OpenShift Container Platform 4.7

OpenShift 的 Windows 容器支持

Red Hat OpenShift for Windows Containers 指南

Last Updated: 2021-05-06

OpenShift Container Platform 4.7 OpenShift 的 Windows 容器支持

Red Hat OpenShift for Windows Containers 指南

法律通告

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

第 1 章 WINDOWS CONTAINER SUPPORT FOR RED HAT OPENSHIFT 发行注记

1.1. 关于 WINDOWS CONTAINER SUPPORT FOR RED HAT OPENSHIFT

Windows Container Support for Red Hat OpenShift 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。在 Windows 节点可用后，您可以在 OpenShift Container Platform 中运行 Windows 容器工作负载。

Red Hat OpenShift for Windows Containers 发行注记包括了 WMCO 的开发信息，WMCO 提供了在 OpenShift Container Platform 中运行 Windows 容器工作负载的功能。

1.2. 获取支持

您必须有订阅才能接受 Red Hat WMCO 支持。在没有订阅的情况下，不支持在生产环境集群中部署 Windows 容器工作负载。如果您没有订阅，您可以使用社区版本的 WMCO，该版本没有官方支持。通过红帽客户门户网站获取支持服务。

1.3. RED HAT WINDOWS MACHINE CONFIG OPERATOR 2.0.0 发行注记

此 WMCO 发行版本为在 OpenShift Container Platform 集群中运行的 Windows 计算节点提供程序错误修正和增强。WMCO 2.0.0 组件在 RHBA-2021:0440 中发布。

WMCO 支持使用在以下云供应商上运行的安装程序置备的基础架构构建的自助管理集群：

- Amazon Web Services (AWS)
- Microsoft Azure
- VMware vSphere

WMCO 发行版本支持以下 Windows 服务器操作系统，具体取决于您的集群安装在哪个平台上：

WMCO 平台	Windows Server 版本
AWS	Windows Server Long-Term Servicing Channel（LTSC）：Windows Server 2019
Azure	Windows Server Long-Term Servicing Channel（LTSC）：Windows Server 2019
vSphere	Windows Server Semi-Annual Channel (SAC): Windows Server 1909，带有 Microsoft patch KB4565351

重要

在受限网络或断开连接的环境中的集群不支持运行 Windows 容器工作负载。

WMCO 版本 2.x 仅与 OpenShift Container Platform 4.7 兼容。

1.3.1. 新功能及改进

此发行版本添加了以下新功能及改进。

1.3.1.1. 支持在 VMware vSphere 上运行的集群

现在，您可以在 VMware vSphere 版本 6.5、6.7 或 7.0 中安装的集群上运行 Windows 节点。您可以在 vSphere 上创建一个 Windows **MachineSet** 对象来托管 Windows Server 计算节点。如需更多信息，请参阅在 vSphere 上创建 Windows **MachineSet** 对象。

1.3.1.2. 增强的 Windows 节点监控

Windows 节点现在与 web 控制台提供的大多数监控功能完全集成。但是，无法查看在此版本中在 Windows 节点上运行的 pod 的工作负载图形。

1.3.2. 已知问题

- web 控制台中可用的文件系统图不会显示 Windows 节点。这是因为文件系统查询的改变所致。以后的 WMCO 发行版本中会解决这个问题。(BZ#1930347)
- WMCO 使用的 Prometheus windows_exporter 目前通过 HTTP 来收集指标，因此被视为不安全。您必须确保只有可信用户才能从端点检索指标。windows_exporter 功能最近添加了对 HTTPS 配置的支持，但还没有在 WMCO 中实施此配置。以后的发行版本中将添加对 WMCO 中的 HTTPS 配置的支持。
- 如果在 Windows Pod 开始运行后创建负载均衡器，kube-proxy 服务会在负载均衡器被创建后意外终止。要避免这个问题，您必须在 Windows pod 状态转换为 **Running** 前创建负载均衡器。（BZ#1939968）

第 2 章 了解 WINDOWS 容器工作负载

Windows Container Support for Red Hat OpenShift 提供了在 OpenShift Container Platform 上运行 Microsoft Windows Server 容器提供内置支持的信息。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。

