき Machine Autoscaler はそのノード範囲を維持します。

**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

**Cluster Autoscaler**

このリソースはアップストリームの Cluster Autoscaler プロジェクトに基づいています。OpenShift Container Platform の実装では、これはマシンセット API を拡張することによってクラスター API に統合されます。コア、ノード、メモリー、および GPU などのリソースのクラスター全体でのスケーリング制限を設定できます。優先順位を設定することにより、重要度の低い Pod のために新規ノードがオンラインにならないようにクラスターで Pod の優先順位付けを実行できます。また、スケーリングポリシーを設定してノードをスケールダウンせずにスケールアップできるようにすることもできます。

**マシンのヘルスチェック**

**MachineHealthCheck** リソースはマシンの正常でない状態を検知し、マシンを削除し、サポートされているプラットフォームでは新規マシンを作成します。

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれのマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。複数のアベイラビリティゾーンを持たないグローバル Azure リージョンでは、アベイラビリティセットを使用して高可用性を確保できます。Autoscaler はクラスターの有効期間中にベストエフォートでバランシングを提供します。

**5.1.2. AWS の Windows MachineSet オブジェクトのサンプル YAML**

このサンプル YAML は、Windows Machine Config Operator (WMCO) が応答する Amazon Web Services (AWS) で実行される Windows **MachineSet** オブジェクトを定義します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-windows-worker-<zone> 2
  namespace: openshift-machine-api
```

```

spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-windows-worker-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-windows-worker-<zone> 6
        machine.openshift.io/os-id: Windows 7
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/worker: "" 8
      providerSpec:
        value:
          ami:
            id: <windows_container_ami> 9
          apiVersion: awsproviderconfig.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
          credentialsSecret:
            name: aws-cloud-credentials
          deviceIndex: 0
          iamInstanceProfile:
            id: <infrastructure_id>-worker-profile 10
          instanceType: m5a.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: <zone> 11
            region: <region> 12
          securityGroups:
            - filters:
                - name: tag:Name
                  values:
                    - <infrastructure_id>-worker-sg 13
          subnet:
            filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-private-<zone> 14
          tags:
            - name: kubernetes.io/cluster/<infrastructure_id> 15
              value: owned
          userDataSecret:
            name: windows-user-data 16
            namespace: openshift-machine-api

```

- 1 3 5 10 13 14 15 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 4 6 インフラストラクチャー ID、ワーカーラベル、およびゾーンを指定します。

- 7 Windows マシンとしてマシンセットを設定します。

- 8 Windows ノードをコンピュータマシンとして設定します。

- 9 コンテナランタイムがインストールされている Windows イメージの AMI ID を指定します。Windows Server 2019 を使用する必要があります。

- 11 **us-east-1a** などの AWS ゾーンを指定します。

- 12 **us-east-1** などの AWS リージョンを指定します。

- 16 最初の Windows マシンの設定時に WMCO によって作成されます。その後、後続のすべてのマシンセットで **windows-user-data** を消費できるようになります。

### 5.1.3. マシンセットの作成

インストールプログラムによって作成されるコンピュータセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

#### 前提条件

- OpenShift Container Platform クラスターをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに **<file\_name>.yaml** という名前を付けます。  
**<clusterID>** および **<role>** パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスターから既存のコンピュータマシンセットを確認できます。
  - a. クラスター内のコンピュータマシンセットをリスト表示するには、次のコマンドを実行します。

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

#### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m

```

agl030519-vplxk-worker-us-east-1b 1      1      1      1      55m
agl030519-vplxk-worker-us-east-1c 1      1      1      1      55m
agl030519-vplxk-worker-us-east-1d 0      0                      55m
agl030519-vplxk-worker-us-east-1e 0      0                      55m
agl030519-vplxk-worker-us-east-1f 0      0                      55m

```

- b. 特定のコンピュータマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

```

$ oc get machineset <machineset_name> \
  -n openshift-machine-api -o yaml

```

## 出力例

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
      ...

```

**1** クラスタインフラストラクチャー ID。

**2** デフォルトのノードラベル。



### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュータマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

**3** コンピュータマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュータマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

## 検証

- 次のコマンドを実行して、コンピュータマシンセットのリストを表示します。

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

## 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-windows-worker-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

## 5.1.4. 関連情報

- [マシン管理の概要](#)

## 5.2. VSPHERE 上での WINDOWS マシンセットの作成

VMware vSphere 上の OpenShift Container Platform クラスターで特定の機能を果たすように Windows **MachineSet** オブジェクトを作成できます。たとえば、インフラストラクチャー Windows マシンセットおよび関連マシンを作成して、サポートする Windows ワークロードを新規の Windows マシンに移動できます。

### 前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Docker フォーマットのコンテナランタイムアドオンが有効な状態で、サポートされている Windows Server をオペレーティングシステムのイメージとして使用している。



### 重要

現時点で、Docker 形式のコンテナランタイムは Windows ノードで使用されます。Kubernetes では、コンテナランタイムとしての Docker を非推奨としています。詳細は、Kubernetes ドキュメントの [Docker の非推奨](#) について参照してください。Containerd は、今後の Kubernetes リリースで Windows ノードについてサポートされる新しいコンテナランタイムになります。

