



# OpenShift Container Platform 4.12

## OpenShift 的 Windows 容器支持

Red Hat OpenShift for Windows Containers 指南



# OpenShift Container Platform 4.12 OpenShift 的 Windows 容器支持

---

Red Hat OpenShift for Windows Containers 指南

## 法律通告

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

Linux<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.

## 摘要

Red Hat OpenShift for Windows Containers 为在 OpenShift Container Platform 上运行 Microsoft Windows Server 容器提供了内置的支持。本指南提供所有详细信息。

# 目录

<b>第 1 章 RED HAT OPENSIFT 对 WINDOWS CONTAINERS 的支持概述</b> .....	<b>3</b>
<b>第 2 章 发行注记</b> .....	<b>4</b>
2.1. RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS 发行注记	4
2.2. WINDOWS MACHINE CONFIG OPERATOR 的过去版本的发行注记	4
2.3. WINDOWS MACHINE CONFIG OPERATOR 的先决条件	6
2.4. WINDOWS MACHINE CONFIG OPERATOR 已知的限制	8
<b>第 3 章 获取支持</b> .....	<b>10</b>
<b>第 4 章 了解 WINDOWS 容器工作负载</b> .....	<b>11</b>
4.1. WINDOWS 工作负载管理	11
4.2. WINDOWS 节点服务	13
<b>第 5 章 启用 WINDOWS 容器工作负载</b> .....	<b>15</b>
先决条件	15
5.1. 安装 WINDOWS MACHINE CONFIG OPERATOR	15
5.2. 为 WINDOWS MACHINE CONFIG OPERATOR 配置 SECRET	18
5.3. 在支持代理的集群中使用 WINDOWS 容器	19
5.4. 其他资源	19
<b>第 6 章 创建 WINDOWS 机器集</b> .....	<b>20</b>
6.1. 在 AWS 上创建 WINDOWS 机器集	20
6.2. 在 AZURE 上创建 WINDOWS 机器集	25
6.3. 在 VSPHERE 上创建 WINDOWS 机器集	30
6.4. 在 GCP 上创建 WINDOWS 机器集	39
<b>第 7 章 调度 WINDOWS 容器工作负载</b> .....	<b>45</b>
先决条件	45
7.1. WINDOWS POD 放置	45
7.2. 创建 RUNTIMECLASS 对象来封装调度机制	45
7.3. WINDOWS 容器工作负载部署示例	47
7.4. 手动扩展计算机器集	48
<b>第 8 章 WINDOWS 节点升级</b> .....	<b>50</b>
8.1. WINDOWS MACHINE CONFIG OPERATOR 升级	50
<b>第 9 章 使用 BRING-YOUR-OWN-HOST (BYOH) WINDOWS 实例作为节点</b> .....	<b>51</b>
9.1. 配置 BYOH WINDOWS 实例	51
9.2. 删除 BYOH WINDOWS 实例	52
<b>第 10 章 删除 WINDOWS 节点</b> .....	<b>53</b>
10.1. 删除一个特定的机器	53
<b>第 11 章 禁用 WINDOWS 容器工作负载</b> .....	<b>54</b>
11.1. 卸载 WINDOWS MACHINE CONFIG OPERATOR	54
11.2. 删除 WINDOWS MACHINE CONFIG OPERATOR 命名空间	54



# 第 1 章 RED HAT OPENSIFT 对 WINDOWS CONTAINERS 的支持概述

Red Hat OpenShift for Windows Containers 提供了在 OpenShift Container Platform 集群中运行 Windows 计算节点的功能。这可以通过使用 Red Hat Windows Machine Config Operator (WMCO) 来安装和管理 Windows 节点来实现。通过红帽订阅，您可以获得对在 OpenShift Container Platform 中运行 Windows 工作负载的支持。WMCO 部署的 Windows 实例使用容器运行时配置。如需更多信息，请参阅[发行注记](#)。

您可以通过创建[计算机器集](#)，或使用一个[配置映射](#)指定现有 Bring-Your-Own-Host (BYOH) Window 实例来添加 Windows 节点。



## 注意

裸机或供应商无关的集群不支持计算机器集。

对于 Linux 和 Windows 等工作负载，OpenShift Container Platform 允许您部署在 Windows Server 容器上运行的 Windows 工作负载，同时还提供托管在 Red Hat Enterprise Linux CoreOS (RHCOS) 或 Red Hat Enterprise Linux (RHEL) 上的传统 Linux 工作负载。如需更多信息，请参阅[开始使用 Windows 容器工作负载](#)。

您需要 WMCO 在集群中运行 Windows 工作负载。WMCO 在集群中管理部署和管理 Windows 工作负载的过程。如需更多信息，请参阅[如何启用 Windows 容器工作负载](#)。

您可以创建一个 Windows **MachineSet** 对象来创建基础架构 Windows 机器集和相关的机器，以便将支持的 Windows 工作负载移到新的 Windows 机器。您可以在多个平台上创建 Windows **MachineSet** 对象。

您可以将 [Windows 工作负载调度到](#) Windows 计算节点。

您可以执行 [Windows Machine Config Operator 升级](#) 以确保 Windows 节点有最新的更新。

您可以通过删除特定机器来删除 [Windows 节点](#)。

您可以[使用 Bring-Your-Own-Host\(BYOH\)Windows 实例](#) 重新使用 Windows Server 虚拟机并将其引入 OpenShift Container Platform。BYOH Windows 实例让希望降低 Windows 服务器离线时发生重大中断的用户受益。您可以使用 BYOH Windows 实例作为 OpenShift Container Platform 4.8 及更新的版本的节点。

您可以通过执行以下操作 [禁用 Windows 容器工作负载](#)：

- 卸载 Windows Machine Config Operator
- 删除 Windows Machine Config Operator 命名空间

## 第 2 章 发行注记

### 2.1. RED HAT OPENSIFT SUPPORT FOR WINDOWS CONTAINERS 发行注记

Red Hat OpenShift for Windows Containers 发行注记包括了 Windows Machine Config Operator (WMCO) 的开发，它在 OpenShift Container Platform 中提供所有 Windows 容器工作负载功能。

#### 2.1.1. Red Hat Windows Machine Config Operator 7.2.2 发行注记

此 WMCO 发行版本为在 OpenShift Container Platform 集群中运行的 Windows 计算节点提供安全更新和程序错误修复。WMCO 7.2.2 组件在 [RHSA-2024:6734](#) 中发布。

### 2.2. WINDOWS MACHINE CONFIG OPERATOR 的过去版本的发行注记

以下发行注记适用于 Windows Machine Config Operator (WMCO) 的早期版本。

有关当前版本，请参阅 [Red Hat OpenShift support for Windows Containers 发行注记](#)。

#### 2.2.1. Red Hat Windows Machine Config Operator 7.2.1 发行注记

此 WMCO 发行版本为在 OpenShift Container Platform 集群中运行 Windows 计算节点提供程序错误修正。WMCO 7.2.1 组件在 [RHBA-2024:1476](#) 中发布。

##### 2.2.1.1. 程序错误修复

- 在以前的版本中，WMCO 无法正确等待 Windows 虚拟机(VM)完成重启。这会导致 WMCO 在重启过程中尝试与节点交互的时间问题，从而导致 WMCO 记录错误并重启节点配置。现在，WMCO 会等待实例完全重启。([OCPBUGS-23036](#))
- 在以前的版本中，WMCO 配置缺少 **DeleteEmptyDirData: true** 字段，这是排空附加了 **emptyDir** 卷的节点所必需的。因此，具有 **emptyDir** 卷的节点的客户会在日志中看到以下错误：**cannot delete Pods with local storage**。在这个版本中，**DeleteEmptyDirData: true** 字段被添加到 WMCO 中的节点排空帮助程序结构中。因此，客户可以排空附加了 **emptyDir** 卷的节点。([OCPBUGS-23081](#))
- 在以前的版本中，由于网络配置脚本中的错误逻辑，WICD 被错误地读取 CNI 配置文件中的 carriage 返回，并将该文件标识为修改。这会导致 CNI 配置被不必要重新加载，可能会导致容器重启和短网络中断。在这个版本中，WICD 现在仅在实际修改 CNI 配置时重新载入 CNI 配置。([OCPBUGS-27771](#))
- 在以前的版本中，WMCO 会错误地为尝试加入集群的所有节点批准节点证书签名请求 (CSR)，而不只是 Windows 节点 CSR。在这个版本中，WMCO 只针对 Windows 节点批准 CSR。([OCPBUGS-27139](#))
- 在以前的版本中，由于 Windows Server 2019 中出现的路由问题，在某些条件下，运行了一小时后，Windows Server 2019 上的工作负载在与集群中的其他容器通信时可能会出现数据包丢失的情况。在这个版本中，在 kube-proxy 中启用直接服务器返回 (DSR) 路由。因此，DSR 现在会导致请求和响应流量使用不同的网络路径，绕过 Windows Server 2019 中的 bug。([OCPBUGS-28254](#))



- 在以前的版本中，因为从 WMCO 6.x 升级到 WMCO 7.x 的升级路径包括以前发布的版本，所以 WMCO 在升级过程中会失败。在这个版本中，您可以成功从 WMCO 6.x 升级到 WMCO 7.x。  
([OCPBUGS-27775](#))
- 在以前的版本中，由于在更新计算机集节点时缺少 Windows 计算机集节点和 Bring-Your-Own-Host (BYOH) 实例之间的同步，并且 BYOH 实例可以同时更新，这可能会同时更新受影响的工作负载。在这个版本中，引入了一个锁定机制，以便计算机集节点和 BYOH 实例单独更新。  
([OCPBUGS-23020](#))

## 2.2.2. Red Hat Windows Machine Config Operator 7.1.0 发行注记

此 WMCO 发行版本为在 OpenShift Container Platform 集群中运行 Windows 计算节点提供程序错误修正。WMCO 7.1.0 组件在 [RHSA-2023:4025](#) 中发布。