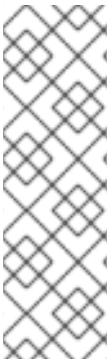
Windows 容器工作负载支持在以下云供应商中运行的集群：

- Amazon Web Services (AWS)
- Microsoft Azure
- VMware vSphere

OpenShift Container Platform 4.7 支持以下 Windows 服务器操作系统：

- Windows Server Long-Term Servicing Channel（LTSC）：Windows Server 2019

如需更多信息，请通过 Windows Server channels 参阅 Microsoft 的文档。



注意

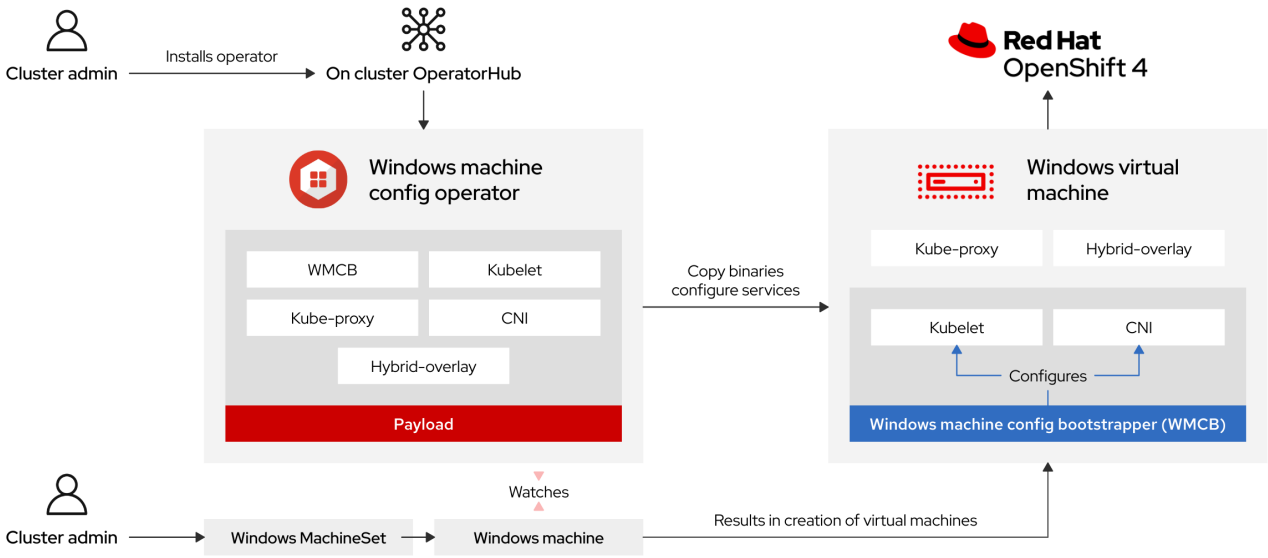
不支持具有 Windows 节点的集群的多租户。托管多租户用法在所有 Kubernetes 环境中都存在安全性问题。额外的安全功能，如 Pod 安全策略，或节点的更精细的访问控制（RBAC），使利用安全漏洞进行工具更困难。但是，如果您选择运行托管多租户工作负载，则管理程序是唯一应使用的安全选项。Kubernetes 的安全域包括整个集群，而不是只限于单个节点。对于可能会有恶意的多租户工作负载，应该使用物理隔离的集群。