### 5.2.1. マシン API の概要

マシン API は、アップストリームのクラスター API プロジェクトおよびカスタム OpenShift Container Platform リソースに基づく重要なリソースの組み合わせです。



OpenShift Container Platform 4.10 クラスターの場合、マシン API はクラスターインストールの終了後にすべてのノードホストのプロビジョニングの管理アクションを実行します。このシステムにより、OpenShift Container Platform 4.10 はパブリックまたはプライベートのクラウドインフラストラクチャーに加えて弾力性があり、動的なプロビジョニング方法を提供します。

以下の 2 つのリソースは重要なリソースになります。

## Machines

ノードのホストを記述する基本的なユニットです。マシンには、複数の異なるクラウドプラットフォーム用に提供されるコンピュートノードのタイプを記述する **providerSpec** 仕様があります。たとえば、Amazon Web Services (AWS) 上のワーカーノードのマシンタイプは特定のマシンタイプおよび必要なメタデータを定義する場合があります。

## マシンセット

**MachineSet** リソースはマシンのグループです。マシンセットとマシンの関係は、レプリカセットと Pod の関係と同様です。マシンを追加する必要がある場合や、マシンの数を縮小したりする必要がある場合、コンピューティングのニーズに応じてマシンセットの **replicas** フィールドを変更します。



### 警告

コントロールプレーンマシンは、マシンセットで管理することはできません。

以下のカスタムリソースは、クラスターに機能を追加します。

## Machine Autoscaler

**MachineAutoscaler** リソースは、クラウド内のコンピューティングマシンを自動的にスケーリングします。指定したコンピューティングマシンセット内のノードの最小および最大スケーリング境界を設定でき Machine Autoscaler はそのノード範囲を維持します。

**MachineAutoscaler** オブジェクトは **ClusterAutoscaler** オブジェクトの設定後に有効になります。**ClusterAutoscaler** および **MachineAutoscaler** リソースは、どちらも **ClusterAutoscalerOperator** オブジェクトによって利用可能にされます。

## Cluster Autoscaler

このリソースはアップストリームの Cluster Autoscaler プロジェクトに基づいています。OpenShift Container Platform の実装では、これはマシンセット API を拡張することによってクラスター API に統合されます。コア、ノード、メモリー、および GPU などのリソースのクラスター全体でのスケーリング制限を設定できます。優先順位を設定することにより、重要度の低い Pod のために新規ノードがオンラインにならないようにクラスターで Pod の優先順位付けを実行できます。また、スケーリングポリシーを設定してノードをスケールダウンせずにスケールアップできるようにすることもできます。

## マシンのヘルスチェック

**MachineHealthCheck** リソースはマシンの正常でない状態を検知し、マシンを削除し、サポートされているプラットフォームでは新規マシンを作成します。

OpenShift Container Platform バージョン 3.11 では、クラスターでマシンのプロビジョニングが管理されないためにマルチゾーンアーキテクチャーを容易に展開することができませんでした。しかし、OpenShift Container Platform バージョン 4.1 以降、このプロセスはより簡単になりました。それぞれ



のマシンセットのスコープが単一ゾーンに設定されるため、インストールプログラムはユーザーに代わって、アベイラビリティゾーン全体にマシンセットを送信します。さらに、コンピューティングは動的に展開されるため、ゾーンに障害が発生した場合の、マシンのリバランスが必要な場合に使用するゾーンを常に確保できます。複数のアベイラビリティゾーンを持たないグローバル Azure リージョンでは、アベイラビリティセットを使用して高可用性を確保できます。Autoscaler はクラスターの有効期間中にベストエフォートでバルランシングを提供します。

## 5.2.2. Windows コンテナワークロード用の vSphere 環境の準備

vSphere Windows 仮想マシンのゴールドイメージを作成し、WMCO の内部 API サーバーとの通信を有効にして、Windows コンテナのワークロード用に vSphere 環境を準備する必要があります。

### 5.2.2.1. vSphere Windows 仮想マシンのゴールドイメージの作成

vSphere Windows 仮想マシン (VM) のゴールドイメージを作成します。

#### 前提条件

- OpenSSH サーバーで鍵ベースの認証を設定するのに使用する、秘密鍵/公開鍵ペアを作成している。プライベートキーは Windows Machine Config Operator(WMCO) namespace でも設定される必要があります。これは、WMCO が Windows 仮想マシンと通信できるようにするために必要です。詳細は、Windows Machine Config Operator のシークレットの設定のセクションを参照してください。



#### 注記

Windows 仮想マシンを作成する際には、複数のケースで [Microsoft PowerShell](#) コマンドを使用する必要があります。本書の PowerShell コマンドは、**PS C:\>** 接頭辞によって区別されます。

#### 手順

1. [Microsoft パッチ KB5012637](#) を含む Windows Server 2022 イメージを使用して、vSphere クラウドに新しい仮想マシンを作成します。



