OpenShift Container Platform 4.9

Windows Container Support for OpenShift

Red Hat OpenShift for Windows Containers Guide

Last Updated: 2023-04-03

OpenShift Container Platform 4.9 Windows Container Support for OpenShift

Red Hat OpenShift for Windows Containers Guide

Abstract

Red Hat OpenShift for Windows Containers provides built-in support for running Microsoft Windows Server containers on OpenShift Container Platform. This guide provides all the details.

Table of Contents

CHAPTER 1. RED HAT OPENSHIFT SUPPORT FOR WINDOWS CONTAINERS OVERVIEW

Red Hat OpenShift support for Windows Containers is a feature providing the ability to run Windows compute nodes in an OpenShift Container Platform cluster. This is possible by using the Red Hat Windows Machine Config Operator (WMCO) to install and manage Windows nodes. With a Red Hat subscription, you can get support for running Windows workloads in OpenShift Container Platform. For more information, see the release notes.

For workloads including both Linux and Windows, OpenShift Container Platform allows you to deploy Windows workloads running on Windows Server containers while also providing traditional Linux workloads hosted on Red Hat Enterprise Linux CoreOS (RHCOS) or Red Hat Enterprise Linux (RHEL). For more information, see getting started with Windows container workloads.

You need the WMCO to run Windows workloads in your cluster. The WMCO orchestrates the process of deploying and managing Windows workloads on a cluster. For more information, see how to enable Windows container workloads.

You can create a Windows **MachineSet** object to create infrastructure Windows machine sets and related machines so that you can move supported Windows workloads to the new Windows machines. You can create a Windows **MachineSet** object on multiple platforms.

You can schedule Windows workloads to Windows compute nodes.

You can perform Windows Machine Config Operator upgrades to ensure that your Windows nodes have the latest updates.

You can remove a Windows node by deleting a specific machine.

You can use Bring-Your-Own-Host (BYOH) Windows instances to repurpose Windows Server VMs and bring them to OpenShift Container Platform. BYOH Windows instances benefit users who are looking to mitigate major disruptions in the event that a Windows server goes offline. You can use BYOH Windows instances as nodes on OpenShift Container Platform 4.8 and later versions.

You can disable Windows container workloads by performing the following:

- Uninstalling the Windows Machine Config Operator
- Deleting the Windows Machine Config Operator namespace

CHAPTER 2. WINDOWS CONTAINER SUPPORT FOR RED HAT OPENSHIFT RELEASE NOTES

2.1. ABOUT WINDOWS CONTAINER SUPPORT FOR RED HAT OPENSHIFT

Red Hat OpenShift support for Windows Containers enables running Windows compute nodes in an OpenShift Container Platform cluster. Running Windows workloads is possible by using the Red Hat Windows Machine Config Operator (WMCO) to install and manage Windows nodes. With Windows nodes available, you can run Windows container workloads in OpenShift Container Platform.

The release notes for Red Hat OpenShift support for Windows Containers tracks the development of the WMCO, which provides all Windows container workload capabilities in OpenShift Container Platform.

2.2. GETTING SUPPORT

Red Hat OpenShift support for Windows Containers is provided and available as an optional, installable component. Red Hat OpenShift support for Windows Containers is not part of the OpenShift Container Platform subscription. It requires an additional Red Hat subscription and is supported according to the Scope of coverage and Service level agreements.

You must have this separate subscription to receive support for Red Hat OpenShift support for Windows Containers. Without this additional Red Hat subscription, deploying Windows container workloads in production clusters is not supported. You can request support through the Red Hat Customer Portal.

For more information, see the Red Hat OpenShift Container Platform Life Cycle Policy document for Red Hat OpenShift support for Windows Containers .

If you do not have this additional Red Hat subscription, you can use the Community Windows Machine Config Operator, a distribution that lacks official support.

2.3. RELEASE NOTES FOR RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.2

Issued: 2023-3-28

The WMCO 4.0.2 is now available with bug fixes. The components of the WMCO were released in RHBA-2023:1488.

2.3.1. Bug fixes

- Previously, containers on Windows nodes were assigned a hard-coded DNS server IP address (such as 172.30.0.10) based on the default subnet. This caused DNS resolution to fail in certain situations. This fix removes the hard-coded DNS address, and Windows containers are now assigned a valid DNS server IP address that is based on the cluster DNS configuration. As a result, DNS resolution works as expected for Windows containers. (**BZ#2020350**)

2.4. RELEASE NOTES FOR RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.1

Issued: 2021-12-13

The WMCO 4.0.1 is now available with bug fixes. The components of the WMCO were released in RHBA-2021:4757.

2.4.1. Bug fixes

- Previously, the **windows-exporter** metrics endpoint object contained a reference to a deleted machine. This incorrect reference caused the WMCO to ignore deleted events for machines with invalid IP addresses. This bug fix removes the validation of the machine object from the event filtering, allowing the **windows-exporter** metrics endpoint object to correctly update when the machine is still in the **Deleting** phase. (BZ#2008992)

- Previously, deleting the node associated with a Windows **Machine** object returned a reconciliation error upon restart of the Operator. This bug fix opts not to react or reconcile when the node referenced by a Windows machine in the **Running** state is not found within the cluster, preventing any error loop and standardizing functionality with Linux machine objects. (BZ#2009474)

- Previously, certain commands being run by the WMCO in Windows VMs were not parsed correctly by PowerShell. This caused Windows VMs with PowerShell as its default SSH shell to be unable to join to a cluster as a node. The WMCO now identifies the default SSH shell of a Windows VM and runs the associated commands accordingly. This new capability allows Windows VMs with PowerShell as the default SSH shell to be configured as nodes in a cluster. (BZ#2014707)

- Previously, encrypted usernames were being generated with extra tags, which caused them not to display correctly. This bug fix removes the extra tags, allowing the encrypted username to display correctly. (BZ#2016712)

- Previously, the WMCO did not properly associate BYOH Windows VMs with their **Node** object when the VM was specified with a DNS object. This caused the WMCO to attempt to configure VMs that were already fully configured. The WMCO now resolves VMs specified by a DNS address when looking for an associated node. (BZ#2020648)

2.5. RELEASE NOTES FOR RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.0

This release of the WMCO provides bug fixes for running Windows compute nodes in an OpenShift Container Platform cluster. The components of the WMCO 4.0.0 were released in RHBA-2021:3702.