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。

2.1. 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 2.1. WMCO 设计



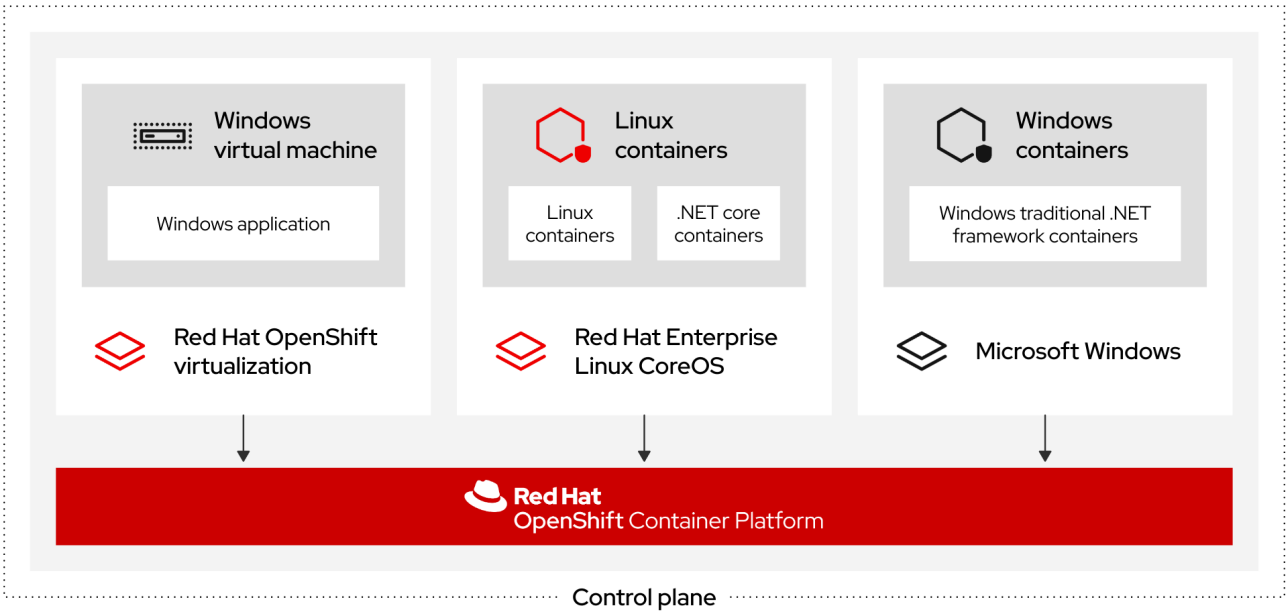
在部署 Windows 工作负载前，您必须创建一个 Windows 计算节点并加入集群。Windows 节点会在集群中托管 Windows 工作负载，并可与其他基于 Linux 的计算节点一起运行。您可以通过创建一个 Windows 机器集来创建一个 Windows 计算节点，以托管 Windows Server 计算机器。您必须对机器集应用特定于 Windows 的标签，该标签指定启用了 Docker 格式的容器运行时附加组件的 Windows OS 镜像。

重要

目前，在 Windows 节点中使用 Docker 格式的容器运行时。Kubernetes 将弃用 Docker 作为容器运行时，详情请参阅 Kubernetes 文档中的 Docker 弃用信息。在未来的 Kubernetes 发行版本中，Containerd 将是 Windows 节点新支持的容器运行时。

WMCO 监视有 Windows 标签的机器。检测到 Windows 机器集并置备相应机器后，WMCO 配置底层 Windows 虚拟机（VM），以便它可以将集群加入为计算节点。

图 2.2. 混合 Windows 和 Linux 工作负载



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。

2.2. 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。
容器网络接口（CNI）插件	为 Windows 节点公开网络。
Windows Machine Config Bootstrapper（WMCB）	配置 kubelet 和 CNI 插件。
hybrid-overlay	创建 OpenShift Container Platform 主机网络服务（HNS）。
kube-proxy	在节点上维护网络规则，以允许外部通信。

第 3 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。

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。使用用户置备的基础架构安装的集群不支持 Windows 容器工作负载。
- 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。

3.1. 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。

3.1.1. 使用 Web 控制台安装 Windows Machine Config Operator

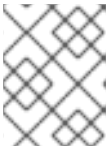
您可以使用 OpenShift Container Platform Web 控制台安装 Windows Machine Config Operator（WMCO）。

流程

1. 从 OpenShift Container Platform Web 控制台中的 **Adminstrator** 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 信息并单击 **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. 点击 **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。

3.1.2. 使用 CLI 安装 Windows Machine Config Operator

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

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 1
  labels:
    openshift.io/cluster-monitoring: "true" 2
```

   1 建议您在 **openshift-windows-machine-config-operator** 命名空间中部署 WMCO。

   2 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" (1)
  installPlanApproval: "Automatic" (2)
  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
```

3.2. 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。

其它资源

- 生成 SSH 私钥并将其添加到代理中
- 在集群中添加 Operator。

第 4 章 创建 WINDOWS MACHINESET 对象

4.1. 在 AWS 上创建 WINDOWS MACHINESET 对象

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 服务器作为操作系统镜像，并启用了 Docker 格式的容器运行时附加组件。

重要

目前，在 Windows 节点中使用 Docker 格式的容器运行时。Kubernetes 将弃用 Docker 作为容器运行时，详情请参阅 Kubernetes 文档中的 Docker 弃用信息。在未来的 Kubernetes 发行版本中，Containerd 将是 Windows 节点新支持的容器运行时。

4.1.1. 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 4.7 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 4.7 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：

Machine

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，Amazon Web Services (AWS) 上的 worker 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 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

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机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**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，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，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。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。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。自动扩展器在集群生命周期内尽可能提供平衡。