#### 重要

VM の仮想ハードウェアバージョンは、OpenShift Container Platform のインフラストラクチャー要件を満たしている必要があります。詳細については、OpenShift Container Platform ドキュメントの VMware vSphere インフラストラクチャーの要件セクションを参照してください。また、[仮想マシンのハードウェアバージョン](#) に関する VMware のドキュメントを参照することもできます。

2. Windows 仮想マシンに VMware Tools バージョン 11.0.6 以降をインストールし、設定します。詳細は、[VMware Tools のドキュメント](#) を参照してください。
3. Windows 仮想マシンに VMware Tools をインストールした後、以下を確認します。
  - a. **C:\ProgramData\VMware\VMware Tools\tools.conf** ファイルが、以下のエントリーとともに存在します。

```
exclude-nics=
```

**tools.conf** ファイルが存在しない場合は、**exclude-nics** オプションでコメント解除した状態でこれを作成し、空の値に設定します。

このエントリーにより、ハイブリッドオーバーレイで Windows 仮想マシンで生成され、クローン作成された vNIC は無視されないようにします。

- b. Windows 仮想マシンには、vCenter で有効な IP アドレスがあります。

```
C:\> ipconfig
```

- c. VMTools Windows サービスが実行中である。

```
PS C:\> Get-Service -Name VMTools | Select Status, StartType
```

- Windows 仮想マシンに OpenSSH Server をインストールし、設定します。詳細は、Microsoft ドキュメントの [Installing OpenSSH](#) を参照してください。
- 管理ユーザーの SSH アクセスを設定します。そのためには、Microsoft のドキュメント [Administrative user](#) を参照してください。



### 重要

命令に使用されるパブリックキーは、シークレットを保持する WMCO namespace で後に作成するプライベートキーに対応している必要があります。詳細は、Windows Machine Config Operator のシークレットの設定のセクションを参照してください。

- [Microsoft ドキュメント](#) に従って、Windows 仮想マシンに **docker** コンテナランタイムをインストールします。
- コンテナログの受信接続を可能にする新しいファイアウォールルールを Windows 仮想マシンに作成する必要があります。以下の PowerShell コマンドを実行して、TCP ポート 10250 にファイアウォールルールを作成します。

```
PS C:\> New-NetFirewallRule -DisplayName "ContainerLogsPort" -LocalPort 10250 -Enabled True -Direction Inbound -Protocol TCP -Action Allow -EdgeTraversalPolicy Allow
```

- Windows 仮想マシンのクローンを作成し、再利用可能なイメージにします。詳細は、VMware ドキュメントで [既存の仮想マシンのクローンを作成](#) する方法を参照してください。
- クローン作成した Windows 仮想マシンで、[Windows Sysprep ツール](#) を実行します。

```
C:\> C:\Windows\System32\Sysprep\sysprep.exe /generalize /oobe /shutdown /unattend:  
<path_to_unattend.xml> ❶
```

- ❶ **unattend.xml** ファイルへのパスを指定します。



### 注記

Windows イメージで **sysprep** コマンドを実行することができる回数に制限があります。詳細は、Microsoft の [ドキュメント](#) を参照してください。

サンプルの **unattend.xml** が提供され、これは WMCO で必要なすべての変更を維持します。この例では変更する必要があります。直接使用することはできません。

### 例5.1 サンプル unattend.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="specialize">
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      International-Core" processorArchitecture="amd64"
      publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <InputLocale>0409:00000409</InputLocale>
      <SystemLocale>en-US</SystemLocale>
      <UILanguage>en-US</UILanguage>
      <UILanguageFallback>en-US</UILanguageFallback>
      <UserLocale>en-US</UserLocale>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      Security-SPP-UX" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS">
      <SkipAutoActivation>true</SkipAutoActivation>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      SQMApi" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS">
      <CEIPEnabled>0</CEIPEnabled>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS">
      <ComputerName>winhost</ComputerName> ❶
    </component>
  </settings>
  <settings pass="oobeSystem">
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS">
      <AutoLogon>
        <Enabled>>false</Enabled> ❷
      </AutoLogon>
      <OOBE>
        <HideEULAPage>true</HideEULAPage>
        <HideLocalAccountScreen>true</HideLocalAccountScreen>
        <HideOEMRegistrationScreen>true</HideOEMRegistrationScreen>
        <HideOnlineAccountScreens>true</HideOnlineAccountScreens>
        <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
        <NetworkLocation>Work</NetworkLocation>
        <ProtectYourPC>1</ProtectYourPC>
        <SkipMachineOOBE>true</SkipMachineOOBE>
        <SkipUserOOBE>true</SkipUserOOBE>
      </OOBE>
    </component>
  </settings>
</unattend>
```