2.5.1. Bug fixes

- Previously, the WMCO used the raw user-provided instance address when creating Bring-Your-Own-Host (BYOH) Windows nodes. This caused BYOH Windows instances to not join an OpenShift Container Platform cluster. This bug fix ensures user-provided DNS names resolve to valid IPv4 addresses, and that the resolved value is used when creating BYOH Windows instances. Now BYOH instances with differing hostnames and DNS addresses can be configured as Windows Nodes. (BZ#1995684)

- Previously, the WMCO performed direct comparisons using unresolved instance addresses when identifying instance-to-node associations. This caused BYOH Windows instances configured to join an OpenShift Container Platform cluster to be removed. This bug fix validates DNS addresses by performing DNS lookups of entries that are added to the **windows-instances** config map. Now the WMCO can properly identify configured instance-to-node relationships, preventing any premature removals of BYOH nodes. (BZ#2005126)

2.5.2. Known issues

- The file system graphs available in the web console do not display for Windows nodes. This issue is caused by changes in the file system queries, which will be fixed in a future release of WMCO. (BZ#1930347)
- For clusters installed on VMware vSphere, the WMCO ignored the **Deleting** phase notification event, leaving incorrect node information in the **windows-exporter** metrics endpoint. This resulted in an invalid mapping for the Prometheus metrics endpoint. This bug has been fixed; the WMCO now recognizes the **Deleting** phase notification event and maps the Prometheus metrics endpoint appropriately. (BZ#1995341)
- When the **RunAsUser** permission is set in the security context of a Linux-based pod, the projected files have the correct permissions set, including container user ownership. However, when the Windows equivalent **RunAsUsername** permission is set in a Windows pod, the kubelet is prevented from setting correct ownership on the files in the projected volume. This problem can get exacerbated when used in conjunction with a hostPath volume where best practices are not followed. For example, giving a pod access to the **C:\var\lib\kubelet\pods\** folder results in that pod being able to access service account tokens from other pods.
  By default, the projected files will have the following ownership, as shown in this example Windows projected volume file:

```
Path   :
Microsoft.PowerShell.Core\FileSystem::C:\var\run\secrets\kubernetes.io\serviceaccount\..2021_
08_31_22_22_18.318230061\ca.crt
Owner  : BUILTIN\Administrators
Group  : NT AUTHORITY\SYSTEM
Access : NT AUTHORITY\SYSTEM Allow  FullControl
         BUILTIN\Administrators Allow  FullControl
         BUILTIN\Users Allow  ReadAndExecute, Synchronize
Audit  :
Sddl   : O:BAG:SYD:AI(A;ID;FA;;;SY)(A;ID;FA;;;BA)(A;ID;0x1200a9;;;BU)
```

  This indicates all administrator users, like someone with the **ContainerAdministrator** role, have read, write, and execute access, while non-administrator users have read and execute access.

  IMPORTANT

  OpenShift Container Platform applies the **RunAsUser** security context to all pods irrespective of its operating system. This means Windows pods automatically have the **RunAsUser** permission applied to its security context.

  In addition, if a Windows pod is created with a projected volume with the default **RunAsUser** permission set, the pod gets stuck in the **ContainerCreating** phase.

  To handle these issues, OpenShift Container Platform forces the file permission handling in projected service account volumes set in the security context of the pod to not be honored for projected volumes on Windows (BZ#1971745). Note that this behavior for Windows pods is how file permission handling used to work for all pod types prior to OpenShift Container Platform 4.7.

2.6. WINDOWS MACHINE CONFIG OPERATOR PREREQUISITES

The following information details the supported platform versions, Windows Server versions, and networking configurations for the Windows Machine Config Operator. See the vSphere documentation for any information that is relevant to only that platform.

2.6.1. Supported platforms based on OpenShift Container Platform and WMCO versions

Platform	Supported OpenShift Container Platform version	Supported WMCO version	Installer-provisioned infrastructure installation support	User-provisioned infrastructure installation support
Amazon Web Services (AWS)	4.6+	WMCO 1.0+	GA	
Microsoft Azure	4.6+	WMCO 1.0+	GA	
VMware vSphere	4.7+	WMCO 2.0+	GA	

2.6.2. Supported platforms for Bring-Your-Own-Host (BYOH) instances based on OpenShift Container Platform and WMCO versions

Platform	Supported OpenShift Container Platform version	Supported WMCO version	BYOH for installer-provisioned infrastructure installation support	BYOH for user-provisioned infrastructure installation support
Amazon Web Services (AWS)	4.8+	WMCO 3.1+	GA	
Microsoft Azure	4.8+	WMCO 3.1+	GA	
VMware vSphere	4.8+	WMCO 3.1+	GA	
Provider agnostic (Platform: none)	4.8+	WMCO 3.1+		GA

2.6.3. Supported Windows Server versions

The following table lists the supported Windows Server version based on the applicable platform. Any unlisted Windows Server version is not supported and will cause errors. To prevent these errors, only use the appropriate version according to the platform in use.

Platform	Supported Windows Server version
Amazon Web Services (AWS)	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
Microsoft Azure	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
VMware vSphere	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2022 (OS Build 20348.681 or later)
bare metal	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019

2.6.4. Supported networking

Hybrid networking with OVN-Kubernetes is the only supported networking configuration. See the additional resources below for more information on this functionality. The following tables outline the type of networking configuration and Windows Server versions to use based on your platform. You must specify the network configuration when you install the cluster. Be aware that OpenShift SDN networking is the default network for OpenShift Container Platform clusters. However, OpenShift SDN is not supported by WMCO.

Table 2.1. Platform networking support

Platform	Supported networking
Amazon Web Services (AWS)	Hybrid networking with OVN-Kubernetes
Microsoft Azure	Hybrid networking with OVN-Kubernetes
VMware vSphere	Hybrid networking with OVN-Kubernetes with a custom VXLAN port
bare metal	Hybrid networking with OVN-Kubernetes

Table 2.2. Hybrid OVN-Kubernetes Windows Server support

Hybrid networking with OVN-Kubernetes	Supported Windows Server version
Default VXLAN port	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
Custom VXLAN port	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2022 (OS Build 20348.681 or later)

IMPORTANT

Running Windows container workloads is not supported for clusters in a restricted network or disconnected environment.

Version 4.x of the WMCO is only compatible with OpenShift Container Platform 4.9.

2.7. KNOWN LIMITATIONS

Note the following limitations when working with Windows nodes managed by the WMCO (Windows nodes):

- The following OpenShift Container Platform features are not supported on Windows nodes:
  - Red Hat OpenShift Developer CLI (odo)
  - Image builds
  - OpenShift Pipelines
  - OpenShift Service Mesh
  - OpenShift monitoring of user-defined projects
  - OpenShift Serverless
  - Horizontal Pod Autoscaling
  - Vertical Pod Autoscaling
- The following Red Hat features are not supported on Windows nodes:
  - Red Hat cost management
  - Red Hat OpenShift Local
- Windows nodes do not support pulling container images from private registries. You can use images from public registries or pre-pull the images.
- Windows nodes do not support workloads created by using deployment configs. You can use a deployment or other method to deploy workloads.
- Windows nodes are not supported in clusters that use a cluster-wide proxy. This is because the WMCO is not able to route traffic through the proxy connection for the workloads.
- Windows nodes are not supported in clusters that are in a disconnected environment.
- Red Hat OpenShift support for Windows Containers does not support adding Windows nodes to a cluster through a trunk port. The only supported networking configuration for adding Windows nodes is through an access port that carries traffic for the VLAN.
- Red Hat OpenShift support for Windows Containers supports only in-tree storage drivers for all cloud providers.
- Kubernetes has identified the following node feature limitations :
  - Huge pages are not supported for Windows containers.

    - Privileged containers are not supported for Windows containers.
    - Pod termination grace periods require the containerd container runtime to be installed on the Windows node.

- Kubernetes has identified several API compatibility issues.

CHAPTER 3. UNDERSTANDING WINDOWS CONTAINER WORKLOADS

Red Hat OpenShift support for Windows Containers provides built-in support for running Microsoft Windows Server containers on OpenShift Container Platform. For those that administer heterogeneous environments with a mix of Linux and Windows workloads, OpenShift Container Platform allows you to deploy Windows workloads running on Windows Server containers while also providing traditional Linux workloads hosted on Red Hat Enterprise Linux CoreOS (RHCOS) or Red Hat Enterprise Linux (RHEL).