### 重要

由于一个已知问题，OpenShift Container Platform 集群必须在将 WMCO 从 7.0.1 更新至 7.1.0 版本前为 4.12.3 或更高版本。如果集群低于 4.12.3 版本，则更新会失败。

### 2.2.2.1. 程序错误修复

- 在以前的版本中，**containerd** 容器运行时在每个 Windows 节点上报告了一个不正确的版本，因为存储库标签没有传播到构建系统。此配置导致 **containerd** 将 **go build** 版本报告为每个 Windows 节点的版本。在这个版本中，在构建期间将正确的版本注入二进制文件，以便 **containerd** 报告每个 Windows 节点的正确版本。  
([OCPBUGS-7843](#))
- 在以前的版本中，Windows Machine Config Operator (WMCO) 无法排空守护进程集工作负载。此问题导致 Windows 守护进程集 pod 阻止 WMCO 尝试删除或更新的 Windows 节点。在这个版本中，WMCO 包括额外的基于角色的访问控制 (RBAC) 权限，以便 WMCO 可以删除守护进程集工作负载。WMCO 还可以删除使用容器 shim 创建的进程，以便在 WMCO 从集群中删除节点后，守护进程集容器不会在 Windows 实例中不存在。  
([OCPBUGS-8056](#))
- 在以前的版本中，在没有安装 Azure 容器服务的 Azure Windows Server 2019 平台上，WMCO 将无法部署 Windows 实例，并显示 **Install-WindowsFeature : Win32 internal error "Access is denied" 0x5 occurred when the console output buffer** 错误信息。发生了故障，因为 Microsoft **Install-WindowsFeature** cmdlet 显示进度条，无法通过 SSH 连接发送。在这个版本中，隐藏了进度条。因此，Windows 实例可以部署为节点。  
([OCPBUGS-14445](#))

## 2.2.3. Release notes for Red Hat Windows Machine Config Operator 7.0.1

此 WMCO 发行版本为在 OpenShift Container Platform 集群中运行 Windows 计算节点提供程序错误修正。WMCO 7.0.1 的组件在 [RHBA-2023:0748](#) 中发布。

### 2.2.3.1. 程序错误修复

- 在以前的版本中，WMCO 7.0.0 不支持在 **openshift-windows-machine-operator** 以外的命名空间中运行。在这个版本中，您可以在自定义命名空间中运行 WMCO，并可以升级在自定义命名空间中安装 WMCO 的集群。  
([OCPBUGS-5065](#))

## 2.2.4. Red Hat Windows Machine Config Operator 7.0.0 发行注记

此 WMCO 发行版本为在 OpenShift Container Platform 集群中运行 Windows 计算节点提供程序错误修正。WMCO 7.0.0 组件在 [RHSA-2022:9096](#) 中发布。

## 2.2.4.1. 新功能及改进

### 2.2.4.1.1. Windows 实例配置守护进程 (WICD)

Windows 实例配置守护进程 (WICD) 现在执行之前由 Windows Machine Config Bootstrapper (WMCB) 执行的许多任务。WICD 安装在您的 Windows 节点上。用户不需要与 WICD 交互，且不应在 WMCO 操作中遇到任何区别。

### 2.2.4.1.2. 支持在 Google Cloud Platform 上运行的集群

现在，您可以在 Google Cloud Platform (GCP) 上安装的集群中运行 Windows Server 2022 节点。您可以在 GCP 上创建 Windows **MachineSet** 对象来托管 Windows Server 2022 计算节点。如需更多信息，请参阅在 [vSphere 上创建 Windows MachineSet 对象](#)。

### 2.2.4.2. 程序错误修复

- 在以前的版本中，重启带有正在运行的 Windows 节点的集群中的 WMCO 会导致删除 windows exporter 端点。因此，每个 Windows 节点都无法报告任何指标数据。在这个版本中，当 WMCO 重启时，端点会被保留。因此，重启 WMCO 后会正确报告指标数据。([BZ#2107261](#))
- 在以前的版本中，测试来确定 Windows Defender antivirus 服务是否在运行，无论状态如何，都错误地检查其名称以 Windows Defender 开头的任何进程。这会导致在没有安装 Windows Defender 的情况下为容器创建防火墙排除时出现错误。在这个版本中，检查是否与 Windows Defender antivirus 服务关联的特定正在运行的进程。因此，WMCO 可以正确地将 Windows 实例配置为节点，无论是否安装了 Windows Defender。([OCPBUGS-3573](#))

### 2.2.4.3. 已知问题

之前的 WMCO 发行版本后会宣布以下已知的限制：

- OpenShift Serverless、Horizontal Pod 自动扩展和 Vertical Pod 自动扩展在 Windows 节点上不支持。
- Red Hat OpenShift support for Windows Containers 不支持通过中继端口在集群中添加 Windows 节点。添加 Windows 节点唯一支持的网络配置是通过为 VLAN 传输流量的访问端口。
- WMCO 7.0.0 不支持在 **openshift-windows-machine-operator** 以外的命名空间中运行。如果使用自定义命名空间，建议您不要升级到 WMCO 7.0.0。相反，您应该在发布了 WMCO 7.0.1 时升级到 WMCO 7.0.1。如果您的 WMCO 配置了 **Automatic** 更新批准策略，您应该将其更改为 WMCO 7.0.0 的 **Manual**。有关更改批准策略的详情，请参考 [安装说明](#)。

#### 其他资源

查看 [已知限制](#) 的完整列表

## 2.3. WINDOWS MACHINE CONFIG OPERATOR 的先决条件

以下信息详细介绍了 Windows Machine Config Operator 支持的平台版本、Windows Server 版本和网络配置。有关仅与该平台相关的任何信息请参阅 [vSphere 文档](#)。

### 2.3.1. WMCO 7.2.x 支持的平台和 Windows 服务器版本

下表根据适用的平台列出了 WMCO 7.2.0 和 7.2.1 支持的 [Windows Server 版本](#)。不支持未列出的 Windows Server 版本，使用它们将可能会导致错误。

平台	支持的 Windows Server 版本
Amazon Web Services (AWS)	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> <li>Windows Server 2019, 版本 1809</li> </ul>
Microsoft Azure	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> <li>Windows Server 2019, 版本 1809</li> </ul>
VMware vSphere	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> </ul>
Google Cloud Platform (GCP)	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> </ul>
裸机或供应商无关	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> <li>Windows Server 2019, 版本 1809</li> </ul>

### 2.3.2. WMCO 7.0 和 7.1 支持的平台和 Windows 服务器版本

下表根据适用的平台列出了 WMCO 7.0.0, 7.0.1 和 7.1.0 支持的 [Windows Server 版本](#)。不支持未列出的 Windows Server 版本，使用它们将可能会导致错误。

平台	支持的 Windows Server 版本
Amazon Web Services (AWS)	<ul style="list-style-type: none"> <li>Windows Server 2019, 版本 1809</li> </ul>
Microsoft Azure	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> <li>Windows Server 2019, 版本 1809</li> </ul>
VMware vSphere	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> </ul>
Google Cloud Platform (GCP)	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> </ul>

平台	支持的 Windows Server 版本
裸机或供应商无关	<ul style="list-style-type: none"> <li>Windows Server 2022, OS Build <a href="#">20348.681</a> 或更高版本</li> <li>Windows Server 2019, 版本 1809</li> </ul>

## 2.4. WINDOWS MACHINE CONFIG OPERATOR 已知的限制

在使用由 WMCO 管理的 Windows 节点(Windows 节点)时, 请注意以下限制。

- Windows 节点上不支持以下 OpenShift Container Platform 功能：
  - 镜像构建
  - OpenShift Pipelines
  - OpenShift Service Mesh
  - OpenShift 监控用户定义的项目
  - OpenShift Serverless
  - Pod 横向自动扩展
  - Pod 垂直自动扩展
- Windows 节点上不支持以下红帽功能：
  - [Red Hat Insights 成本管理](#)
  - [Red Hat OpenShift Local](#)
- WMCO 管理的 Windows 实例不支持双 NIC。
- Windows 节点不支持使用部署配置创建的工作负载。您可以使用部署或其他方法部署工作负载。
- 使用集群范围代理的集群中不支持 Windows 节点。这是因为 WMCO 无法通过工作负载的代理连接路由流量。
- 在断开连接的环境中, 不支持 Windows 节点。
- Red Hat OpenShift support for Windows Containers 不支持通过中继端口在集群中添加 Windows 节点。添加 Windows 节点唯一支持的网络配置是通过为 VLAN 传输流量的访问端口。
- Red Hat OpenShift support for Windows Containers 只支持所有云供应商的树内存储驱动程序。
- Red Hat OpenShift support for Windows Containers 不支持 English (United States)以外的任何 Windows 操作系统语言。
- 由于 Windows 操作系统中的一个限制, 类 E 的 **clusterNetwork** CIDR 地址 (如 **240.0.0.0**) 与 Windows 节点不兼容。
- Kubernetes 有以下 [节点功能限制](#)：
  - Windows 容器不支持巨页。

- Windows 容器不支持特权容器。
- Kubernetes 已发现[几个 API 兼容性问题](#)。

## 第 3 章 获取支持

Windows Container Support for Red Hat OpenShift 作为一个可选的、可安装的组件提供。Windows Container Support for Red Hat OpenShift 不是 OpenShift Container Platform 订阅的一部分。它需要额外的红帽订阅，并根据[覆盖范围](#)和[服务等级协议](#)进行支持。

您必须具有此单独的订阅才能获得对 Red Hat OpenShift 的 Windows Container 支持。如果没有此附加红帽订阅，则不支持在生产环境中部署 Windows 容器工作负载。您可以通过[红帽客户门户网站](#)请求支持。

如需更多信息，请参阅 Red Hat OpenShift Container Platform 生命周期政策文档中关于 [Red Hat OpenShift support for Windows Containers](#) 的部分。