```

<RegisteredOrganization>Organization</RegisteredOrganization>
<RegisteredOwner>Owner</RegisteredOwner>
<DisableAutoDaylightTimeSet>>false</DisableAutoDaylightTimeSet>
<TimeZone>Eastern Standard Time</TimeZone>
<UserAccounts>
  <AdministratorPassword>
    <Value>MyPassword</Value> 3
    <PlainText>>true</PlainText>
  </AdministratorPassword>
</UserAccounts>
</component>
</settings>
</unattend>

```

- 1 **Kubernetes の名前の仕様** に従いなければならない **ComputerName** を指定します。これらの仕様は、新規仮想マシンの作成時に作成されるテンプレートで実行されるゲスト OS のカスタマイズにも適用されます。
- 2 自動ログオンを無効にして、起動時に管理者権限で開いているターミナルをそのまま残すセキュリティの問題を回避します。これはデフォルト値であるため、変更しないでください。
- 3 **MyPassword** プレースホルダーを Administrator アカウントのパスワードに置き換えます。これにより、組み込みの Administrator アカウントはデフォルトで空のパスワードを持つことを防ぎます。Microsoft の [パスワードを選択するベストプラクティス](#) に従ってください。

Sysprep ツールが完了すると、Windows 仮想マシンの電源がオフになります。この仮想マシンで使用または電源は使用しないでください。

10. Windows 仮想マシンを [vCenter のテンプレート](#) に変換します。

#### 5.2.2.1.1. 関連情報

- [Windows Machine Config Operator のシークレットの設定](#)
- [VMware vSphere インフラストラクチャーの要件](#)

#### 5.2.2.2. vSphere での WMCO についての内部 API サーバーとの通信の有効化

Windows Machine Config Operator (WMCO) は Ignition 設定ファイルを内部 API サーバーエンドポイントからダウンロードします。Windows 仮想マシン (VM) が Ignition 設定ファイルをダウンロードできるように、また設定された仮想マシンが kubelet が内部 API サーバーとのみ通信できるように内部 API サーバーとの通信を有効にする必要があります。

#### 前提条件

- クラスタを vSphere にインストールしている。

#### 手順

1. 外部 API サーバーに `https://api-internal.kubelet.kubernetes.io` を参照する `api-internal`

- 外部 API サーバー URL `api.<cluster_name>.<base_domain>` を参照する `api-int.<cluster_name>.<base_domain>` の新規 DNS エントリー追加します。これには、CNAME または追加の A レコードを指定できます。



### 注記

外部 API エンドポイントは、vSphere への初期クラスターインストールの一部としてすでに作成されています。

## 5.2.3. vSphere での Windows の MachineSet オブジェクトのサンプル YAML

このサンプル YAML は、Windows Machine Config Operator (WMCO) が応答する VMware vSphere で実行される Windows **MachineSet** オブジェクトを定義します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <windows_machine_set_name> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❸
      machine.openshift.io/cluster-api-machineset: <windows_machine_set_name> ❹
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❺
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: <windows_machine_set_name> ❻
        machine.openshift.io/os-id: Windows ❼
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/worker: "" ❽
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 128 ❾
          kind: VSphereMachineProviderSpec
          memoryMiB: 16384
          network:
            devices:
              - networkName: "<vm_network_name>" ❿
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <windows_vm_template_name> ⓫
```

```

userDataSecret:
  name: windows-user-data 12
workspace:
  datacenter: <vcenter_datacenter_name> 13
  datastore: <vcenter_datastore_name> 14
  folder: <vcenter_vm_folder_path> 15
  resourcePool: <vsphere_resource_pool> 16
  server: <vcenter_server_ip> 17

```

- 1 3 5 クラスターのプロビジョニング時に設定したクラスター ID を基にするインフラストラクチャー ID を指定します。以下のコマンドを実行してインフラストラクチャー ID を取得できます。

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 4 6 Windows マシンセット名を指定します。マシンセットの名前は、マシン名が vSphere で生成される方法により、9 文字を超えることができません。
- 7 Windows マシンとしてマシンセットを設定します。
- 8 Windows ノードをコンピュートマシンとして設定します。
- 9 vSphere 仮想マシンディスク (VMDK) のサイズを指定します。



#### 注記

このパラメーターは、Windows パーティションのサイズを設定しません。**unattend.xml** ファイルを使用するか、必要なディスクサイズで vSphere Windows 仮想マシン (VM) ゴールデンイメージを作成することにより、Windows パーティションのサイズを変更できます。

- 10 マシンセットをデプロイする vSphere 仮想マシンネットワークを指定します。この仮想マシンネットワークは、他の Linux コンピューティングマシンがクラスター内に存在する場所である必要があります。
- 11 **golden-images/windows-server-template** などの、使用する Windows vSphere 仮想マシンテンプレートの完全パスを指定します。名前は一意である必要があります。



#### 重要

元の仮想マシンテンプレートは指定しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規の Windows マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

- 12 **windows-user-data** は、最初の Windows マシンの設定時に WMCO によって作成されます。その後、後続のすべてのマシンセットで **windows-user-data** を消費できるようになります。
- 13 マシンセットをデプロイする vCenter Datacenter を指定します。
- 14 マシンセットをデプロイする vCenter Datastore を指定します。



- 15 `/dc1/vm/user-inst-5ddjd` などの vCenter の vSphere 仮想マシンフォルダーへのパスを指定します。
- 16 オプション: Windows 仮想マシンの vSphere リソースプールを指定します。
- 17 vCenter サーバーの IP または完全修飾ドメイン名を指定します。

#### 5.2.4. マシンセットの作成

インストールプログラムによって作成されるコンピュートセットセットに加えて、独自のマシンセットを作成して、選択した特定のワークロードのマシンコンピューティングリソースを動的に管理できます。