NOTE

Multi-tenancy for clusters that have Windows nodes is not supported. Hostile multi-tenant usage introduces security concerns in all Kubernetes environments. Additional security features like pod security policies, or more fine-grained role-based access control (RBAC) for nodes, make exploits more difficult. However, if you choose to run hostile multi-tenant workloads, a hypervisor is the only security option you should use. The security domain for Kubernetes encompasses the entire cluster, not an individual node. For these types of hostile multi-tenant workloads, you should use physically isolated clusters.

Windows Server Containers provide resource isolation using a shared kernel but are not intended to be used in hostile multitenancy scenarios. Scenarios that involve hostile multitenancy should use Hyper-V Isolated Containers to strongly isolate tenants.

3.1. WINDOWS MACHINE CONFIG OPERATOR PREREQUISITES

The following information details the supported platform versions, Windows Server versions, and networking configurations for the Windows Machine Config Operator. See the vSphere documentation for any information that is relevant to only that platform.

3.1.1. Supported platforms based on OpenShift Container Platform and WMCO versions

Platform	Supported OpenShift Container Platform version	Supported WMCO version	Installer-provisioned infrastructure installation support	User-provisioned infrastructure installation support
Amazon Web Services (AWS)	4.6+	WMCO 1.0+	GA	
Microsoft Azure	4.6+	WMCO 1.0+	GA	
VMware vSphere	4.7+	WMCO 2.0+	GA	

3.1.2. Supported platforms for Bring-Your-Own-Host (BYOH) instances based on OpenShift Container Platform and WMCO versions

Platform	Supported OpenShift Container Platform version	Supported WMCO version	BYOH for installer-provisioned infrastructure installation support	BYOH for user-provisioned infrastructure installation support
Amazon Web Services (AWS)	4.8+	WMCO 3.1+	GA	
Microsoft Azure	4.8+	WMCO 3.1+	GA	
VMware vSphere	4.8+	WMCO 3.1+	GA	
Provider agnostic (Platform: none)	4.8+	WMCO 3.1+		GA

3.1.3. Supported Windows Server versions

The following table lists the supported Windows Server version based on the applicable platform. Any unlisted Windows Server version is not supported and will cause errors. To prevent these errors, only use the appropriate version according to the platform in use.

Platform	Supported Windows Server version
Amazon Web Services (AWS)	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
Microsoft Azure	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
VMware vSphere	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2022 (OS Build 20348.681 or later)
bare metal	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019

3.1.4. Supported networking

Hybrid networking with OVN-Kubernetes is the only supported networking configuration. See the additional resources below for more information on this functionality. The following tables outline the type of networking configuration and Windows Server versions to use based on your platform. You must specify the network configuration when you install the cluster. Be aware that OpenShift SDN networking is the default network for OpenShift Container Platform clusters. However, OpenShift SDN is not supported by WMCO.

Table 3.1. Platform networking support

Platform	Supported networking
Amazon Web Services (AWS)	Hybrid networking with OVN-Kubernetes
Microsoft Azure	Hybrid networking with OVN-Kubernetes
VMware vSphere	Hybrid networking with OVN-Kubernetes with a custom VXLAN port
bare metal	Hybrid networking with OVN-Kubernetes

Table 3.2. Hybrid OVN-Kubernetes Windows Server support

Hybrid networking with OVN-Kubernetes	Supported Windows Server version
Default VXLAN port	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
Custom VXLAN port	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2022 (OS Build 20348.681 or later)

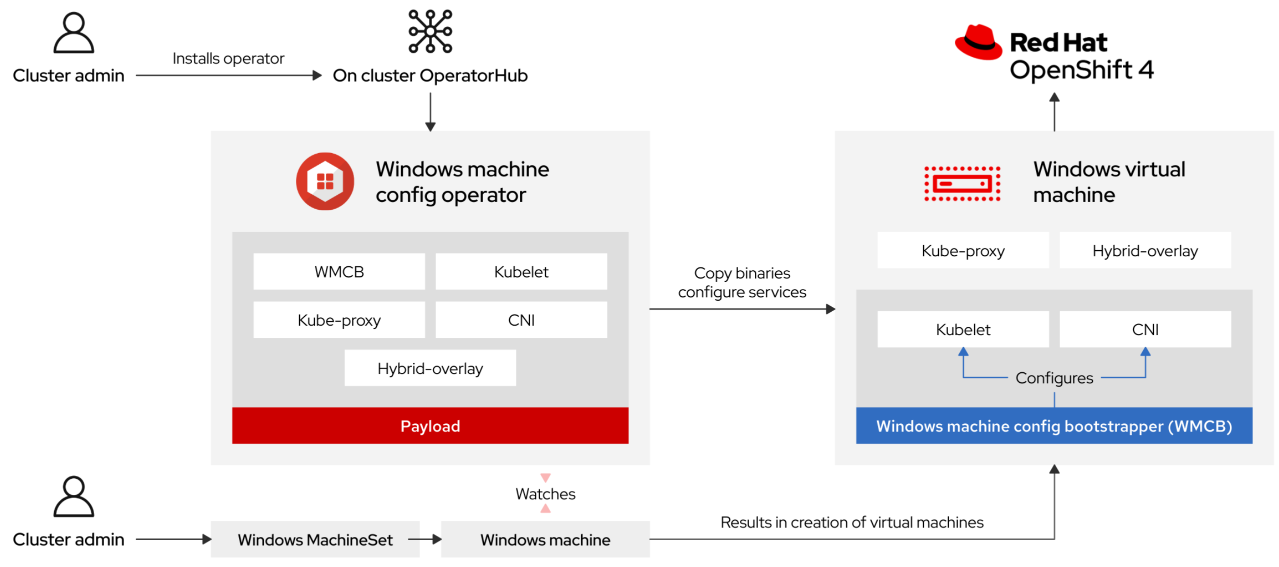
Additional resources

- See Configuring hybrid networking with OVN-Kubernetes

3.2. WINDOWS WORKLOAD MANAGEMENT

To run Windows workloads in your cluster, you must first install the Windows Machine Config Operator (WMCO). The WMCO is a Linux-based Operator that runs on Linux-based control plane and compute nodes. The WMCO orchestrates the process of deploying and managing Windows workloads on a cluster.