如果您没有额外的红帽订阅，可以使用 Community Windows Machine Config Operator，它是一个不被官方支持的版本。

## 第 4 章 了解 WINDOWS 容器工作负载

Red Hat OpenShift for Windows Containers 为在 OpenShift Container Platform 上运行 Microsoft Windows Server 容器提供了内置的支持。对于使用 Linux 和 Windows 工作负载管理异构环境的管理员，OpenShift Container Platform 允许您部署在 Windows Server 容器上运行的 Windows 工作负载，同时也提供托管在 Red Hat Enterprise Linux CoreOS (RHCOS) 或 Red Hat Enterprise Linux (RHEL) 上的传统 Linux 工作负载。



### 注意

不支持具有 Windows 节点的集群的多租户。当多个工作负载在共享基础架构和资源上运行时，集群被视为 **多租户**。如果基础架构上运行的一个或多个工作负载无法信任，则多租户环境被视为 **恶意**。

恶意多租户集群在所有 Kubernetes 环境中都引入安全问题。额外的安全功能，如 [Pod 安全策略](#)，或节点的更精细的访问控制 (RBAC)，使利用安全漏洞进行工具更困难。但是，如果您选择运行托管多租户工作负载，则管理程序是唯一应使用的安全选项。Kubernetes 的安全域包括整个集群，而不是只限于单个节点。对于可能会有恶意的多租户工作负载，应该使用物理隔离的集群。

Windows 服务器容器使用共享内核提供资源隔离，但它并不适用于托管可能会有恶意的多租户工作负载。涉及主机多租户的情况应该使用 Hyper-V 隔离容器来严格隔离租户。

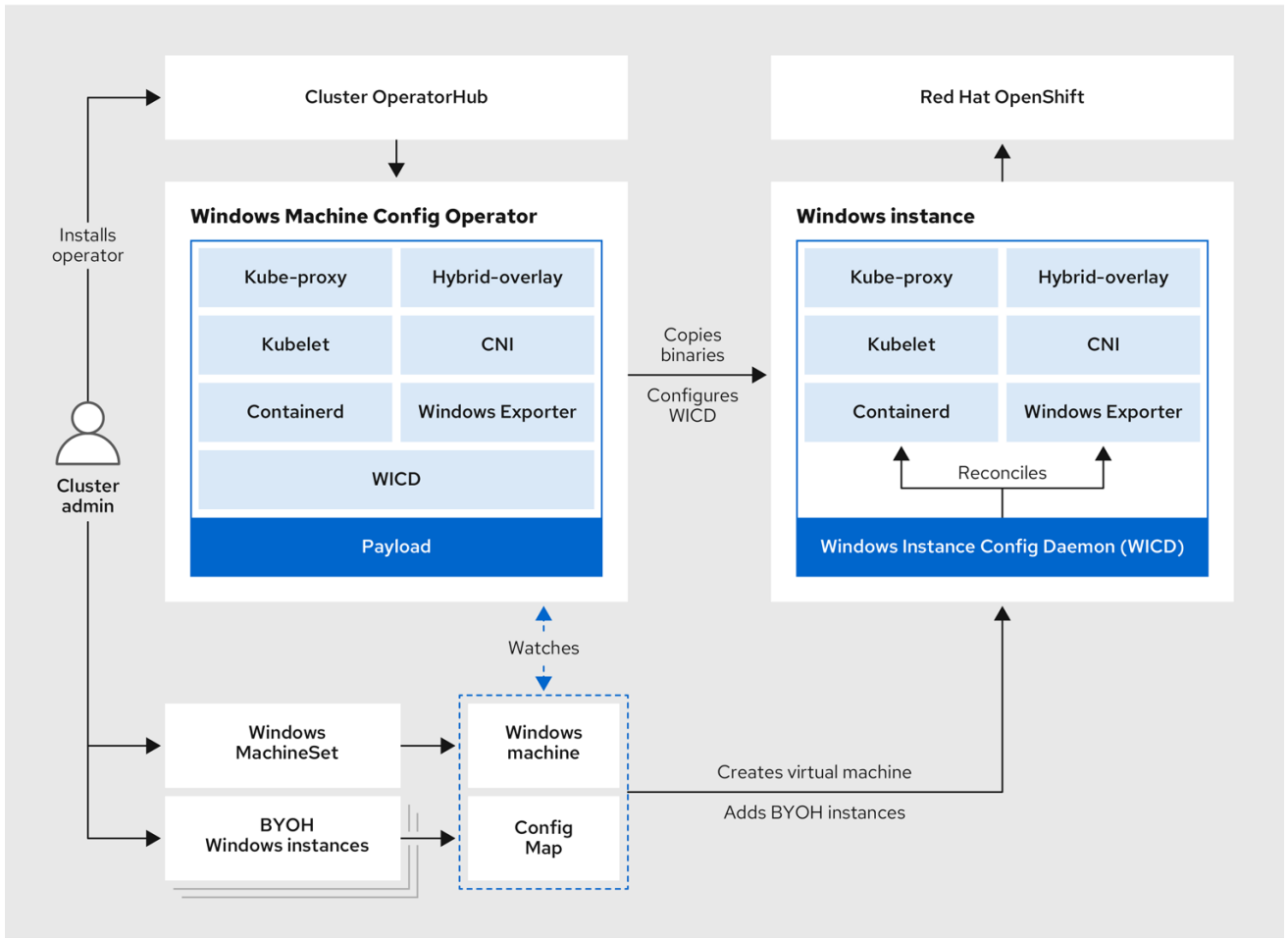
### 其他资源

- [请参阅使用 OVN-Kubernetes 配置混合网络](#)

## 4.1. WINDOWS 工作负载管理

要在集群中运行 Windows 工作负载，必须首先安装 Windows Machine Config Operator (WMCO)。WMCO 是一个基于 Linux 的 Operator，它运行在基于 Linux 的 control plane 和计算节点上。WMCO 在集群中管理部署和管理 Windows 工作负载的过程。

图 4.1. WMCO 设计



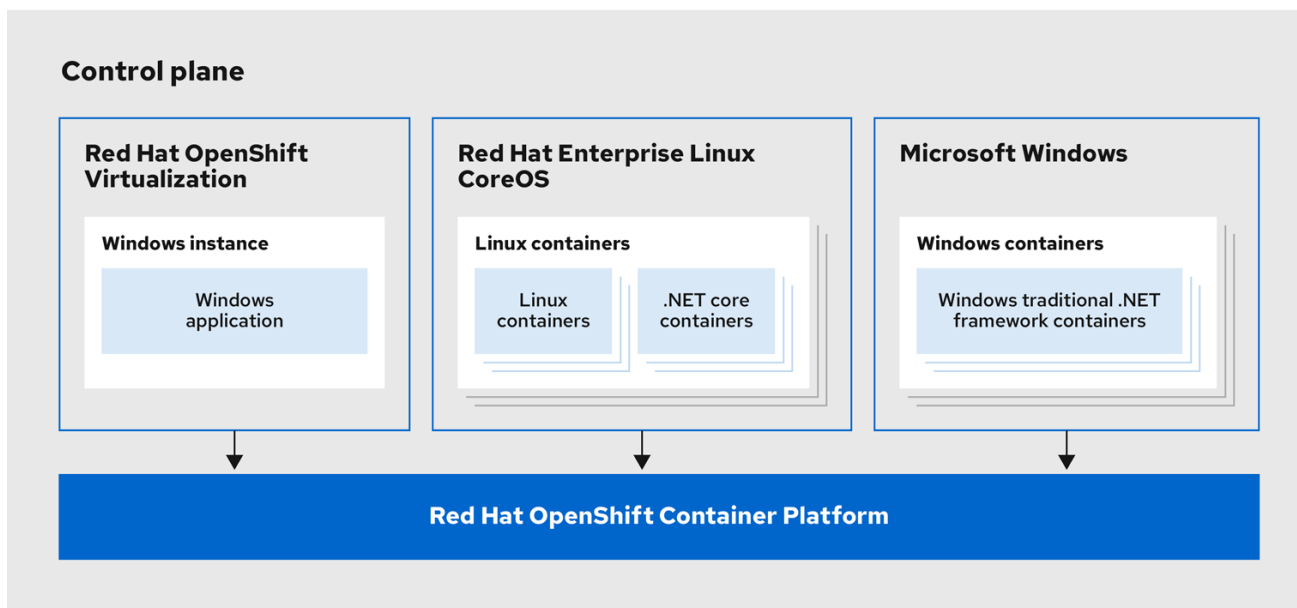
293\_OpenShift\_1122

在部署 Windows 工作负载前，您必须创建一个 Windows 计算节点并加入集群。Windows 节点会在集群中托管 Windows 工作负载，并可与其他基于 Linux 的计算节点一起运行。您可以通过创建一个 Windows 计算机器设置为托管 Windows Server 计算机器来创建 Windows 计算节点。您必须将特定于 Windows 的标签应用到指定 Windows OS 镜像的计算机器集中。

WMCO 监视有 Windows 标签的机器。在检测到 Windows 计算机器并设置并置备相应机器后，WMCO 配置底层 Windows 虚拟机(VM)，以便它可以为集群添加计算节点。



图 4.2. 混合 Windows 和 Linux 工作负载



293\_OpenShift\_1122

WMCO 在命名空间中需要一个预先确定的 secret，该 secret 包含一个用于与 Windows 实例交互的私钥。WMCO 在引导时检查此 secret，并创建一个用户数据 secret，您必须在您创建的 Windows **MachineSet** 对象中引用该 secret。然后，WMCO 使用与私钥对应的公钥填充用户数据 secret。通过这些数据，集群可以使用 SSH 连接到 Windows 虚拟机。

当集群与 Windows 虚拟机建立连接后，您可以使用类似管理 Linux 节点一样的方法管理 Windows 节点。



### 注意

OpenShift Container Platform Web 控制台为 Linux 节点可用的 Windows 节点提供大多数相同的监控功能。但是，目前无法监控 Windows 节点上运行的 pod 的工作负载图形。

将 Windows 工作负载调度到 Windows 节点可使用典型的 pod 调度实践，如污点、容限和节点选择器。或者，您也可以使用 **RuntimeClass** 对象来把 Windows 工作负载与 Linux 工作负载和其他 Windows 版本工作负载进行区分。