##### 前提条件

- OpenShift Container Platform クラスタをデプロイすること。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

##### 手順

1. 説明されているようにマシンセット カスタムリソース (CR) サンプルを含む新規 YAML ファイルを作成し、そのファイルに `<file_name>.yaml` という名前を付けます。  
`<clusterID>` および `<role>` パラメーターの値を設定していることを確認します。
2. オプション: 特定のフィールドに設定する値がわからない場合は、クラスタから既存のコンピュートマシンセットを確認できます。
  - a. クラスタ内のコンピュートマシンセットをリスト表示するには、次のコマンドを実行します。

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

##### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 特定のコンピュートマシンセットカスタムリソース(CR)の値を表示するには、以下のコマンドを実行します。

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

##### 出力例

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...

```

❶ クラスタインフラストラクチャー ID。

❷ デフォルトのノードラベル。



#### 注記

ユーザーがプロビジョニングしたインフラストラクチャーを持つクラスターの場合、コンピュートマシンセットは **worker** および **infra** タイプのマシンのみを作成できます。

❸ コンピュートマシンセット CR の **<providerSpec>** セクションの値は、プラットフォーム固有です。CR の **<providerSpec>** パラメーターの詳細については、プロバイダーのサンプルコンピュートマシンセット CR 設定を参照してください。

3. 次のコマンドを実行して **MachineSet** CR を作成します。

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

#### 検証

- 次のコマンドを実行して、コンピュートマシンセットのリストを表示します。

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

#### 出力例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxx-windows-worker-us-east-1a	1	1	1	1	11m



agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

新規のマシンセットが利用可能な場合、**DESIRED** および **CURRENT** の値は一致します。マシンセットが利用可能でない場合、数分待機してからコマンドを再度実行します。

### 5.2.5. 関連情報

- [マシン管理の概要](#)

## 第6章 WINDOWS コンテナワークロードのスケジューリング

Windows ワークロードを Windows コンピュートノードにスケジュールすることができます。



### 注記

WMCO はワークロードのプロキシ接続を介してトラフィックをルーティングできないため、[クラスター全体のプロキシ](#)を使用するクラスターではサポートされません。

### 前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Docker 形式のコンテナランタイムが有効な状態で Windows コンテナを OS イメージとして使用している。
- Windows マシンセットを作成している。



### 重要

現時点で、Docker 形式のコンテナランタイムは Windows ノードで使用されます。Kubernetes では、コンテナランタイムとしての Docker を非推奨としています。詳細は、Kubernetes ドキュメントの [Docker の非推奨](#) について参照してください。Containerd は、今後の Kubernetes リリースで Windows ノードについてサポートされる新しいコンテナランタイムになります。

## 6.1. WINDOWS POD の配置

Windows ワークロードをクラスターにデプロイする前に、Pod が適切に割り当てられるように Windows ノードのスケジューリングを設定する必要があります。Windows ノードをホストするマシンがあるので、これは Linux ベースのノードと同じように管理できます。同様に、テイント、容認およびノードセクターなどのメカニズムを使用して、Windows Pod の適切な Windows ノードへのスケジュールも同様に実行されます。

複数のオペレーティングシステム、および同じクラスターで複数の Windows OS バリエーションを実行する機能で、**RuntimeClass** オブジェクトを使用して Windows Pod をベース Windows OS バリエーションにマップする必要があります。たとえば、複数の Windows ノードが複数の Windows Server コンテナのバージョンで実行されている場合、クラスターは Windows Pod を互換性のない Windows OS バリエーションにスケジュールする可能性があります。クラスター上の Windows OS バリエーションごとに **RuntimeClass** オブジェクトを設定する必要があります。クラスターで1つの Windows OS バリエーションのみが利用可能である場合、**RuntimeClass** オブジェクトを使用することも推奨されます。

詳細は、[ホストとコンテナのバージョンの互換性](#) を参照してください。



### 重要

コンテナの基本イメージは、コンテナがスケジュールされるノードで実行されているものと同じ Windows OS バージョンおよびビルド番号である必要があります。

また、Windows ノードをあるバージョンから別のバージョンにアップグレードする場合（たとえば、20H2 から 2022 に移行する場合）、新しいバージョンに一致するようにコンテナの基本イメージをアップグレードする必要があります。詳細については、[Windows コンテナのバージョンの互換性](#) を参照してください。

## 関連情報

- [スケジューラーによる Pod 配置の制御](#)
- [ノードテイントを使用した Pod 配置の制御](#)
- [ノードセクターの使用による特定ノードへの Pod の配置](#)

## 6.2. スケジューリングメカニズムをカプセル化するための RUNTIMECLASS オブジェクトの作成

**RuntimeClass** オブジェクトを使用することにより、テイントおよび容認などのスケジューリングの仕組みの使用を単純化できます。テイントおよび容認をカプセル化するランタイムクラスをデプロイしてから、これを Pod に適用して Pod を適切なノードにスケジューリングできるようにします。ランタイムクラスの作成は、複数のオペレーティングシステムのバリエーションをサポートするクラスターでも必要になります。

### 手順

1. **RuntimeClass** オブジェクト YAML ファイルを作成します。例: **runtime-class.yaml**

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: <runtime_class_name> ❶
handler: 'docker'
scheduling:
  nodeSelector: ❷
    kubernetes.io/os: 'windows'
    kubernetes.io/arch: 'amd64'
    node.kubernetes.io/windows-build: '10.0.17763'
  tolerations: ❸
    - effect: NoSchedule
      key: os
      operator: Equal
      value: "Windows"
```