4.1.2. 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}{"\n"}' infrastructure cluster
```

2 4 6 指定基础架构 ID、worker 标签和区。

7 将机器配置为 Windows 机器。

8 将 Windows 节点配置为计算机器。

9 指定安装的容器运行时的 Windows 镜像的 AMI ID。您必须使用 Windows Server 2019。

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，所有后续机器组都可以使用 **windows-user-data**。

4.1.3. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集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。

   a. 如果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的机器集。

   ```
   $ 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. 检查特定机器集的值：

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

   输出示例

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
         machine.openshift.io/cluster-api-machine-role: worker 2
         machine.openshift.io/cluster-api-machine-type: worker
         machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
   ```

   1. 集群 ID。
   2. 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

   ```
   $ 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
```

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

4. 新机器设置可用后，检查机器及其引用的节点的状态：

```
$ oc describe machine <name> -n openshift-machine-api
```

例如：

```
$ oc describe machine agl030519-vplxk-windows-worker-us-east-1a -n openshift-machine-api
```

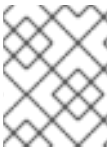
输出示例

```
status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 查看新节点，并确认新节点具有您指定的标签：

```
$ oc get node <node_name> --show-labels
```

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。



注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

4.1.4. 其他资源

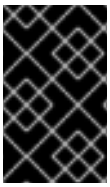
- 有关管理机器集的更多信息，请参阅 *机器管理* 部分。

4.2. 在 AZURE 上创建 WINDOWS MACHINESET 对象

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 服务器作为操作系统镜像，并启用了 Docker 格式的容器运行时附加组件。



重要

目前，在 Windows 节点中使用 Docker 格式的容器运行时。Kubernetes 将弃用 Docker 作为容器运行时，详情请参阅 Kubernetes 文档中的 Docker 弃用信息。在未来的 Kubernetes 发行版本中，Containerd 将是 Windows 节点新支持的容器运行时。

4.2.1. 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 4.7 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 4.7 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：

Machine

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，Amazon Web Services (AWS) 上的 worker 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 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

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机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**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，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，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。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。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。自动扩展器在集群生命周期内尽可能提供平衡。

4.2.2. 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}{"\n"}' infrastructure cluster
```

2 4 6 指定 Windows 机器集名称。Azure 上的 Windows 机器名称不能超过 15 个字符。因此，由于机器名称的生成方式，机器集名称不能超过 9 个字符。

7 将机器配置为 Windows 机器。

8 将 Windows 节点配置为计算机器。

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，所有后续机器组都可以使用 **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。

4.2.3. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集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。

   a. 如果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的机器集。

   ```
   $ 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. 检查特定机器集的值：

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

   输出示例

   ```
   ...
   template:
     metadata:
       labels:
         machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
         machine.openshift.io/cluster-api-machine-role: worker 2
         machine.openshift.io/cluster-api-machine-type: worker
         machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
   ```

   1. 集群 ID。
   2. 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

   ```
   $ 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
```

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

4. 新机器设置可用后，检查机器及其引用的节点的状态：

```
$ oc describe machine <name> -n openshift-machine-api
```

例如：

```
$ oc describe machine agl030519-vplxk-windows-worker-us-east-1a -n openshift-machine-api
```

输出示例

```
status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 查看新节点，并确认新节点具有您指定的标签：

```
$ oc get node <node_name> --show-labels
```

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。

注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

4.2.4. 其他资源

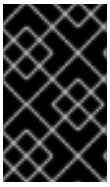
- 有关管理机器集的更多信息，请参阅 *机器管理* 部分。

4.3. 在 VSPHERE 上创建 WINDOWS MACHINESET 对象

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 服务器作为操作系统镜像，并启用了 Docker 格式的容器运行时附加组件。



重要

目前，在 Windows 节点中使用 Docker 格式的容器运行时。Kubernetes 将弃用 Docker 作为容器运行时，详情请参阅 Kubernetes 文档中的 Docker 弃用信息。在未来的 Kubernetes 发行版本中，Containerd 将是 Windows 节点新支持的容器运行时。

4.3.1. 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 4.7 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 4.7 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：

Machine

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，Amazon Web Services (AWS) 上的 worker 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 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

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机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**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，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，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。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。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。自动扩展器在集群生命周期内尽可能提供平衡。

4.3.2. 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。

4.3.2.1. 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

- 您已在使用 OVN-Kubernetes 配置的混合网络的 vSphere 上安装集群。
- 您已在混合网络配置中定义了自定义 VXLAN 端口，用于处理主机间的 pod 到 pod 连接问题。

流程

1. 从 Windows 服务器 1909 虚拟机镜像的更新版本（其中包括 Microsoft 补丁 KB4565351）创建虚拟机。需要这个补丁来设置在 vSphere 上安装的集群所需的 VXLAN UDP 端口。这个补丁不适用于 **Windows 服务器 2019** 虚拟机镜像。

2. 在 Windows 虚拟机中创建 **C:\Users\Administrator\.ssh\authorized_keys** 文件，该文件与您在 **openshift-windows-machine-config-operator** 命名空间中创建的 secret 中的私钥对应。在第一次安装 Windows Machine Config Operator（WMCO）时，会创建 secret 的私钥，以便为 OpenShift Container Platform 访问 Windows 虚拟机。**authorized_keys** 文件用于在 Windows 虚拟机中配置 SSH。

3. 在 Windows 虚拟机上运行以下 Powershell 脚本配置 SSH：