## 4.2. WINDOWS 节点服务

每个 Windows 节点均安装以下与 Windows 相关的服务：

Service	描述
kubelet	注册 Windows 节点并管理其状态。
Container Network Interface (CNI) 插件	为 Windows 节点公开网络。
Windows 实例配置守护进程 (WICD)	维护在 Windows 实例上运行的所有服务的状态，以确保实例充当 worker 节点。

Service	描述
<a href="#">Windows Exporter</a>	从 Windows 节点导出 Prometheus 指标
<a href="#">Kubernetes Cloud Controller Manager (CCM)</a>	与底层 Azure 云平台交互。
hybrid-overlay	创建 OpenShift Container Platform <a href="#">主机网络服务 (HNS)</a> 。
kube-proxy	在节点上维护网络规则，以允许外部通信。
容器运行时	管理完整的容器生命周期。

## 第 5 章 启用 WINDOWS 容器工作负载

在向集群中添加 Windows 工作负载前，您必须安装由 OpenShift Container Platform OperatorHub 提供的 Windows Machine Config Operator (WMCO)。WMCO 在集群中管理部署和管理 Windows 工作负载的过程。



### 注意

WMCO 管理的 Windows 实例不支持双 NIC。

### 先决条件

- 可以使用具有 **cluster-admin** 权限的账户访问 OpenShift Container Platform 集群。
- 已安装 OpenShift CLI (**oc**)。
- 已使用安装程序置备的基础架构安装集群，或使用用户置备的基础架构在 **install-config.yaml** 文件中设置了 **platform: none** 字段。
- 您已为集群配置了带有 OVN-Kubernetes 的混合网络。这必须在集群安装过程中完成。如需更多信息，[请参阅配置混合网络](#)。
- 运行 OpenShift Container Platform 集群版本 4.6.8 或更高版本。



### 注意

WMCO 部署的 Windows 实例使用容器运行时配置。因为 WMCO 安装和管理运行时，所以请不要在节点上手动安装容器。

### 其他资源

- 有关 Windows Machine Config Operator 的完整先决条件，[请参阅 Windows Machine Config Operator 先决条件](#)。

## 5.1. 安装 WINDOWS MACHINE CONFIG OPERATOR

您可以使用 Web 控制台或 OpenShift CLI (**oc**) 安装 Windows Machine Config Operator。

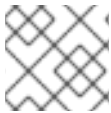


### 注意

- [使用集群范围代理](#) 的集群中不支持 WMCO，因为 WMCO 无法通过工作负载的代理连接路由流量。
- 由于 Windows 操作系统中的一个限制，类 E 的 **clusterNetwork** CIDR 地址（如 **240.0.0.0**）与 Windows 节点不兼容。

### 5.1.1. 使用 Web 控制台安装 Windows Machine Config Operator

您可以使用 OpenShift Container Platform Web 控制台安装 Windows Machine Config Operator (WMCO)。

**注意**

WMCO 管理的 Windows 实例不支持双 NIC。

**流程**

1. 从 OpenShift Container Platform Web 控制台中的 **Administrator** 视角进入 **Operators** → **OperatorHub** 页面。
2. 使用 **Filter by keyword** 复选框在目录中搜索 **Windows Machine Config Operator**。点 **Windows Machine Config Operator** 标题。
3. 查看 Operator 信息并点 **Install**。
4. 在 **Install Operator** 页面中：
  - a. 选择 **stable** 频道作为 **更新频道**。**stable** 频道允许安装 WMCO 的最新稳定版本。
  - b. **安装模式** 被预先配置，因为 WMCO 只能在单一命名空间中可用。
  - c. 为 WMCO 选择 **Installed Namespace**。默认 Operator 建议命名空间为 **openshift-windows-machine-config-operator**。
  - d. 点 **Enable Operator recommended cluster monitoring on the Namespace** 为 WMCO 启用集群监控。
  - e. 选择一个 **批准策略**。
    - **Automatic** 策略允许 Operator Lifecycle Manager (OLM) 在有新版本可用时自动更新 Operator。
    - **Manual** 策略需要拥有适当凭证的用户批准 Operator 更新。
1. 点 **Install**。WMCO 现在列在 **Installed Operators** 页中。

**注意**

WMCO 会自动安装到您定义的命名空间中，如 **openshift-windows-machine-config-operator**。

2. 验证 **Status** 显示为 **Succeeded** 以确认成功安装了 WMCO。

**5.1.2. 使用 CLI 安装 Windows Machine Config Operator**

您可以使用 OpenShift CLI (**oc**) 安装 Windows Machine Config Operator (WMCO)。

**注意**

WMCO 管理的 Windows 实例不支持双 NIC。

**流程**

1. 为 WMCO 创建命名空间。
  - a. 为 WMCO 创建一个 **Namespace** 对象 YAML 文件。例如，**wmco-namespace.yaml**：

■

```

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

```

- ❶ 建议您在 **openshift-windows-machine-config-operator** 命名空间中部署 WMCO。
- ❷ 为 WMCO 启用集群监控需要此标签。

b. 创建命名空间：

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

例如：

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

2. 为 WMCO 创建 Operator 组。

a. 创建 **OperatorGroup** 对象 YAML 文件。例如，**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. 创建 Operator 组：

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

例如：

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

3. 为 WMCO 订阅命名空间。

a. 创建 **Subscription** 对象 YAML 文件。例如，**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" 3
sourceNamespace: "openshift-marketplace" 4
```

- 1 指定 **stable** 作为频道。
- 2 设置批准策略。可设置 **Automatic** 或 **Manual**。
- 3 指定包含 **windows-machine-config-operator** 软件包清单的 **redhat-operators** 目录源。如果 OpenShift Container Platform 集群安装在受限网络中（也称为断开连接的集群），请指定配置 Operator Lifecycle Manager (OLM) 时创建的 **CatalogSource** 对象的名称。
- 4 目录源的命名空间。将 **openshift-marketplace** 用于默认的 OperatorHub 目录源。

b. 创建订阅：

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

例如：

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

WMCO 现已安装到 **openshift-windows-machine-config-operator** 中。

4. 验证 WMCO 安装：

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

输出示例

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

## 5.2. 为 WINDOWS MACHINE CONFIG OPERATOR 配置 SECRET

要运行 Windows Machine Config Operator (WMCO)，您必须在包含私钥的 WMCO 命名空间中创建一个 secret。这需要允许 WMCO 与 Windows 虚拟机 (VM) 进行通信。

先决条件

- 已使用 Operator Lifecycle Manager (OLM) 安装 Windows Machine Config Operator (WMCO)。
- 您创建包含 RSA 密钥的 PEM 编码文件。

流程

- 定义访问 Windows 虚拟机所需的 secret:

```
$ oc create secret generic cloud-private-key --from-file=private-  
key.pem=${HOME}/.ssh/<key> \  
-n openshift-windows-machine-config-operator 1
```

- 1 您必须在 WMCO 命名空间中创建私钥，如 **openshift-windows-machine-config-operator**。

建议您使用与安装集群时所用的私钥不同的私钥。

### 5.3. 在支持代理的集群中使用 WINDOWS 容器

在向集群的内部网络外发出外部请求时，Windows Machine Config Operator (WMCO) 可以使用集群范围的出口代理配置。

这可让您在支持代理的集群中添加 Windows 节点并运行工作负载，允许 Windows 节点从代理服务器后面的 registry 中拉取镜像，或者向使用自定义公钥基础架构的非集群服务和 service 发出请求。



#### 注意

集群范围的代理仅影响系统组件，而不会影响用户工作负载。

在启用了代理的集群中，WMCO 知道为集群设置的 **NO\_PROXY**、**HTTP\_PROXY** 和 **HTTPS\_PROXY** 值。WMCO 定期检查代理环境变量是否已更改。如果存在差异，WMCO 会协调并更新 Windows 实例上的代理环境变量。

在支持代理的集群的 Windows 节点上创建的 Windows 工作负载不会继承节点上的代理设置，与 Linux 节点相同。另外，默认情况下，PowerShell 会话不会在启用了代理的集群中继承 Windows 节点上的代理设置。

#### 其他资源

- [配置集群范围代理](#).

### 5.4. 其他资源

- [为集群节点 SSH 访问生成密钥对](#)
- [将 Operator 添加到集群](#).

## 第 6 章 创建 WINDOWS 机器集

### 6.1. 在 AWS 上创建 WINDOWS 机器集

您可以在 Amazon Web Services (AWS) 上的 OpenShift Container Platform 集群中创建 Windows **MachineSet** 对象来满足特定目的。例如，您可以创建基础架构 Windows 机器集和相关机器，以便将支持的 Windows 工作负载转移到新的 Windows 机器上。

#### 先决条件

- 已使用 Operator Lifecycle Manager (OLM) 安装 Windows Machine Config Operator (WMCO)。
- 您可以使用受支持的 Windows Server 作为操作系统镜像。  
根据您的 Windows Server 发行版本，使用以下 **aws** 命令之一来查询有效的 AMI 镜像：

#### Windows Server 2022 命令示例