Figure 3.1. WMCO design



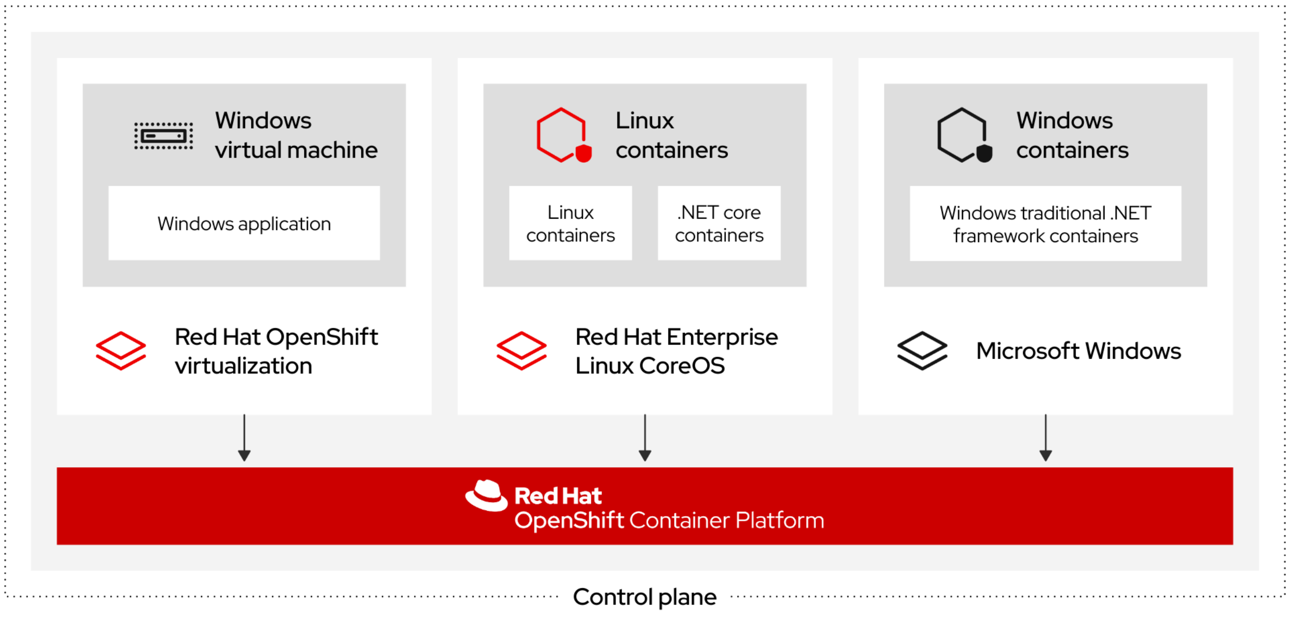
Before deploying Windows workloads, you must create a Windows compute node and have it join the cluster. The Windows node hosts the Windows workloads in a cluster, and can run alongside other Linux-based compute nodes. You can create a Windows compute node by creating a Windows machine set to host Windows Server compute machines. You must apply a Windows-specific label to the machine set that specifies a Windows OS image that has the Docker-formatted container runtime add-on enabled.

IMPORTANT

Currently, the Docker-formatted container runtime is used in Windows nodes. Kubernetes is deprecating Docker as a container runtime; you can reference the Kubernetes documentation for more information in Docker deprecation. Containerd will be the new supported container runtime for Windows nodes in a future release of Kubernetes.

The WMCO watches for machines with the Windows label. After a Windows machine set is detected and its respective machines are provisioned, the WMCO configures the underlying Windows virtual machine (VM) so that it can join the cluster as a compute node.

Figure 3.2. Mixed Windows and Linux workloads


Windows virtual machine
Windows application
Red Hat OpenShift virtualization
Linux containers
Linux containers
.NET core containers
Red Hat Enterprise Linux CoreOS
Windows containers
Windows traditional .NET framework containers
Microsoft Windows
Red Hat OpenShift Container Platform
Control plane

The WMCO expects a predetermined secret in its namespace containing a private key that is used to interact with the Windows instance. WMCO checks for this secret during boot up time and creates a user data secret which you must reference in the Windows **MachineSet** object that you created. Then the WMCO populates the user data secret with a public key that corresponds to the private key. With this data in place, the cluster can connect to the Windows VM using an SSH connection.

After the cluster establishes a connection with the Windows VM, you can manage the Windows node using similar practices as you would a Linux-based node.

NOTE

The OpenShift Container Platform web console provides most of the same monitoring capabilities for Windows nodes that are available for Linux nodes. However, the ability to monitor workload graphs for pods running on Windows nodes is not available at this time.

Scheduling Windows workloads to a Windows node can be done with typical pod scheduling practices like taints, tolerations, and node selectors; alternatively, you can differentiate your Windows workloads from Linux workloads and other Windows-versioned workloads by using a **RuntimeClass** object.

3.3. WINDOWS NODE SERVICES

The following Windows-specific services are installed on each Windows node:

Service	Description
kubelet	Registers the Windows node and manages its status.
Container Network Interface (CNI) plugins	Exposes networking for Windows nodes.
Windows Machine Config Bootstrapper (WMCB)	Configures the kubelet and CNI plugins.
Windows Exporter	Exports Prometheus metrics from Windows nodes
hybrid-overlay	Creates the OpenShift Container Platform Host Network Service (HNS).
kube-proxy	Maintains network rules on nodes allowing outside communication.

3.4. KNOWN LIMITATIONS

Note the following limitations when working with Windows nodes managed by the WMCO (Windows nodes):

- The following OpenShift Container Platform features are not supported on Windows nodes:
  - Red Hat OpenShift Developer CLI (odo)
  - Image builds
  - OpenShift Pipelines
  - OpenShift Service Mesh
  - OpenShift monitoring of user-defined projects
  - OpenShift Serverless
  - Horizontal Pod Autoscaling
  - Vertical Pod Autoscaling
- The following Red Hat features are not supported on Windows nodes:
  - Red Hat cost management
  - Red Hat OpenShift Local
- Windows nodes do not support pulling container images from private registries. You can use images from public registries or pre-pull the images.

- Windows nodes do not support workloads created by using deployment configs. You can use a deployment or other method to deploy workloads.
- Windows nodes are not supported in clusters that use a cluster-wide proxy. This is because the WMCO is not able to route traffic through the proxy connection for the workloads.
- Windows nodes are not supported in clusters that are in a disconnected environment.
- Red Hat OpenShift support for Windows Containers does not support adding Windows nodes to a cluster through a trunk port. The only supported networking configuration for adding Windows nodes is through an access port that carries traffic for the VLAN.
- Red Hat OpenShift support for Windows Containers supports only in-tree storage drivers for all cloud providers.
- Kubernetes has identified the following node feature limitations :
  - Huge pages are not supported for Windows containers.
  - Privileged containers are not supported for Windows containers.
  - Pod termination grace periods require the containerd container runtime to be installed on the Windows node.
- Kubernetes has identified several API compatibility issues.

CHAPTER 4. ENABLING WINDOWS CONTAINER WORKLOADS

Before adding Windows workloads to your cluster, you must install the Windows Machine Config Operator (WMCO), which is available in the OpenShift Container Platform OperatorHub. The WMCO orchestrates the process of deploying and managing Windows workloads on a cluster.

NOTE

Dual NIC is not supported on WMCO-managed Windows instances.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have installed your cluster using installer-provisioned infrastructure, or using user-provisioned infrastructure with the **platform: none** field set in your **install-config.yaml** file.
- You have configured hybrid networking with OVN-Kubernetes for your cluster. This must be completed during the installation of your cluster. For more information, see Configuring hybrid networking.
- You are running an OpenShift Container Platform cluster version 4.6.8 or later.

NOTE

The WMCO is not supported in clusters that use a cluster-wide proxy because the WMCO is not able to route traffic through the proxy connection for the workloads.

Additional resources

- For the comprehensive prerequisites for the Windows Machine Config Operator, see Understanding Windows container workloads.