- ❶ このランタイムクラスで管理する必要のある Pod で定義される **RuntimeClass** オブジェクト名を指定します。
- ❷ このランタイムクラスをサポートするノードに存在する必要があるラベルを指定します。このランタイムクラスを使用する Pod は、このセクターに一致するノードにのみスケジューリングできます。ランタイムクラスのノードセクターは Pod の既存のノードセクターとマージされます。競合が発生した場合は、Pod をノードにスケジューリングできなくなります。
- ❸ Pod に追加する容認を指定します。ただし、受付時にこのランタイムクラスで実行される重複を除きます。これによって、Pod によって許容されるノードのセットとランタイムクラスが組み合わされます。

2. **RuntimeClass** オブジェクトを作成します。

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

以下に例を示します。

```
$ oc create -f runtime-class.yaml
```

3. **RuntimeClass** オブジェクトを Pod に適用し、これが適切なオペレーティングシステムバリエーションにスケジュールされていることを確認します。

```
apiVersion: v1
kind: Pod
metadata:
  name: my-windows-pod
spec:
  runtimeClassName: <runtime_class_name> ❶
...
```

- ❶ Pod のスケジュールを管理するためにランタイムクラスを指定します。

### 6.3. WINDOWS コンテナワークロードのデプロイメント例

Windows コンピュートノードが利用可能になる時点で、Windows コンテナワークロードをクラスターにデプロイできます。



#### 注記

このサンプルデプロイメントは参照用にのみ提供されます。

#### Service オブジェクトの例

```
apiVersion: v1
kind: Service
metadata:
  name: win-webserver
labels:
  app: win-webserver
spec:
  ports:
    # the port that this service should serve on
    - port: 80
      targetPort: 80
  selector:
    app: win-webserver
  type: LoadBalancer
```

#### Deployment オブジェクトの例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: win-webserver
  name: win-webserver
spec:
```

```

selector:
  matchLabels:
    app: win-webserver
replicas: 1
template:
  metadata:
    labels:
      app: win-webserver
      name: win-webserver
  spec:
    tolerations:
      - key: "os"
        value: "Windows"
        Effect: "NoSchedule"
    containers:
      - name: windowswebserver
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        imagePullPolicy: IfNotPresent
        command:
          - powershell.exe
          - -command
            - $listener = New-Object System.Net.HttpListener; $listener.Prefixes.Add('http://*:80/');
              $listener.Start(); Write-Host('Listening at http://*:80/'); while ($listener.IsListening) { $context =
                $listener.GetContext(); $response = $context.Response; $content='<html><body><H1>Red Hat
                OpenShift + Windows Container Workloads</H1></body></html>'; $buffer =
                [System.Text.Encoding]::UTF8.GetBytes($content); $response.ContentLength64 = $buffer.Length;
                $response.OutputStream.Write($buffer, 0, $buffer.Length); $response.Close(); };
        securityContext:
          runAsNonRoot: false
          windowsOptions:
            runAsUserName: "ContainerAdministrator"
    nodeSelector:
      kubernetes.io/os: windows

```



### 注記

**mcr.microsoft.com/powershell:** コンテナイメージを使用する場合、コマンドを **pwsh.exe** として定義する必要があります。**mcr.microsoft.com/windows/servercore:** **<tag>** コンテナイメージを使用している場合、コマンドを **powershell.exe** として定義する必要があります。詳細は、Microsoft のドキュメントを参照してください。

## 6.4. マシンセットの手動によるスケーリング

マシンセットのマシンのインスタンスを追加したり、削除したりする必要がある場合、マシンセットを手動でスケーリングできます。

本書のガイダンスは、完全に自動化されたインストーラーでプロビジョニングされるインフラストラクチャのインストールに関連します。ユーザーによってプロビジョニングされるインフラストラクチャのカスタマイズされたインストールにはマシンセットがありません。

### 前提条件

- OpenShift Container Platform クラスターおよび **oc** コマンドラインをインストールすること。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

## 手順

1. クラスタにあるマシンセットを表示します。

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

マシンセットは **<clusterid>-worker-<aws-region-az>** の形式で一覧表示されます。

2. クラスタ内にあるマシンを表示します。

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

3. 削除するマシンに注釈を設定します。

```
$ oc annotate machine/<machine_name> -n openshift-machine-api
machine.openshift.io/cluster-api-delete-machine="true"
```

4. 次のいずれかのコマンドを実行して、コンピュータマシンセットをスケーリングします。

```
$ oc scale --replicas=2 machineset <machineset> -n openshift-machine-api
```

または、以下を実行します。

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

## ヒント

または、以下の YAML を適用してマシンセットをスケーリングすることもできます。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machineset>
  namespace: openshift-machine-api
spec:
  replicas: 2
```