```
$ aws ec2 describe-images --region <aws_region_name> --filters
"Name=name,Values=Windows_Server-2022*English*Core*Base*" "Name=is-
public,Values=true" --query "reverse(sort_by(Images, &CreationDate))[*].{name: Name, id:
ImageId}" --output table
```

#### Windows Server 2019 命令示例

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

其中：

**<aws\_region\_name>**

指定 AWS 区域的名称。

#### 6.1.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.12 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。由于此系统的缘故，OpenShift Container Platform 4.12 在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

#### Machines

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，计算节点的机器类型可能会定义特定的机器类型和所需的元数据。

#### 机器集

**MachineSet** 资源是计算机器组。计算机器集适用于计算机器，因为副本集是针对 pod。如果需要更多计算机器或必须缩减规模，您可以更改 **MachineSet** 资源的 **replicas** 字段来满足您的计算需求。





### 警告

control plane 机器不能由计算机器集管理。

control plane 机器集为 control plane 机器提供管理功能，与为计算机器集提供的计算机器集类似。

如需更多信息，请参阅“管理 control plane 机器”。

以下自定义资源可为集群添加更多功能：

### 机器自动扩展

**MachineAutoscaler** 资源自动扩展云中的计算机器。您可以为指定计算机器集中的节点设置最小和最大扩展界限，机器自动扩展则维护该范围内的节点。

**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

### 集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展计算机器设置 API 来与 Machine API 集成。您可以使用以下方法使用集群自动扩展来管理集群：

- 为内核、节点、内存和 GPU 等资源设置集群范围的扩展限制
- 设置优先级，以便集群对 pod 和新节点进行优先排序，而在不太重要的 pod 时不会上线
- 设置扩展策略，以便您可以扩展节点，但不会缩减节点

### 机器健康检查

**MachineHealthCheck** 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个计算机器集限定在一个区，因此安装程序可以代表您的可用区向计算机器集发送。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。自动扩展器在集群生命周期内尽可能提供平衡。

## 6.1.2. AWS 上 Windows MachineSet 对象的 YAML 示例

此 YAML 示例定义了一个在 Amazon Web Services (AWS) 上运行的 Windows **MachineSet** 对象，它可响应 Windows Machine Config Operator (WMCO)。

```
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** 指定基于置备集群时所设置的集群 ID 的基础架构 ID。您可以运行以下命令来获取基础架构 ID:

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

- 2 4 6** 指定基础架构 ID、worker 标签和区。

- 7** 将计算机器设置为 Windows 机器。

- 8** 将 Windows 节点配置为计算机器。

- 9** 指定安装了容器运行时的受支持的 Windows 镜像的 AMI ID。

- 11** 指定 AWS 区域，如 **us-east-1a**。

- 12** 指定 AWS 区域，如 **us-east-1**。

- 16** 由 WMCO 配置第一个 Windows 机器时创建的。之后，所有后续计算机器集都可以使用 **windows-user-data**。

### 6.1.3. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的来动态管理您选择的特定工作负载的机器计算资源。

#### 先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI (**oc**) 。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

#### 流程

1. 创建一个包含计算机器集自定义资源 (CR) 示例的新 YAML 文件，并将其命名为 **<file\_name>.yaml**。  
确保设置 **<clusterID>** 和 **<role>** 参数值。
2. 可选：如果您不确定要为特定字段设置哪个值，您可以从集群中检查现有计算机器集：
  - a. 要列出集群中的计算机器集，请运行以下命令：

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

#### 输出示例

```

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

```

- b. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

```

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

```

### 输出示例

```

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 集群基础架构 ID。

❷ 默认节点标签。



### 注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

❸ 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

```

$ oc create -f <file_name>.yaml

```

## 验证

- 运行以下命令，查看计算机器集列表：

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

### 输出示例

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	

当新的计算机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果 compute 机器集不可用，请等待几分钟，然后再次运行命令。

### 6.1.4. 其他资源

- [机器管理概述](#)

## 6.2. 在 AZURE 上创建 WINDOWS 机器集

您可以在 Microsoft Azure 上的 OpenShift Container Platform 集群中创建 Windows **MachineSet** 对象来满足特定目的。例如，您可以创建基础架构 Windows 机器集和相关机器，以便将支持的 Windows 工作负载转移到新的 Windows 机器上。

### 先决条件

- 已使用 Operator Lifecycle Manager (OLM) 安装 Windows Machine Config Operator (WMCO)。
- 您可以使用受支持的 Windows Server 作为操作系统镜像。

### 6.2.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.12 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。由于此系统的缘故，OpenShift Container Platform 4.12 在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

#### Machines

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，计算节点的机器类型可能会定义特定的机器类型和所需的元数据。

#### 机器集

**MachineSet** 资源是计算机器组。计算机器集适用于计算机器，因为副本集是针对 pod。如果需要更多计算机器或必须缩减规模，您可以更改 **MachineSet** 资源的 **replicas** 字段来满足您的计算需求。



### 警告

control plane 机器不能由计算机器集管理。

control plane 机器集为 control plane 机器提供管理功能，与为计算机器提供的计算机器集类似。

如需更多信息，请参阅“管理 control plane 机器”。

以下自定义资源可为集群添加更多功能：

### 机器自动扩展

**MachineAutoscaler** 资源自动扩展云中的计算机器。您可以为指定计算机器集中的节点设置最小和最大扩展界限，机器自动扩展则维护该范围内的节点。

**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

### 集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展计算机器设置 API 来与 Machine API 集成。您可以使用以下方法使用集群自动扩展来管理集群：

- 为内核、节点、内存和 GPU 等资源设置集群范围的扩展限制
- 设置优先级，以便集群对 pod 和新节点进行优先排序，而在不太重要的 pod 时不会上线
- 设置扩展策略，以便您可以扩展节点，但不会缩减节点

### 机器健康检查

**MachineHealthCheck** 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个计算机器集限定在一个区，因此安装程序可以代表您的可用区向计算机器集发送。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。自动扩展器在集群生命周期内尽可能提供平衡。

## 6.2.2. Azure 上 Windows MachineSet 对象的 YAML 示例

此 YAML 示例定义了一个在 Microsoft Azure 上运行的 Windows **MachineSet** 对象，Windows Machine Config Operator (WMCO) 可以响应。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
```

```

labels:
  machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
name: <windows_machine_set_name> 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: <windows_machine_set_name> 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: <windows_machine_set_name> 6
      machine.openshift.io/os-id: Windows 7
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/worker: "" 8
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image: 9
        offer: WindowsServer
        publisher: MicrosoftWindowsServer
        resourceID: ""
        sku: 2019-Datacenter-with-Containers
        version: latest
        kind: AzureMachineProviderSpec
        location: <location> 10
        managedIdentity: <infrastructure_id>-identity 11
        networkResourceGroup: <infrastructure_id>-rg 12
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Windows
        publicIP: false
        resourceGroup: <infrastructure_id>-rg 13
        subnet: <infrastructure_id>-worker-subnet
        userDataSecret:
          name: windows-user-data 14
          namespace: openshift-machine-api
        vmSize: Standard_D2s_v3
        vnet: <infrastructure_id>-vnet 15
        zone: "<zone>" 16

```

1 3 5 11 12 13 15 指定基于置备集群时所设置的集群 ID 的基础架构 ID。您可以运行以下命令来获取

基础架构 ID:

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

- 2 4 6 指定 Windows 计算机设置名称。Azure 上的 Windows 机器名称不能超过 15 个字符。因此，因为机器名称的生成方式，计算机设置名称不能超过 9 个字符。
- 7 将计算机设置为 Windows 机器。
- 8 将 Windows 节点配置为计算机。
- 9 指定一个 **WindowsServer** 镜像，它定义了 **2019-Datacenter-with-Containers** SKU。
- 10 指定 Azure 区域，如 **centralus**。
- 14 由 WMCO 配置第一个 Windows 机器时创建的。之后，所有后续计算机集都可以使用 **windows-user-data**。
- 16 指定您所在地区（region）内要放置机器的区域（zone）。确保您的地区支持您指定的区域。

### 6.2.3. 创建计算机集

除了安装程序创建的计算机集外，您还可以创建自己的来动态管理您选择的特定工作负载的机器计算资源。

#### 先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI (**oc**) 。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

#### 流程

1. 创建一个包含计算机集自定义资源（CR）示例的新 YAML 文件，并将其命名为 **<file\_name>.yaml**。  
确保设置 **<clusterID>** 和 **<role>** 参数值。
2. 可选：如果您不确定要为特定字段设置哪个值，您可以从集群中检查现有计算机集：
  - a. 要列出集群中的计算机集，请运行以下命令：

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

#### 输出示例

```
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        0      0          55m
agl030519-vplxk-worker-us-east-1e  0        0        0      0          55m
agl030519-vplxk-worker-us-east-1f  0        0        0      0          55m
```