4.1. INSTALLING THE WINDOWS MACHINE CONFIG OPERATOR

You can install the Windows Machine Config Operator using either the web console or OpenShift CLI (**oc**).

4.1.1. Installing the Windows Machine Config Operator using the web console

You can use the OpenShift Container Platform web console to install the Windows Machine Config Operator (WMCO).

NOTE

Dual NIC is not supported on WMCO-managed Windows instances.

Procedure

1. From the **Administrator** perspective in the OpenShift Container Platform web console, navigate to the **Operators → OperatorHub** page.
2. Use the **Filter by keyword** box to search for **Windows Machine Config Operator** in the catalog. Click the **Windows Machine Config Operator** tile.
3. Review the information about the Operator and click **Install**.
4. On the **Install Operator** page:
   a. Select the **stable** channel as the **Update Channel**. The **stable** channel enables the latest stable release of the WMCO to be installed.
   b. The **Installation Mode** is preconfigured because the WMCO must be available in a single namespace only.
   c. Choose the **Installed Namespace** for the WMCO. The default Operator recommended namespace is **openshift-windows-machine-config-operator**.
   d. Click the **Enable Operator recommended cluster monitoring on the Namespace** checkbox to enable cluster monitoring for the WMCO.
   e. Select an **Approval Strategy**.
      - The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
      - The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.

1. Click **Install**. The WMCO is now listed on the **Installed Operators** page.

   NOTE

   The WMCO is installed automatically into the namespace you defined, like **openshift-windows-machine-config-operator**.

2. Verify that the **Status** shows **Succeeded** to confirm successful installation of the WMCO.

4.1.2. Installing the Windows Machine Config Operator using the CLI

You can use the OpenShift CLI (**oc**) to install the Windows Machine Config Operator (WMCO).

NOTE

Dual NIC is not supported on WMCO-managed Windows instances.

Procedure

1. Create a namespace for the WMCO.
   a. Create a **Namespace** object YAML file for the WMCO. For example, **wmco-namespace.yaml**:

```
apiVersion: v1
kind: Namespace
```

```
metadata:
  name: openshift-windows-machine-config-operator (1)
  labels:
    openshift.io/cluster-monitoring: "true" (2)
```

(1) It is recommended to deploy the WMCO in the **openshift-windows-machine-config-operator** namespace.

(2) This label is required for enabling cluster monitoring for the WMCO.

b. Create the namespace:

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

For example:

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

2. Create the Operator group for the WMCO.

a. Create an **OperatorGroup** object YAML file. For example, **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. Create the Operator group:

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

For example:

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

3. Subscribe the namespace to the WMCO.

a. Create a **Subscription** object YAML file. For example, **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) Specify **stable** as the channel.

(2) Set an approval strategy. You can set **Automatic** or **Manual**.

(3) Specify the **redhat-operators** catalog source, which contains the **windows-machine-config-operator** package manifests. If your OpenShift Container Platform is installed on a restricted network, also known as a disconnected cluster, specify the name of the **CatalogSource** object you created when you configured the Operator LifeCycle Manager (OLM).

(4) Namespace of the catalog source. Use **openshift-marketplace** for the default OperatorHub catalog sources.

b. Create the subscription:

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

For example:

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

The WMCO is now installed to the **openshift-windows-machine-config-operator**.

4. Verify the WMCO installation:

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

Example output

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

4.2. CONFIGURING A SECRET FOR THE WINDOWS MACHINE CONFIG OPERATOR

To run the Windows Machine Config Operator (WMCO), you must create a secret in the WMCO namespace containing a private key. This is required to allow the WMCO to communicate with the Windows virtual machine (VM).

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You created a PEM-encoded file containing an RSA key.

Procedure

- Define the secret required to access the Windows VMs:

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

(1) You must create the private key in the WMCO namespace, like **openshift-windows-machine-config-operator**.

It is recommended to use a different private key than the one used when installing the cluster.

4.3. ADDITIONAL RESOURCES

- Generating a key pair for cluster node SSH access
- Adding Operators to a cluster .

CHAPTER 5. CREATING WINDOWS MACHINESET OBJECTS

5.1. CREATING A WINDOWS MACHINESET OBJECT ON AWS

You can create a Windows **MachineSet** object to serve a specific purpose in your OpenShift Container Platform cluster on Amazon Web Services (AWS). For example, you might create infrastructure Windows machine sets and related machines so that you can move supporting Windows workloads to the new Windows machines.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You are using a supported Windows Server as the operating system image with the Docker-formatted container runtime add-on enabled.
  Use the following **aws** command to query valid AMI images:

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

IMPORTANT

Currently, the Docker-formatted container runtime is used in Windows nodes. Kubernetes is deprecating Docker as a container runtime; you can reference the Kubernetes documentation for more information in Docker deprecation. Containerd will be the new supported container runtime for Windows nodes in a future release of Kubernetes.

5.1.1. Machine API overview

The Machine API is a combination of primary resources that are based on the upstream Cluster API project and custom OpenShift Container Platform resources.

For OpenShift Container Platform 4.9 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.9 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

A fundamental unit that describes the host for a node. A machine has a **providerSpec** specification, which describes the types of compute nodes that are offered for different cloud platforms. For example, a machine type for a worker node on Amazon Web Services (AWS) might define a specific machine type and required metadata.

Machine sets

MachineSet resources are groups of machines. Machine sets are to machines as replica sets are to pods. If you need more machines or must scale them down, you change the **replicas** field on the machine set to meet your compute need.

WARNING

Control plane machines cannot be managed by machine sets.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales compute machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified compute machine set, and the machine autoscaler maintains that range of nodes.
The **MachineAutoscaler** object takes effect after a **ClusterAutoscaler** object exists. Both **ClusterAutoscaler** and **MachineAutoscaler** resources are made available by the **ClusterAutoscalerOperator** object.

Cluster autoscaler

This resource is based on the upstream cluster autoscaler project. In the OpenShift Container Platform implementation, it is integrated with the Machine API by extending the machine set API. You can set cluster-wide scaling limits for resources such as cores, nodes, memory, GPU, and so on. You can set the priority so that the cluster prioritizes pods so that new nodes are not brought online for less important pods. You can also set the scaling policy so that you can scale up nodes but not scale them down.

Machine health check

The **MachineHealthCheck** resource detects when a machine is unhealthy, deletes it, and, on supported platforms, makes a new machine.

In OpenShift Container Platform version 3.11, you could not roll out a multi-zone architecture easily because the cluster did not manage machine provisioning. Beginning with OpenShift Container Platform version 4.1, this process is easier. Each machine set is scoped to a single zone, so the installation program sends out machine sets across availability zones on your behalf. And then because your compute is dynamic, and in the face of a zone failure, you always have a zone for when you must rebalance your machines. The autoscaler provides best-effort balancing over the life of a cluster.