コンピュータマシンセットをスケールアップまたはスケールダウンできます。新規マシンが利用可能になるまで数分の時間がかかります。



### 重要

デフォルトでは、マシンコントローラーは、成功するまでマシンによってサポートされるノードをドレイン (解放) しようとします。Pod 中断バジレットの設定が間違っているなど、状況によっては、ドレイン操作が成功しない可能性があります。排水操作が失敗した場合、マシンコントローラーはマシンの取り外しを続行できません。

特定のマシンの **machine.openshift.io/exclude-node-draining** にアノテーションを付けると、ノードのドレイン (解放) を省略できます。

## 検証

- 目的のマシンの削除を確認します。

```
$ oc get machines
```

## 第7章 WINDOWS ノードのアップグレード

Windows Machine Config Operator (WMCO) をアップグレードすることで、Windows ノードに最新の更新が含まれることを確認できます。

### 7.1. WINDOWS MACHINE CONFIG OPERATOR のアップグレード

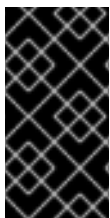
現在のクラスターバージョンと互換性のある Windows Machine Config Operator (WMCO) の新規バージョンがリリースされると、Operator はアップグレードチャネル、および Operator Lifecycle Manager (OLM) を使用する際にインストールに使用されたサブスクリプションの承認ストラテジーに基づいてアップグレードされます。WMCO のアップグレードにより、Windows マシンの Kubernetes コンポーネントがアップグレードされます。



#### 注記

WMCO の新規バージョンにアップグレードする場合でクラスターモニターリングを使用する必要がある場合は、WMCO namespace に **openshift.io/cluster-monitoring=true** ラベルが必要です。ラベルを既存の WMCO namespace に追加し、すでに Windows ノードが設定されている場合は、WMCO Pod を再起動してモニターリンググラフを表示できるようにします。

中断なしのアップグレードの場合、WMCO は以前のバージョンの WMCO で設定された Windows マシンを中止し、現行バージョンを使用してそれらを再作成します。これは、**Machine** オブジェクトを削除して実行されます。これにより、Windows ノードのドレイン (解放) および削除が実行されます。アップグレードを容易にするために、WMCO は設定されたすべてのノードにバージョンのアノテーションを追加します。アップグレード時に、バージョンのアノテーションで不一致があると、Windows マシンが削除され、再作成されます。アップグレード時のサービスの中断を最小限にするために、WMCO は一度に1つの Windows マシンのみを更新します。



#### 重要

WMCO は Kubernetes コンポーネントの更新のみを行い、Windows オペレーティングシステムの更新は行いません。仮想マシンの作成時に Windows イメージを指定できるため、更新されたイメージを指定できます。**MachineSet** 仕様でイメージ設定を変更して、更新された Windows イメージを指定できます。

Operator Lifecycle Manager (OLM) を使用した Operator のアップグレードの詳細は、[インストールされた Operator のアップグレード](#) を参照してください。



## 第8章 BYOH (BRING-YOUR-OWN-HOST) WINDOWS インスタンスをノードとして使用

BYOH (Bring-Your-Own-Host) を使用すると、ユーザーは Windows Server 仮想マシンを再利用して、OpenShift Container Platform に移動できます。BYOH Windows インスタンスは、Windows サーバーがオフラインになった場合に、主要な中断を軽減するのに役立ちます。

### 8.1. BYOH WINDOWS インスタンスの設定

BYOH Windows インスタンスを作成するには、Windows Machine Config Operator (WMCO) namespace に Config Map を作成する必要があります。

#### 前提条件

ノードとしてクラスターに割り当てられる Windows インスタンスはすべて、以下の要件を満たす必要があります。

- Docker コンテナランタイムがインスタンスにインストールされている必要があります。
- インスタンスは、クラスターの Linux ワーカーノードと同じネットワーク上にある必要があります。
- ポート 22 が開いていて、SSH サーバーを実行している必要があります。
- SSH サーバーのデフォルトシェルは、[Windows コマンドシェル](#) または **cmd.exe** である必要があります。
- ログコレクションに対してポート 10250 を開く必要があります。
- 管理者ユーザーは、認可された SSH キーとしてシークレットセットで使用される秘密鍵とともに存在します。
- インストーラーによってプロビジョニングされたインフラストラクチャー (IPI) AWS クラスター用に BYOH Windows インスタンスを作成する場合は、ワーカーノード用に設定されたマシンの **spec.template.spec.value.tag** 値と一致する AWS インスタンスにタグを追加する必要があります。たとえば、**kubernetes.io/cluster/<cluster\_id>: owned** または **kubernetes.io/cluster/<cluster\_id>: shared** です。
- vSphere で BYOH Windows インスタンスを作成する場合は、内部 API サーバーとの通信を有効にする必要があります。
- インスタンスのホスト名は、以下の標準を含む [RFC 1123](#) DNS ラベルの要件に従う必要があります。
  - 小文字の英数字またはハイフン (-) のみが含まれます。
  - 英数字から始まります。
  - 英数字の文字で終わります。