- b. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

### 输出示例

```
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** 集群基础架构 ID。

**2** 默认节点标签。



### 注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

**3** 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

### 验证

- 运行以下命令，查看计算机器集列表：

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

## 输出示例

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	

当新的计算机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果 compute 机器集不可用，请等待几分钟，然后再次运行命令。

### 6.2.4. 其他资源

- [机器管理概述](#)

## 6.3. 在 VSPHERE 上创建 WINDOWS 机器集

您可以在 VMware vSphere 上的 OpenShift Container Platform 集群中创建 Windows **MachineSet** 对象来满足特定目的。例如，您可以创建基础架构 Windows 机器集和相关机器，以便将支持的 Windows 工作负载转移到新的 Windows 机器上。

### 先决条件

- 已使用 Operator Lifecycle Manager (OLM) 安装 Windows Machine Config Operator (WMCO)。
- 您可以使用受支持的 Windows Server 作为操作系统镜像。

### 6.3.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.12 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。由于此系统的缘故，OpenShift Container Platform 4.12 在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

#### Machines

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，计算节点的机器类型可能会定义特定的机器类型和所需的元数据。

#### 机器集

**MachineSet** 资源是计算机器组。计算机器集适用于计算机器，因为副本集是针对 pod。如果需要更多计算机器或必须缩减规模，您可以更改 **MachineSet** 资源的 **replicas** 字段来满足您的计算需求。



### 警告

control plane 机器不能由计算机器集管理。

control plane 机器集为 control plane 机器提供管理功能，与为计算机器集提供的计算机器集类似。

如需更多信息，请参阅“管理 control plane 机器”。

以下自定义资源可为集群添加更多功能：

#### 机器自动扩展

**MachineAutoscaler** 资源自动扩展云中的计算机器。您可以为指定计算机器集中的节点设置最小和最大扩展界限，机器自动扩展则维护该范围内的节点。

**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

#### 集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展计算机器设置 API 来与 Machine API 集成。您可以使用以下方法使用集群自动扩展来管理集群：

- 为内核、节点、内存和 GPU 等资源设置集群范围的扩展限制
- 设置优先级，以便集群对 pod 和新节点进行优先排序，而在不太重要的 pod 时不会上线
- 设置扩展策略，以便您可以扩展节点，但不会缩减节点

#### 机器健康检查

**MachineHealthCheck** 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个计算机器集限定在一个区，因此安装程序可以代表您的可用区向计算机器集发送。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。自动扩展器在集群生命周期内尽可能提供平衡。

### 6.3.2. 为 Windows 容器工作负载准备 vSphere 环境

您必须通过创建 vSphere Windows VM 金级镜像并启用与 WMCO 的内部 API 服务器通信来为 Windows 容器工作负载准备 vSphere 环境。

#### 6.3.2.1. 创建 vSphere Windows 虚拟机金级镜像

创建 vSphere Windows 虚拟机 (VM) 金级镜像。

#### 先决条件

- 您已创建了私钥/公钥对，用于在 OpenSSH 服务器中配置基于密钥的身份验证。还必须在 Windows Machine Config Operator (WMCO) 命名空间中配置私钥。这需要允许 WMCO 与 Windows 虚拟机进行通信。如需了解更多详细信息，请参阅“为 Windows Machine Config Operator 配置 secret”部分。



### 注意

在创建 Windows 虚拟机的多个情形中，您必须使用 [Microsoft PowerShell](#) 命令。本指南中的 PowerShell 命令由 **PS C:\>** 前缀区分。

### 流程

1. 选择兼容的 Windows Server 版本。目前，Windows Machine Config Operator (WMCO) 稳定版本支持 Windows Server 2022 Long-Term Servicing Channel 及 OS-level 容器网络补丁 [KB5012637](#)。
2. 使用带有兼容 Windows Server 版本的 VM 金级镜像，在 vSphere 客户端中创建新虚拟机。有关兼容版本的更多信息，请参阅 Red Hat OpenShift support for Windows Containers 发行注册中的“Windows Machine Config Operator prerequisites”部分



### 重要

您的虚拟机的虚拟硬件版本必须满足 OpenShift Container Platform 的基础架构要求。如需更多信息，请参阅 OpenShift Container Platform 文档中的“VMware vSphere infrastructure requirements”部分。另外，您还可以参考 VMware 有关 [虚拟机硬件版本的文档](#)。

3. 在 Windows 虚拟机上安装和配置 VMware Tools 版本 11.0.6 或更高版本。如需更多信息，参阅 [VMware Tools 文档](#)。
4. 在 Windows 虚拟机上安装了 VMware Tools 后，请验证以下内容：

- a. The **C:\ProgramData\VMware\VMware Tools\tools.conf** 文件存在，并带有以下条目：

```
exclude-nics=
```

如果 **tools.conf** 文件不存在，则使用 **exclude-nics** 选项取消注释并设置为空值创建该文件。

此条目确保通过混合覆盖在 Windows 虚拟机上生成的克隆 vNIC 不被忽略。

- b. Windows 虚拟机在 vCenter 中有一个有效的 IP 地址：

```
C:\> ipconfig
```

- c. VMTools Windows 服务正在运行：

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

5. 在 Windows 虚拟机上安装和配置 OpenSSH 服务器。如需了解更多详细信息，请参阅 Microsoft 的有关 [安装 OpenSSH](#) 的文档。
6. 为管理用户设置 SSH 访问权限。详情请参阅 Microsoft 文档的 [管理员用户](#)。



## 重要

指令中使用的公钥必须与稍后在 WMCO 命名空间中创建的私钥对应，后者包含您的 secret。如需了解更多详细信息，请参阅“为 Windows Machine Config Operator 配置 secret”部分。

- 您必须在 Windows 虚拟机中创建新的防火墙规则，以允许进入连接的容器日志。运行以下命令，在 TCP 端口 10250 中创建防火墙规则：

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

- 克隆 Windows 虚拟机，使其成为可重复利用的镜像。遵循 VMware 文档来了解如何[克隆现有虚拟机](#)以了解更多详细信息。
- 在克隆的 Windows 虚拟机上，运行 [Windows Sysprep 工具](#)：

```
C:\> C:\Windows\System32\Sysprep\sysprep.exe /generalize /oobe /shutdown /unattend:
<path_to_unattend.xml> 1
```

- 指定 **unattend.xml** 文件的路径。



## 注意

对于您可以在 Windows 镜像中运行 **sysprep** 命令的次数有一个限制。如需更多信息，请参阅 Microsoft [文档](#)。

提供了 **unattend.xml** 示例，它维护 WMCO 所需的所有更改。您必须修改这个示例，它不能直接使用。

### 例 6.1. unattend.xml 示例

```
<?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> 1
</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> 2
  </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>
  <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 指定 **ComputerName**，它必须遵循 [Kubernetes 名称规格](#)。这些规格也适用于在创建新虚拟机时在生成的模板中执行的 Guest OS 自定义。
- 2 禁用自动登录，以避免在启动时使用管理员特权打开终端的安全问题。这是默认值，不得更改。
- 3 将 **MyPassword** 占位符替换为 Administrator 帐户的密码。这可防止内置的 Administrator 帐户默认拥有空白密码。遵循 [Microsoft 选择密码的最佳实践](#)。

Sysprep 工具完成后，Windows 虚拟机将关闭。您不得再使用此虚拟机或打开此虚拟机。

10. 将 Windows 虚拟机转换为 vCenter 中的模板。

### 6.3.2.1.1. 其他资源

- [为 Windows Machine Config Operator 配置 secret](#)
- [VMware vSphere 基础架构要求](#)

### 6.3.2.2. 在 vSphere 上启用与 WMCO 的内部 API 服务器通信

Windows Machine Config Operator (WMCO) 从内部 API 服务器端点下载 Ignition 配置文件。您必须启用与内部 API 服务器通信，以便您的 Windows 虚拟机可以下载 Ignition 配置文件，而配置的虚拟机上的 kubelet 只能与内部 API 服务器通信。

#### 先决条件

- 您已在 vSphere 上安装了集群。

#### 流程

- 为 `api-int.<cluster_name>.<base_domain>` 添加一个新的 DNS 项，它指向外部 API 服务器 URL `api.<cluster_name>.<base_domain>`。这可以是一个 CNAME 或一个 A 记录。



#### 注意

外部 API 端点已创建，作为 vSphere 上初始集群安装的一部分。

### 6.3.3. vSphere 上 Windows MachineSet 对象的 YAML 示例

此 YAML 示例定义了一个在 VMware vSphere 上运行的 Windows **MachineSet** 对象，Windows Machine Config Operator (WMCO) 可响应。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <windows_machine_set_name> 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: <windows_machine_set_name> 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: <windows_machine_set_name> 6
        machine.openshift.io/os-id: Windows 7
```

```

spec:
  metadata:
    labels:
      node-role.kubernetes.io/worker: "" 8
  providerSpec:
    value:
      apiVersion: vsphereprovider.openshift.io/v1beta1
      credentialsSecret:
        name: vsphere-cloud-credentials
      diskGiB: 128 9
      kind: VSphereMachineProviderSpec
      memoryMiB: 16384
      network:
        devices:
          - networkName: "<vm_network_name>" 10
      numCPUs: 4
      numCoresPerSocket: 1
      snapshot: ""
      template: <windows_vm_template_name> 11
      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 指定基于置备集群时所设置的集群 ID 的基础架构 ID。您可以运行以下命令来获取基础架构 ID:

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

2 4 6 指定 Windows 计算机设置名称。由于 vSphere 中生成机器名称的方式，计算机设置的名称不能超过 9 个字符。

7 将计算机设置为 Windows 机器。

8 将 Windows 节点配置为计算机。

9 指定 vSphere Virtual Machine Disk (VMDK) 的大小。



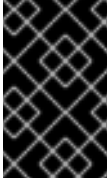
### 注意

这个参数不会设置 Windows 分区的大小。您可以使用 **unattend.xml** 文件或创建带有所需磁盘大小的 vSphere Windows 虚拟机 (VM) 金级镜像来重新定义 Windows 分区大小。

10 指定要将计算机设置为的 vSphere VM 网络。此虚拟机网络必须是集群中其他 Linux 计算机所处的位置。

11 指定要使用的 Windows vSphere 虚拟机模板的完整路径，如 **golden-images/windows-server-template**。名称必须是唯一的。





### 重要

不要指定原始虚拟机模板。VM 模板必须保持关闭，必须为新的 Windows 机器克隆。启动虚拟机模板会将虚拟机模板配置为平台上的虚拟机，这样可防止它被用作计算机器集可应用配置的模板。

- 12 当配置了第一个 Windows 机器时，**windows-user-data** 由 WMCO 创建。之后，所有后续计算机器集都可以使用 **windows-user-data**。
- 13 指定要将计算机器设置为的 vCenter Datacenter。
- 14 指定要部署计算机器的 vCenter Datastore。
- 15 指定 vCenter 中 vSphere 虚拟机文件夹的路径，如 **/dc1/vm/user-inst-5ddjd**。
- 16 可选：为您的 Windows 虚拟机指定 vSphere 资源池。
- 17 指定 vCenter 服务器 IP 或完全限定域名。

### 6.3.4. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的来动态管理您选择的特定工作负载的机器计算资源。

#### 先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI (**oc**)。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

#### 流程

1. 创建一个包含计算机器集自定义资源 (CR) 示例的新 YAML 文件，并将其命名为 **<file\_name>.yaml**。  
确保设置 **<clusterID>** 和 **<role>** 参数值。
2. 可选：如果您不确定要为特定字段设置哪个值，您可以从集群中检查现有计算机器集：
  - a. 要列出集群中的计算机器集，请运行以下命令：

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

#### 输出示例

```
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        0      0          55m
agl030519-vplxk-worker-us-east-1e  0        0        0      0          55m
agl030519-vplxk-worker-us-east-1f  0        0        0      0          55m
```