```
# Powershell script to configure SSH on vSphere Windows VMs
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
$firewallRuleName = "ContainerLogsPort"
$containerLogsPort = "10250"
New-NetFirewallRule -DisplayName $firewallRuleName -Direction Inbound -Action Allow -
Protocol TCP -LocalPort $containerLogsPort -EdgeTraversalPolicy Allow
```

```
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
Install-Module -Force OpenSSHUtils
Set-Service -Name ssh-agent -StartupType 'Automatic'
Set-Service -Name sshd -StartupType 'Automatic'
Start-Service ssh-agent
Start-Service sshd
$pubKeyConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'#PubkeyAuthentication yes','PubkeyAuthentication yes'
$pubKeyConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
$passwordConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'#PasswordAuthentication yes','PasswordAuthentication yes'
$passwordConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
$authFileConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'AuthorizedKeysFile
__PROGRAMDATA__/ssh/administrators_authorized_keys','#AuthorizedKeysFile
__PROGRAMDATA__/ssh/administrators_authorized_keys'
$authFileConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
$pubKeyLocationConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'Match Group administrators','#Match Group administrators'
$pubKeyLocationConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
Restart-Service sshd
New-item -Path $env:USERPROFILE -Name .ssh -ItemType Directory -force
```

4. 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。

5. 在 Windows 虚拟机上安装了 VMware Tools 后，请验证以下内容：

   a. **C:\ProgramData\VMware\VMware Tools\tools.conf** 文件有以下条目：

   ```
   exclude-nics=
   ```

   这个条目确保了以下内容：

   - 通过混合式 overlay 在 Windows 虚拟机上生成的克隆的 vNIC 不被忽略。
   - 虚拟机在 vCenter 中有一个 IP 地址。

   b. VMTools Windows 服务正在运行。

6. 拉取应用程序所需的所有 Windows 容器镜像。您拉取的镜像取决于您使用的 Windows 内核。如需更多信息，请参阅 Microsoft 文档中有关拉取 Windows 容器镜像 的内容。

7. 在 Windows 虚拟机上运行 Windows Sysprep 工具 ：

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

提供了 **unattend.xml** 示例，它维护 WMCO 所需的所有更改。例如： **unattend.xml** 文件必须保证管理员主目录与 SSH 公钥保持完整。您必须自定义示例以符合您的需要。

例 4.1. unattend.xml 示例

```
<?xml version="1.0" encoding="UTF-8"?>
<!--A sample unattend.xml which helps in setting admin password and running scripts on
first boot-->
<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>windows-host</ComputerName>
      <ProductKey>My_Product_key</ProductKey>
    </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>
        <Password>
          <Value>MyPassword</Value>
          <PlainText>true</PlainText>
        </Password>
        <Enabled>true</Enabled>
        <Username>Administrator</Username>
      </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>
            <PlainText>true</PlainText>
          </AdministratorPassword>
          <LocalAccounts>
            <LocalAccount wcm:action="add">
              <Description>Administrator</Description>
              <DisplayName>Administrator</DisplayName>
              <Group>Administrators</Group>
              <Name>Administrator</Name>
            </LocalAccount>
          </LocalAccounts>
        </UserAccounts>
      </component>
    </settings>
  </unattend>
```

4.3.2.2. 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。

4.3.3. 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
        kind: VSphereMachineProviderSpec
        memoryMiB: 16384
        network:
          devices:
          - networkName: "<vm_network_name>" 9
        numCPUs: 4
        numCoresPerSocket: 1
        snapshot: ""
        template: <windows_vm_template_name> 10
        userDataSecret:
          name: windows-user-data 11
        workspace:
          datacenter: <vcenter_datacenter_name> 12
          datastore: <vcenter_datastore_name> 13
          folder: <vcenter_vm_folder_path> 14
          resourcePool: <vsphere_resource_pool> 15
          server: <vcenter_server_ip> 16
```