5.1.2. Sample YAML for a Windows MachineSet object on AWS

This sample YAML defines a Windows **MachineSet** object running on Amazon Web Services (AWS) that the Windows Machine Config Operator (WMCO) can react upon.

```
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 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. You can obtain the infrastructure ID by running the following command:

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

2 4 6 Specify the infrastructure ID, worker label, and zone.

7 Configure the machine set as a Windows machine.

8 Configure the Windows node as a compute machine.

9 Specify the AMI ID of a Windows image with a container runtime installed. You must use Windows Server 2019.

11 Specify the AWS zone, like **us-east-1a**.

12 Specify the AWS region, like **us-east-1**.

16 Created by the WMCO when it is configuring the first Windows machine. After that, the **windows-user-data** is available for all subsequent machine sets to consume.

5.1.3. Creating a machine set

In addition to the compute machine sets created by the installation program, you can create your own to dynamically manage the machine compute resources for specific workloads of your choice.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.

Procedure

1. Create a new YAML file that contains the machine set custom resource (CR) sample and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.
2. Optional: If you are not sure which value to set for a specific field, you can check an existing compute machine set from your cluster.
   a. To list the compute machine sets in your cluster, run the following command:

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

Example output

```
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. To view values of a specific compute machine set custom resource (CR), run the following command:

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

Example output

```
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 The cluster infrastructure ID.

2 A default node label.

NOTE

For clusters that have user-provisioned infrastructure, a compute machine set can only create **worker** and **infra** type machines.

3 The values in the **<providerSpec>** section of the compute machine set CR are platform-specific. For more information about **<providerSpec>** parameters in the CR, see the sample compute machine set CR configuration for your provider.

3. Create a **MachineSet** CR by running the following command:

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

Verification

- View the list of compute machine sets by running the following command:

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

Example output

```
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
```

When the new machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

5.1.4. Additional resources

- For more information on managing machine sets, see the *Machine management* section.

5.2. CREATING A WINDOWS MACHINESET OBJECT ON AZURE

You can create a Windows **MachineSet** object to serve a specific purpose in your OpenShift Container Platform cluster on Microsoft Azure. For example, you might create infrastructure Windows machine sets and related machines so that you can move supporting Windows workloads to the new Windows machines.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).

- You are using a supported Windows Server as the operating system image with the Docker-formatted container runtime add-on enabled.

IMPORTANT

Currently, the Docker-formatted container runtime is used in Windows nodes. Kubernetes is deprecating Docker as a container runtime; you can reference the Kubernetes documentation for more information in Docker deprecation. Containerd will be the new supported container runtime for Windows nodes in a future release of Kubernetes.

5.2.1. Machine API overview

The Machine API is a combination of primary resources that are based on the upstream Cluster API project and custom OpenShift Container Platform resources.

For OpenShift Container Platform 4.9 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.9 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

A fundamental unit that describes the host for a node. A machine has a **providerSpec** specification, which describes the types of compute nodes that are offered for different cloud platforms. For example, a machine type for a worker node on Amazon Web Services (AWS) might define a specific machine type and required metadata.

Machine sets

MachineSet resources are groups of machines. Machine sets are to machines as replica sets are to pods. If you need more machines or must scale them down, you change the **replicas** field on the machine set to meet your compute need.

> **WARNING**
>
> Control plane machines cannot be managed by machine sets.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales compute machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified compute machine set, and the machine autoscaler maintains that range of nodes.
The **MachineAutoscaler** object takes effect after a **ClusterAutoscaler** object exists. Both **ClusterAutoscaler** and **MachineAutoscaler** resources are made available by the **ClusterAutoscalerOperator** object.

Cluster autoscaler

This resource is based on the upstream cluster autoscaler project. In the OpenShift Container Platform implementation, it is integrated with the Machine API by extending the machine set API. You can set cluster-wide scaling limits for resources such as cores, nodes, memory, GPU, and so on. You can set the priority so that the cluster prioritizes pods so that new nodes are not brought online for less important pods. You can also set the scaling policy so that you can scale up nodes but not scale them down.

Machine health check

The **MachineHealthCheck** resource detects when a machine is unhealthy, deletes it, and, on supported platforms, makes a new machine.

In OpenShift Container Platform version 3.11, you could not roll out a multi-zone architecture easily because the cluster did not manage machine provisioning. Beginning with OpenShift Container Platform version 4.1, this process is easier. Each machine set is scoped to a single zone, so the installation program sends out machine sets across availability zones on your behalf. And then because your compute is dynamic, and in the face of a zone failure, you always have a zone for when you must rebalance your machines. The autoscaler provides best-effort balancing over the life of a cluster.

5.2.2. Sample YAML for a Windows MachineSet object on Azure

This sample YAML defines a Windows **MachineSet** object running on Microsoft Azure that the Windows Machine Config Operator (WMCO) can react upon.

```
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 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. You can obtain the infrastructure ID by running the following command:

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

2 4 6 Specify the Windows machine set name. Windows machine names on Azure cannot be more than 15 characters long. Therefore, the machine set name cannot be more than 9 characters long, due to the way machine names are generated from it.

7 Configure the machine set as a Windows machine.

8 Configure the Windows node as a compute machine.

9 Specify a **WindowsServer** image offering that defines the **2019-Datacenter-with-Containers** SKU.

10 Specify the Azure region, like **centralus**.

14 Created by the WMCO when it is configuring the first Windows machine. After that, the **windows-user-data** is available for all subsequent machine sets to consume.

16 Specify the zone within your region to place machines on. Be sure that your region supports the zone that you specify.

5.2.3. Creating a machine set

In addition to the compute machine sets created by the installation program, you can create your own to dynamically manage the machine compute resources for specific workloads of your choice.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.

Procedure

1. Create a new YAML file that contains the machine set custom resource (CR) sample and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.
2. Optional: If you are not sure which value to set for a specific field, you can check an existing compute machine set from your cluster.
   a. To list the compute machine sets in your cluster, run the following command:

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

   Example output

```
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. To view values of a specific compute machine set custom resource (CR), run the following command:

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

Example output

```
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 The cluster infrastructure ID.

2 A default node label.

NOTE

For clusters that have user-provisioned infrastructure, a compute machine set can only create **worker** and **infra** type machines.

3 The values in the **<providerSpec>** section of the compute machine set CR are platform-specific. For more information about **<providerSpec>** parameters in the CR, see the sample compute machine set CR configuration for your provider.

3. Create a **MachineSet** CR by running the following command:

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

Verification

- View the list of compute machine sets by running the following command:

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

Example output

```
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
```

When the new machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

5.2.4. Additional resources

- For more information on managing machine sets, see the *Machine management* section.

5.3. CREATING A WINDOWS MACHINESET OBJECT ON VSPHERE

You can create a Windows **MachineSet** object to serve a specific purpose in your OpenShift Container Platform cluster on VMware vSphere. For example, you might create infrastructure Windows machine sets and related machines so that you can move supporting Windows workloads to the new Windows machines.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You are using a supported Windows Server as the operating system image with the Docker-formatted container runtime add-on enabled.

IMPORTANT

Currently, the Docker-formatted container runtime is used in Windows nodes. Kubernetes is deprecating Docker as a container runtime; you can reference the Kubernetes documentation for more information on Docker deprecation. Containerd will be the new supported container runtime for Windows nodes in a future release of Kubernetes.

5.3.1. Machine API overview

The Machine API is a combination of primary resources that are based on the upstream Cluster API project and custom OpenShift Container Platform resources.

For OpenShift Container Platform 4.9 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.9 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

A fundamental unit that describes the host for a node. A machine has a **providerSpec** specification, which describes the types of compute nodes that are offered for different cloud platforms. For example, a machine type for a worker node on Amazon Web Services (AWS) might define a specific machine type and required metadata.