- b. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

### 输出示例

```
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** 集群基础架构 ID。

**2** 默认节点标签。



### 注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

**3** 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

### 验证

- 运行以下命令，查看计算机器集列表：

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

## 输出示例

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

当新的计算机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果 compute 机器集不可用，请等待几分钟，然后再次运行命令。

### 6.3.5. 其他资源

- [机器管理概述](#)

## 6.4. 在 GCP 上创建 WINDOWS 机器集

您可以在 Google Cloud Platform (GCP) 上的 OpenShift Container Platform 集群中创建 Windows **MachineSet** 对象来满足特定目的。例如，您可以创建基础架构 Windows 机器集和相关机器，以便将支持的 Windows 工作负载转移到新的 Windows 机器上。

### 先决条件

- 已使用 Operator Lifecycle Manager (OLM) 安装 Windows Machine Config Operator (WMCO)。
- 您可以使用受支持的 Windows Server 作为操作系统镜像。

### 6.4.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.12 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。由于此系统的缘故，OpenShift Container Platform 4.12 在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

#### Machines

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，计算节点的机器类型可能会定义特定的机器类型和所需的元数据。

#### 机器集

**MachineSet** 资源是计算机器组。计算机器集适用于计算机器，因为副本集是针对 pod。如果需要更多计算机器或必须缩减规模，您可以更改 **MachineSet** 资源的 **replicas** 字段来满足您的计算需求。



### 警告

control plane 机器不能由计算机器集管理。

control plane 机器集为 control plane 机器提供管理功能，与为计算机器集提供的计算机器集类似。

如需更多信息，请参阅“管理 control plane 机器”。

以下自定义资源可为集群添加更多功能：

### 机器自动扩展

**MachineAutoscaler** 资源自动扩展云中的计算机器。您可以为指定计算机器集中的节点设置最小和最大扩展界限，机器自动扩展则维护该范围内的节点。

**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

### 集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展计算机器设置 API 来与 Machine API 集成。您可以使用以下方法使用集群自动扩展来管理集群：

- 为内核、节点、内存和 GPU 等资源设置集群范围的扩展限制
- 设置优先级，以便集群对 pod 和新节点进行优先排序，而在不太重要的 pod 时不会上线
- 设置扩展策略，以便您可以扩展节点，但不会缩减节点

### 机器健康检查

**MachineHealthCheck** 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个计算机器集限定在一个区，因此安装程序可以代表您的可用区向计算机器集发送。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。自动扩展器在集群生命周期内尽可能提供平衡。

## 6.4.2. GCP 上 Windows MachineSet 对象的 YAML 示例

此 YAML 文件示例定义了一个在 Google Cloud Platform (GCP) 上运行的 Windows **MachineSet** 对象，Windows Machine Config Operator (WMCO) 可以使用它。

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-windows-worker-<zone_suffix> 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_suffix>
  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_suffix>
  6
      machine.openshift.io/os-id: Windows 7
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/worker: "" 8
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          canIPForward: false
          credentialsSecret:
            name: gcp-cloud-credentials
          deletionProtection: false
          disks:
            - autoDelete: true
              boot: true
              image: <windows_server_image> 9
              sizeGb: 128
              type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          networkInterfaces:
            - network: <infrastructure_id>-network 10
              subnet: <infrastructure_id>-worker-subnet
          projectID: <project_id> 11
          region: <region> 12
          serviceAccounts:
            - email: <infrastructure_id>-w@<project_id>.iam.gserviceaccount.com
              scopes:
                - https://www.googleapis.com/auth/cloud-platform
          tags:
            - <infrastructure_id>-worker
          userDataSecret:
            name: windows-user-data 13
          zone: <zone> 14

```

1 3 5 10 指定基于置备集群时所设置的集群 ID 的基础架构 ID。您可以运行以下命令来获取基础架构 ID:

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

- 2 4 6 指定基础架构 ID、worker 标签和区域后缀（如 **a**）。
- 7 将机器配置为 Windows 机器。
- 8 将 Windows 节点配置为计算机。
- 9 指定支持的 Windows Server 镜像的完整路径。
- 11 指定在其中创建此集群的 GCP 项目。
- 12 指定 GCP 区域，如 **us-central1**。
- 13 由 WMCO 配置第一个 Windows 机器时创建的。之后，所有后续机器组都可以使用 **windows-user-data**。
- 14 指定所选区域中的区域，如 **us-central1-a**。

### 6.4.3. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的来动态管理您选择的特定工作负载的机器计算资源。

#### 先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI (**oc**) 。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

#### 流程

1. 创建一个包含计算机器集自定义资源 (CR) 示例的新 YAML 文件，并将其命名为 **<file\_name>.yaml**。  
确保设置 **<clusterID>** 和 **<role>** 参数值。
2. 可选：如果您不确定要为特定字段设置哪个值，您可以从集群中检查现有计算机器集：
  - a. 要列出集群中的计算机器集，请运行以下命令：

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

#### 输出示例

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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

### 输出示例

```
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** 集群基础架构 ID。

**2** 默认节点标签。



### 注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

**3** 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

### 验证

- 运行以下命令，查看计算机器集列表：

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

## 输出示例

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-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

当新的计算机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果 compute 机器集不可用，请等待几分钟，然后再次运行命令。

### 6.4.4. 其他资源

- [机器管理概述](#)



## 第 7 章 调度 WINDOWS 容器工作负载

您可以将 Windows 工作负载调度到 Windows 计算节点。



### 注意

使用[集群范围代理](#)的集群中不支持 WMCO，因为 WMCO 无法通过工作负载的代理连接路由流量。

### 先决条件

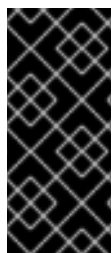
- 已使用 Operator Lifecycle Manager (OLM) 安装 Windows Machine Config Operator (WMCO)。
- 您可以使用 Windows 容器作为 OS 镜像。
- 您已创建了 Windows 计算机器集。

## 7.1. WINDOWS POD 放置

在集群中部署 Windows 工作负载前，您必须配置 Windows 节点调度，以便正确分配 pod。在有了托管 Windows 节点的机器后，就可以使用管理 Linux 节点相同的方法进行管理。同样，将 Windows pod 调度到适当的 Windows 节点会以相似的方式完成，可以使用污点、容限和节点选择器等机制。

如果在一个集群中，有多个操作系统以及运行多个 Windows OS 变体，您必须使用 **RuntimeClass** 对象将 Windows pod 映射到基本 Windows OS 变体。例如，如果您在不同 Windows Server 容器版本中运行多个 Windows 节点，集群可将 Windows pod 调度到不兼容的 Windows OS 变体。您必须为集群中的每个 Windows OS 变体配置 **RuntimeClass** 对象。如果集群中只有一个 Windows OS 变体，则建议使用 **RuntimeClass** 对象。

如需更多信息，请参阅微软有关[主机和容器版本兼容性](#)的文档。



### 重要

容器基础镜像与容器要调度到的节点的 Windows OS 版本和构建号需要相同。

另外，如果您将 Windows 节点从一个版本升级到另一个版本，例如从 20H2 升级到 2022，则需要升级容器基础镜像以匹配新版本。如需更多信息，请参阅[Windows 容器版本兼容性](#)。

### 其他资源

- [使用调度程序控制 pod 放置](#)
- [使用节点污点控制 pod 放置](#)
- [使用节点选择器将 pod 放置到特定节点](#)

## 7.2. 创建 RUNTIMECLASS 对象来封装调度机制

使用 **RuntimeClass** 对象简化了调度机制的使用，如污点和容限；您可以部署一个运行时类来封装您的污点和容限，然后将其应用到 pod，再将它们调度到适当的节点。在支持多个操作系统变体的集群中，还需要创建运行时类。

## 流程

1. 创建 **RuntimeClass** 对象 YAML 文件。例如，`runtime-class.yaml`：

```

apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: windows2019 1
handler: 'runhcs-wcow-process'
scheduling:
  nodeSelector: 2
    kubernetes.io/os: 'windows'
    kubernetes.io/arch: 'amd64'
    node.kubernetes.io/windows-build: '10.0.17763'
  tolerations: 3
    - effect: NoSchedule
      key: os
      operator: Equal
      value: "windows"
    - effect: NoSchedule
      key: os
      operator: Equal
      value: "Windows"

```

- 1** 指定 **RuntimeClass** 对象名称，该名称在您要由此运行时类管理的 pod 中定义。
- 2** 指定支持这个运行时类的节点必须存在的标签。使用此运行时类的 Pod 只能调度到与此选择器匹配的节点。运行时类的节点选择器与 pod 的现有节点选择器合并。任何冲突都会阻止 pod 调度到节点。
  - 对于 Windows 2019，指定 `node.kubernetes.io/windows-build: '10.0.17763'` 标签。
  - 对于 Windows 2022，指定 `node.kubernetes.io/windows-build: '10.0.20348'` 标签。
- 3** 指定要在 pod 附加的容限（不包括重复），在准入过程中使用此运行时类运行。这将合并 pod 和运行时类容许的节点集合。

2. 创建 **RuntimeClass** 对象：

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

例如：

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

3. 将 **RuntimeClass** 对象应用到您的 pod，以确保其被调度到适当的操作系统变体：

```

apiVersion: v1
kind: Pod
metadata:
  name: my-windows-pod
spec:
  runtimeClassName: windows2019 1
# ...

```

- 1 指定管理 pod 调度的运行时类。

## 7.3. WINDOWS 容器工作负载部署示例

在有 Windows 计算节点可用后，您可以将 Windows 容器工作负载部署到集群中。



### 注意

这个示例部署仅供参考。

### Service 对象示例

```
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 对象示例

```
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:
      containers:
        - name: windowswebserver
          image: mcr.microsoft.com/windows/servercore:ltsc2019 1
          imagePullPolicy: IfNotPresent
          command:
            - powershell.exe 2
```

```

- -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"
os:
  name: "windows"
runtimeClassName: windows2019 3