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}{"\n"}' infrastructure cluster
```

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

7 将机器配置为 Windows 机器。

8 将 Windows 节点配置为计算机器。

9 指定要将机器集部署到的 vSphere VM 网络。

10 指定要使用的 Windows vSphere 虚拟机模板的完整路径，如 **/Datacenter/vm/ocp4-llplx/windows-golden-image**。名称必须是唯一的。

11 当配置了第一个 Windows 机器时，**windows-user-data** 由 WMCO 创建。之后，所有后续机器组都可以使用 **windows-user-data**。

12 指定要将机器集部署到的 vCenter Datacenter。

13 指定要部署机器集的 vCenter Datastore。

14 指定 vCenter 中 vSphere 虚拟机文件夹的路径，如 **/dc1/vm/user-inst-5ddjd**。

15 可选：为您的 Windows 虚拟机指定 vSphere 资源池。

16 指定 vCenter 服务器 IP 或完全限定域名。

4.3.4. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集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。

   a. 如果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的机器集。

   ```
   $ 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. 检查特定机器集的值：

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

输出示例

```
...
template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: agl030519-vplxk 1
        machine.openshift.io/cluster-api-machine-role: worker 2
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

1. 集群 ID。
2. 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

```
$ 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
```

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

4. 新机器设置可用后，检查机器及其引用的节点的状态：

```
$ oc describe machine <name> -n openshift-machine-api
```

例如：

```
$ oc describe machine agl030519-vplxk-windows-worker-us-east-1a -n openshift-machine-api
```

输出示例

```
status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
```

```
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2019-05-03T10:34:31Z"
      lastTransitionTime: "2019-05-03T10:34:31Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-09ca0701454124294
    instanceState: running
    kind: AWSMachineProviderStatus
```

5. 查看新节点，并确认新节点具有您指定的标签：

```
$ oc get node <node_name> --show-labels
```

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。

注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

4.3.5. 其他资源

- 有关管理机器集的更多信息，请参阅 *机器管理* 部分。

第 5 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。

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，并启用了 Docker 格式的容器运行时附加组件。
- 您已创建了 Windows 机器集。

重要

目前，在 Windows 节点中使用 Docker 格式的容器运行时。Kubernetes 将弃用 Docker 作为容器运行时，详情请参阅 Kubernetes 文档中的 Docker 弃用信息。在未来的 Kubernetes 发行版本中，Containerd 将是 Windows 节点新支持的容器运行时。

5.1. 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** 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。

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

5.2. 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/v1beta1
kind: RuntimeClass
metadata:
  name: <runtime_class_name> 1
```

```
handler: 'docker'
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"
```

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。
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: <runtime_class_name> 1
...
```

1. 指定管理 pod 调度的运行时类。

5.3. 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:
      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:
          windowsOptions:
```

```
      runAsUserName: "ContainerAdministrator"
  nodeSelector:
    beta.kubernetes.io/os: windows
```

注意

当使用 **mcr.microsoft.com/powershell:<tag>** 容器镜像时，您必须将命令定义为 **pwsh.exe**。如果使用 **mcr.microsoft.com/windows/servercore:<tag>** 容器镜像，则必须将该命令定义为 **powershell.exe**。如需更多信息，请参阅微软的文档。

5.4. 手动扩展机器集

如果您必须在机器集中添加或移除机器实例，则可以手动扩展机器集。

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. 查看集群中的机器集：

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

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

2. 扩展机器集：

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

   或者：

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

   您可以扩展或缩减机器集。需要过几分钟以后新机器才可用。

第 6 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。

6.1. 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。

如需有关使用 Operator Lifecycle Manager（OLM）进行 Operator 升级的更多信息，请参阅升级已安装的 Operator。

第 7 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。

7.1. 删除一个特定的机器

您可以删除特定的机器。

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. 查看集群中的机器，找到要删除的机器：

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

命令输出包含 **<clusterid>-worker-<cloud_region>** 格式的机器列表。

2. 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 的中断预算，节点排空操作可能无法成功完成，从而导致机器无法被删除。您可以在特定机器上使用 "machine.openshift.io/exclude-node-draining" 注解来跳过排空节点的过程。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 8 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。

8.1. 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**。

8.2. 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
```

其它资源

- 从集群中删除 Operator
- 删除 Windows 节点。