Machine sets

MachineSet resources are groups of machines. Machine sets are to machines as replica sets are to pods. If you need more machines or must scale them down, you change the **replicas** field on the machine set to meet your compute need.

WARNING

Control plane machines cannot be managed by machine sets.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales compute machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified compute machine set, and the machine autoscaler maintains that range of nodes.
The **MachineAutoscaler** object takes effect after a **ClusterAutoscaler** object exists. Both **ClusterAutoscaler** and **MachineAutoscaler** resources are made available by the **ClusterAutoscalerOperator** object.

Cluster autoscaler

This resource is based on the upstream cluster autoscaler project. In the OpenShift Container Platform implementation, it is integrated with the Machine API by extending the machine set API. You can set cluster-wide scaling limits for resources such as cores, nodes, memory, GPU, and so on. You can set the priority so that the cluster prioritizes pods so that new nodes are not brought online for less important pods. You can also set the scaling policy so that you can scale up nodes but not scale them down.

Machine health check

The **MachineHealthCheck** resource detects when a machine is unhealthy, deletes it, and, on supported platforms, makes a new machine.

In OpenShift Container Platform version 3.11, you could not roll out a multi-zone architecture easily because the cluster did not manage machine provisioning. Beginning with OpenShift Container Platform version 4.1, this process is easier. Each machine set is scoped to a single zone, so the installation program sends out machine sets across availability zones on your behalf. And then because your compute is dynamic, and in the face of a zone failure, you always have a zone for when you must rebalance your machines. The autoscaler provides best-effort balancing over the life of a cluster.

5.3.2. Preparing your vSphere environment for Windows container workloads

You must prepare your vSphere environment for Windows container workloads by creating the vSphere Windows VM golden image and enabling communication with the internal API server for the WMCO.

5.3.2.1. Creating the vSphere Windows VM golden image

Create a vSphere Windows virtual machine (VM) golden image.

Prerequisites

- You have created a private/public key pair, which is used to configure key-based authentication in the OpenSSH server. The private key must also be configured in the Windows Machine Config Operator (WMCO) namespace. This is required to allow the WMCO to communicate with the Windows VM. See the "Configuring a secret for the Windows Machine Config Operator" section for more details.

> **NOTE**
>
> You must use Microsoft PowerShell commands in several cases when creating your Windows VM. PowerShell commands in this guide are distinguished by the **PS C:\>** prefix.

Procedure

1. Create a new VM in the vSphere client using the Windows Server Semi-Annual Channel (SAC): Windows Server 20H2 ISO image that includes the Microsoft patch KB4565351. This patch is required to set the VXLAN UDP port, which is required for clusters installed on vSphere. See the VMware documentation for more information.

   > **IMPORTANT**
   >
   > The virtual hardware version for your VM must meet the infrastructure requirements for OpenShift Container Platform. For more information, see the "VMware vSphere infrastructure requirements" section in the OpenShift Container Platform documentation. Also, you can refer to VMware's documentation on virtual machine hardware versions.

2. Install and configure VMware Tools version 11.0.6 or greater on the Windows VM. See the VMware Tools documentation for more information.

3. After installing VMware Tools on the Windows VM, verify the following:

   a. The **C:\ProgramData\VMware\VMware Tools\tools.conf** file exists with the following entry:

      ```
      exclude-nics=
      ```

      If the **tools.conf** file does not exist, create it with the **exclude-nics** option uncommented and set as an empty value.

This entry ensures the cloned vNIC generated on the Windows VM by the hybrid-overlay is not ignored.

b. The Windows VM has a valid IP address in vCenter:

```
C:\> ipconfig
```

c. The VMTools Windows service is running:

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

4. Install and configure the OpenSSH Server on the Windows VM. See Microsoft's documentation on installing OpenSSH for more details.

5. Set up SSH access for an administrative user. See Microsoft's documentation on the Administrative user to do this.

IMPORTANT

The public key used in the instructions must correspond to the private key you create later in the WMCO namespace that holds your secret. See the "Configuring a secret for the Windows Machine Config Operator" section for more details.

6. Install the **docker** container runtime on your Windows VM following the Microsoft documentation.

7. You must create a new firewall rule in the Windows VM that allows incoming connections for container logs. Run the following PowerShell command to create the firewall rule on TCP port 10250:

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

8. Clone the Windows VM so it is a reusable image. Follow the VMware documentation on how to clone an existing virtual machine for more details.

9. In the cloned Windows VM, run the Windows Sysprep tool:

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

1 Specify the path to your **unattend.xml** file.

NOTE

There is a limit on how many times you can run the **sysprep** command on a Windows image. Consult Microsoft's documentation for more information.

An example **unattend.xml** is provided, which maintains all the changes needed for the WMCO. You must modify this example; it cannot be used directly.

Example 5.1. Example **unattend.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="specialize">
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
International-Core" processorArchitecture="amd64"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <InputLocale>0409:00000409</InputLocale>
      <SystemLocale>en-US</SystemLocale>
      <UILanguage>en-US</UILanguage>
      <UILanguageFallback>en-US</UILanguageFallback>
      <UserLocale>en-US</UserLocale>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Security-SPP-UX" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
      <SkipAutoActivation>true</SkipAutoActivation>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
SQMApi" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
      <CEIPEnabled>0</CEIPEnabled>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
      <ComputerName>winhost</ComputerName> 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. Specify the **ComputerName**, which must follow the Kubernetes' names specification. These specifications also apply to Guest OS customization performed on the resulting template while creating new VMs.

2. Disable the automatic logon to avoid the security issue of leaving an open terminal with Administrator privileges at boot. This is the default value and must not be changed.

3. Replace the **MyPassword** placeholder with the password for the Administrator account. This prevents the built-in Administrator account from having a blank password by default. Follow Microsoft's best practices for choosing a password .

After the Sysprep tool has completed, the Windows VM will power off. You must not use or power on this VM anymore.

10. Convert the Windows VM to a template in vCenter .

5.3.2.1.1. Additional resources

- Configuring a secret for the Windows Machine Config Operator
- VMware vSphere infrastructure requirements

5.3.2.2. Enabling communication with the internal API server for the WMCO on vSphere

The Windows Machine Config Operator (WMCO) downloads the Ignition config files from the internal API server endpoint. You must enable communication with the internal API server so that your Windows virtual machine (VM) can download the Ignition config files, and the kubelet on the configured VM can only communicate with the internal API server.

Prerequisites

- You have installed a cluster on vSphere.

Procedure

- Add a new DNS entry for **api-int.<cluster_name>.<base_domain>** that points to the external API server URL **api.<cluster_name>.<base_domain>**. This can be a CNAME or an additional A record.

NOTE

The external API endpoint was already created as part of the initial cluster installation on vSphere.