```

**1** 指定要使用的容器镜像：[mcr.microsoft.com/powershell:<tag>](https://mcr.microsoft.com/powershell) 或 [mcr.microsoft.com/windows/servercore:<tag>](https://mcr.microsoft.com/windows/servercore)。容器镜像必须与节点上运行的 Windows 版本匹配。

- 对于 Windows 2019，使用 **Itsc2019** 标签。
- 对于 Windows 2022，使用 **Itsc2022** 标签。

**2** 指定要在容器上执行的命令。

- 对于 [mcr.microsoft.com/powershell:<tag>](https://mcr.microsoft.com/powershell) 容器镜像，您必须将命令定义为 **pwsh.exe**。
- 对于 [mcr.microsoft.com/windows/servercore:<tag>](https://mcr.microsoft.com/windows/servercore) 容器镜像，您必须将该命令定义为 **powershell.exe**。

**3** 指定您为集群中的 Windows 操作系统变体创建的运行时类。

## 7.4. 手动扩展计算机器集

要在计算机器集中添加或删除机器实例，您可以手动扩展计算机器集。

这个指南与全自动的、安装程序置备的基础架构安装相关。自定义的、用户置备的基础架构安装没有计算机器集。

### 先决条件

- 安装 OpenShift Container Platform 集群和 **oc** 命令行。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

### 流程

1. 运行以下命令，查看集群中的计算机器：

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

计算机器集以 **<clusterid>-worker-<aws-region-az>** 的形式列出。

2. 运行以下命令，查看集群中的计算机器：

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

- 运行以下命令，在要删除的计算机上设置注解：

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

- 运行以下命令来扩展计算机集：

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

或者：

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

## 提示

您还可以应用以下 YAML 来扩展计算机集：

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

您可以扩展或缩减计算机。需要过几分钟以后新机器才可用。



## 重要

默认情况下，机器控制器会尝试排空在机器上运行的节点，直到成功为止。在某些情况下，如错误配置了 pod 中断预算，排空操作可能无法成功。如果排空操作失败，机器控制器无法继续删除机器。

您可以通过在特定机器上注解 **machine.openshift.io/exclude-node-draining** 来跳过排空节点。

## 验证

- 运行以下命令，验证删除所需的机器：

```
$ oc get machines
```

## 第 8 章 WINDOWS 节点升级

您可以通过升级 Windows Machine Config Operator (WMCO)，以确保 Windows 节点具有最新的更新。

### 8.1. WINDOWS MACHINE CONFIG OPERATOR 升级

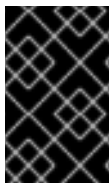
当发布与当前集群版本兼容的 Windows Machine Config Operator (WMCO) 的新版本时，Operator 会根据升级频道和订阅批准策略升级，在使用 Operator Lifecycle Manager (OLM) 时会安装它。WMCO 升级会在 Windows 机器中产生 Kubernetes 组件升级。



#### 注意

如果要升级到 WMCO 的新版本并希望使用集群监控，则必须在 WMCO 命名空间中具有 **openshift.io/cluster-monitoring=true** 标签。如果将该标签添加到已存在的 WMCO 命名空间，并且已经配置了 Windows 节点，重启 WMCO pod 以允许显示监控图形。

对于非破坏性升级，WMCO 会终止之前 WMCO 版本配置的 Windows 机器，并使用当前版本重新创建它们。这可以通过删除 **Machine** 对象来完成，这会导致 Windows 节点排空和删除。为便于升级，WMCO 会为所有配置的节点添加版本注解。在升级过程中，版本注解中的不匹配会导致删除和重新创建 Windows 机器。要在升级过程中减少服务的中断，WMCO 一次只更新一个 Windows 机器。



#### 重要

WMCO 仅负责更新 Kubernetes 组件，而不负责 Windows 操作系统更新。您在创建虚拟机时提供 Windows 镜像，因此您需要提供更新的镜像。您可以通过更改 **MachineSet** spec 中的镜像配置来提供更新的 Windows 镜像。

有关使用 Operator Lifecycle Manager (OLM) 升级 Operator 的更多信息，[请参阅更新已安装的 Operator。](#)

## 第 9 章 使用 BRING-YOUR-OWN-HOST (BYOH) WINDOWS 实例作为节点

通过自带主机 (BYOH)，用户可以重新指定 Windows Server 虚拟机的用途，并将它们带入 OpenShift Container Platform。BYOH Windows 实例让希望在 Windows 服务器脱机时缓解重大中断的用户受益。

### 9.1. 配置 BYOH WINDOWS 实例

创建 BYOH Windows 实例需要在 Windows Machine Config Operator (WMCO) 命名空间中创建配置映射。

#### 先决条件

要作为节点附加到集群的任何 Windows 实例都必须满足以下要求：

- 实例必须与集群中的 Linux worker 节点位于同一个网络中。
- 端口 22 必须处于打开状态，并且正在运行 SSH 服务器。
- SSH 服务器的默认 shell 必须是 [Windows 命令 shell](#) 或 `cmd.exe`。
- 端口 10250 必须处于打开状态才能收集日志。
- 管理员用户会看到一个密钥，它在 secret 集中使用作为一个授权 SSH 密钥。
- 如果您要为安装程序置备的基础架构 (IPI) AWS 集群创建 BYOH Windows 实例，您必须在与 worker 节点的计算机集中的 `spec.template.spec.value.tag` 值匹配的 AWS 实例中添加标签。例如 `kubernetes.io/cluster/<cluster_id>: owned` 或 `kubernetes.io/cluster/<cluster_id>: shared`。
- 如果您要在 vSphere 上创建 BYOH Windows 实例，则必须启用与内部 API 服务器的通信。
- 实例的主机名必须遵循 [RFC 1123](#) DNS 标签要求，其中包括以下标准：
  - 仅包含小写字母数字字符或 '-'。
  - 以字母数字字符开头。
  - 以字母数字字符结尾。



#### 注意

WMCO 部署的 Windows 实例使用容器运行时配置。由于 WMCO 安装和管理运行时，建议您不要在节点上手动安装容器。

#### 流程

1. 在 WMCO 命名空间中创建一个名为 **windows-instances** 的 ConfigMap，用于描述要添加的 Windows 实例。



#### 注意

格式化配置映射的 data 部分的每个条目，使用地址作为关键字，值为 `username=<username>`。

## 配置映射示例

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: windows-instances
  namespace: openshift-windows-machine-config-operator
data:
  10.1.42.1: |- 1
    username=Administrator 2
  instance.example.com: |-
    username=core
```

- 1 WMCO 通过 SSH 访问实例的地址，可以是 DNS 名称或 IPv4 地址。这个地址必须存在 DNS PTR 记录。如果您的组织使用 DHCP 来分配 IP 地址，建议您将 DNS 名称与 BYOH 实例搭配使用。如果没有，则需要在实例分配新 IP 地址时更新 **windows-instances** ConfigMap。
- 2 先决条件中创建的管理员用户的名称。

## 9.2. 删除 BYOH WINDOWS 实例

您可以通过删除配置映射中实例条目来删除附加到集群的 BYOH 实例。删除实例会在添加到集群之前将该实例恢复到其状态。任何日志和容器运行时工件都不会添加到这些实例中。

要完全删除实例，必须使用提供给 WMCO 的当前私钥访问该实例。例如，要从上例中删除 **10.1.42.1** 实例，配置映射将更改为以下内容：

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

删除 **windows-instances** 被视为取消构建添加为节点的所有 Windows 实例的请求。



## 第 10 章 删除 WINDOWS 节点

您可以通过删除其主机 Windows 机器来删除 Windows 节点。

### 10.1. 删除一个特定的机器

您可以删除特定的机器。



#### 重要

除非集群使用 control plane 机器集，否则不要删除 control plane 机器。

#### 先决条件

- 安装 OpenShift Container Platform 集群。
- 安装 OpenShift CLI (**oc**)。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

#### 流程

1. 运行以下命令，查看集群中的机器：

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

命令输出包含 **<clusterid>-<role>-<cloud\_region>** 格式的机器列表。

2. 找到您要删除的机器。
3. 运行以下命令来删除机器：

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



#### 重要

默认情况下，机器控制器会尝试排空在机器上运行的节点，直到成功为止。在某些情况下，如错误配置了 pod 中断预算，排空操作可能无法成功。如果排空操作失败，机器控制器无法继续删除机器。

您可以通过在特定机器上注解 **machine.openshift.io/exclude-node-draining** 来跳过排空节点。

如果要删除的机器属于机器集，则会立即创建一个新机器来满足指定的副本数要求。

## 第 11 章 禁用 WINDOWS 容器工作负载

您可以通过卸载 Windows Machine Config Operator (WMCO)，并删除安装 WMCO 时默认添加的命名空间，来禁用运行 Windows 容器工作负载的能力。

### 11.1. 卸载 WINDOWS MACHINE CONFIG OPERATOR

您可以从集群卸载 Windows Machine Config Operator (WMCO)。

#### 先决条件

- 删除托管 Windows 工作负载的 Windows **Machine** 对象。

#### 流程

1. 在 **Operators → OperatorHub** 页面中，使用 **Filter by keyword** 复选框来搜索 **Red Hat Windows Machine Config Operator**。
2. 点 **Red Hat Windows Machine Config Operator** 标题。Operator 标题表示已安装该 Operator。
3. 在 **Windows Machine Config Operator** 描述符页面中，点 **Uninstall**。

### 11.2. 删除 WINDOWS MACHINE CONFIG OPERATOR 命名空间

您可以删除默认为 Windows Machine Config Operator (WMCO) 生成的命名空间。

#### 先决条件

- WMCO 已从集群中移除。

#### 流程

1. 删除 **openshift-windows-machine-config-operator** 命名空间中创建的所有 Windows 工作负载：

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

2. 验证 **openshift-windows-machine-config-operator** 命名空间中的所有 pod 是否已删除，或处于终止状态：

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

3. 删除 **openshift-windows-machine-config-operator** 命名空间：

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

#### 其他资源

- [从集群中删除 Operator](#)
- [删除 Windows 节点](#)