#### 手順

1. 追加する Windows インスタンスを記述する WMCO namespace に **windows-instances** という名前の ConfigMap を作成します。



### 注記

値を **username=<username>** としてフォーマットし、アドレスをキーとして使用し、設定マップの data セクションの各エントリーをフォーマットします。

### 設定マップの例

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: windows-instances
  namespace: openshift-windows-machine-config-operator
data:
  10.1.42.1: |- ❶
    username=Administrator ❷
  instance.example.com: |-
    username=core

```

- ❶ WMCO が SSH 経由でインスタンスに到達するために使用するアドレス (DNS 名または IPv4 アドレス)。このアドレスの DNS PTR レコードが存在する必要があります。組織が DHCP を使用して IP アドレスを割り当てる場合は、BYOH インスタンスで DNS 名を使用することをお勧めします。そうでない場合は、インスタンスに新しい IP アドレスが割り当てられるたびに、**windows-instances** ConfigMap を更新する必要があります。
- ❷ 前提条件で作成した管理者ユーザーの名前。

## 8.2. BYOH WINDOWS インスタンスの削除

設定マップでインスタンスのエントリーを削除して、クラスターに割り当てられた BYOH インスタンスを削除できます。インスタンスを削除すると、クラスターに追加する前にそのインスタンスがその状態に戻ります。ログおよびコンテナランタイムアーティファクトは、これらのインスタンスには追加されません。

インスタンスを正常に削除するには、WMCO に提供される現在のプライベートキーを使用してアクセスする必要があります。たとえば、直前の例から **10.1.42.1** インスタンスを削除するには、設定マップを以下に変更します。

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: windows-instances
  namespace: openshift-windows-machine-config-operator
data:
  instance.example.com: |-
    username=core

```

**windows-instances** を削除すると、ノードとして追加されるすべての Windows インスタンスを分解する要求として表示されます。

## 第9章 WINDOWS ノードの削除

ホスト Windows マシンを削除して、Windows ノードを削除できます。

### 9.1. 特定マシンの削除

特定のマシンを削除できます。



#### 注記

コントロールプレーンマシンは削除できません。

#### 前提条件

- OpenShift Container Platform クラスターをインストールします。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** パーミッションを持つユーザーとして、**oc** にログインする。

#### 手順

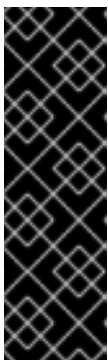
1. 次のコマンドを実行して、クラスター内のマシンを表示します。

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

コマンド出力には、**<clusterid>-<role>-<cloud\_region>** 形式のマシンのリストが含まれます。

2. 削除するマシンを特定します。
3. 次のコマンドを実行してマシンを削除します。

```
$ oc delete machine <machine> -n openshift-machine-api
```



#### 重要

デフォルトでは、マシンコントローラーは、成功するまでマシンによってサポートされるノードをドレイン (解放) しようとします。Pod 中断バジェットの設定が間違っているなど、状況によっては、ドレイン操作が成功しない可能性があります。排水操作が失敗した場合、マシンコントローラーはマシンの取り外しを続行できません。

特定のマシンの **machine.openshift.io/exclude-node-draining** にアノテーションを付けると、ノードのドレイン (解放) を省略できます。

削除するマシンがマシンセットに属している場合は、指定された数のレプリカを満たす新しいマシンがすぐに作成されます。

## 第10章 WINDOWS コンテナワークロードの無効化

Windows コンテナワークロードを実行する機能を無効にするには、Windows Machine Config Operator (WMCO) をアンインストールし、WMCO のインストール時にデフォルトで追加された namespace を削除します。

### 10.1. WINDOWS MACHINE CONFIG OPERATOR のアンインストール

クラスターから Windows Machine Config Operator (WMCO) をアンインストールできます。

#### 前提条件

- Windows ワークロードをホストする Windows **Machine** オブジェクトを削除します。

#### 手順

- Operators → OperatorHub ページから、Filter by keyword ボックスを使用して、**Red Hat Windows Machine Config Operator** を検索します。
- Red Hat Windows Machine Config Operator** タイルをクリックします。Operator タイルはこれがインストールされていることを示します。
- Windows Machine Config Operator** 記述子ページで、**Uninstall** をクリックします。

### 10.2. WINDOWS MACHINE CONFIG OPERATOR NAMESPACE の削除

デフォルトで Windows Machine Config Operator (WMCO) 用に生成された namespace を削除できます。

#### 前提条件

- WMCO がクラスターから削除される。

#### 手順

- openshift-windows-machine-config-operator** namespace で作成されたすべての Windows ワークロードを削除します。

```
$ oc delete --all pods --namespace=openshift-windows-machine-config-operator
```

- openshift-windows-machine-config-operator** namespace のすべての Pod が削除されているか、または終了状態を報告していることを確認します。

```
$ oc get pods --namespace openshift-windows-machine-config-operator
```

- openshift-windows-machine-config-operator** namespace を削除します。

```
$ oc delete namespace openshift-windows-machine-config-operator
```

#### 関連情報

- [クラスターからの Operator の削除](#)

- [Windows ノードの削除](#)