5.3.3. Sample YAML for a Windows MachineSet object on vSphere

This sample YAML defines a Windows **MachineSet** object running on VMware vSphere that the Windows Machine Config Operator (WMCO) can react upon.

```
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 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. You can obtain the infrastructure ID by running the following command:

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

2 4 6 Specify the Windows machine set name. The machine set name cannot be more than 9 characters long, due to the way machine names are generated in vSphere.

7 Configure the machine set as a Windows machine.

8 Configure the Windows node as a compute machine.

9 Specify the size of the vSphere Virtual Machine Disk (VMDK).

NOTE

This parameter does not set the size of the Windows partition. You can resize the Windows partition by using the **unattend.xml** file or by creating the vSphere Windows virtual machine (VM) golden image with the required disk size.

10 Specify the vSphere VM network to deploy the machine set to. This VM network must be where other Linux compute machines reside in the cluster.

11 Specify the full path of the Windows vSphere VM template to use, such as **golden-images/windows-server-template**. The name must be unique.

IMPORTANT

Do not specify the original VM template. The VM template must remain off and must be cloned for new Windows machines. Starting the VM template configures the VM template as a VM on the platform, which prevents it from being used as a template that machine sets can apply configurations to.

12 The **windows-user-data** is created by the WMCO when the first Windows machine is configured. After that, the **windows-user-data** is available for all subsequent machine sets to consume.

13 Specify the vCenter Datacenter to deploy the machine set on.

14 Specify the vCenter Datastore to deploy the machine set on.

15 Specify the path to the vSphere VM folder in vCenter, such as **/dc1/vm/user-inst-5ddjd**.

16 Optional: Specify the vSphere resource pool for your Windows VMs.

17 Specify the vCenter server IP or fully qualified domain name.

5.3.4. Creating a machine set

In addition to the compute machine sets created by the installation program, you can create your own to dynamically manage the machine compute resources for specific workloads of your choice.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.

Procedure

1. Create a new YAML file that contains the machine set custom resource (CR) sample and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.

2. Optional: If you are not sure which value to set for a specific field, you can check an existing compute machine set from your cluster.

   a. To list the compute machine sets in your cluster, run the following command:

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

   Example output

   ```
   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. To view values of a specific compute machine set custom resource (CR), run the following command:

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

   Example output

   ```
   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 The cluster infrastructure ID.

2 A default node label.

NOTE

For clusters that have user-provisioned infrastructure, a compute machine set can only create **worker** and **infra** type machines.

3 The values in the **<providerSpec>** section of the compute machine set CR are platform-specific. For more information about **<providerSpec>** parameters in the CR, see the sample compute machine set CR configuration for your provider.

3. Create a **MachineSet** CR by running the following command:

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

Verification

- View the list of compute machine sets by running the following command:

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

Example output

```
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
```

When the new machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

5.3.5. Additional resources

- For more information on managing machine sets, see the *Machine management* section.

CHAPTER 6. SCHEDULING WINDOWS CONTAINER WORKLOADS

You can schedule Windows workloads to Windows compute nodes.

NOTE

The WMCO is not supported in clusters that use a cluster-wide proxy because the WMCO is not able to route traffic through the proxy connection for the workloads.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You are using a Windows container as the OS image with the Docker-formatted container runtime add-on enabled.
- You have created a Windows machine set.

IMPORTANT

Currently, the Docker-formatted container runtime is used in Windows nodes. Kubernetes is deprecating Docker as a container runtime; you can reference the Kubernetes documentation for more information in Docker deprecation. Containerd will be the new supported container runtime for Windows nodes in a future release of Kubernetes.

6.1. WINDOWS POD PLACEMENT

Before deploying your Windows workloads to the cluster, you must configure your Windows node scheduling so pods are assigned correctly. Since you have a machine hosting your Windows node, it is managed the same as a Linux-based node. Likewise, scheduling a Windows pod to the appropriate Windows node is completed similarly, using mechanisms like taints, tolerations, and node selectors.

With multiple operating systems, and the ability to run multiple Windows OS variants in the same cluster, you must map your Windows pods to a base Windows OS variant by using a **RuntimeClass** object. For example, if you have multiple Windows nodes running on different Windows Server container versions, the cluster could schedule your Windows pods to an incompatible Windows OS variant. You must have **RuntimeClass** objects configured for each Windows OS variant on your cluster. Using a **RuntimeClass** object is also recommended if you have only one Windows OS variant available in your cluster.

For more information, see Microsoft's documentation on Host and container version compatibility.

IMPORTANT

The container base image must be the same Windows OS version and build number that is running on the node where the conainer is to be scheduled.

Also, if you upgrade the Windows nodes from one version to another, for example going from 20H2 to 2022, you must upgrade your container base image to match the new version. For more information, see Windows container version compatibility.

Additional resources

- Controlling pod placement using the scheduler
- Controlling pod placement using node taints
- Placing pods on specific nodes using node selectors

6.2. CREATING A RUNTIMECLASS OBJECT TO ENCAPSULATE SCHEDULING MECHANISMS

Using a **RuntimeClass** object simplifies the use of scheduling mechanisms like taints and tolerations; you deploy a runtime class that encapsulates your taints and tolerations and then apply it to your pods to schedule them to the appropriate node. Creating a runtime class is also necessary in clusters that support multiple operating system variants.

Procedure

1. Create a **RuntimeClass** object YAML file. For example, **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) Specify the **RuntimeClass** object name, which is defined in the pods you want to be managed by this runtime class.

(2) Specify labels that must be present on nodes that support this runtime class. Pods using this runtime class can only be scheduled to a node matched by this selector. The node selector of the runtime class is merged with the existing node selector of the pod. Any conflicts prevent the pod from being scheduled to the node.

(3) Specify tolerations to append to pods, excluding duplicates, running with this runtime class during admission. This combines the set of nodes tolerated by the pod and the runtime class.

2. Create the **RuntimeClass** object:

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

For example:

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

3. Apply the **RuntimeClass** object to your pod to ensure it is scheduled to the appropriate operating system variant:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-windows-pod
spec:
  runtimeClassName: <runtime_class_name> (1)
...
```

(1) Specify the runtime class to manage the scheduling of your pod.

6.3. SAMPLE WINDOWS CONTAINER WORKLOAD DEPLOYMENT

You can deploy Windows container workloads to your cluster once you have a Windows compute node available.

NOTE

This sample deployment is provided for reference only.

Example Service object

```
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
```

Example Deployment object

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: win-webserver
  name: win-webserver
spec:
  selector:
    matchLabels:
      app: win-webserver
  replicas: 1
```

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

NOTE

When using the **mcr.microsoft.com/powershell:<tag>** container image, you must define the command as **pwsh.exe**. If you are using the **mcr.microsoft.com/windows/servercore:<tag>** container image, you must define the command as **powershell.exe**. For more information, see Microsoft's documentation.

6.4. SCALING A MACHINE SET MANUALLY

To add or remove an instance of a machine in a machine set, you can manually scale the machine set.

This guidance is relevant to fully automated, installer-provisioned infrastructure installations. Customized, user-provisioned infrastructure installations do not have machine sets.

Prerequisites

- Install an OpenShift Container Platform cluster and the **oc** command line.
- Log in to **oc** as a user with **cluster-admin** permission.

Procedure

1. View the machine sets that are in the cluster:

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

The machine sets are listed in the form of **<clusterid>-worker-<aws-region-az>**.

2. View the machines that are in the cluster:

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

3. Set the annotation on the machine that you want to delete:

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

4. Cordon and drain the node that you want to delete:

```
$ oc adm cordon <node_name>
$ oc adm drain <node_name>
```

5. Scale the machine set:

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

Or:

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

TIP

You can alternatively apply the following YAML to scale the machine set:

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

You can scale the machine set up or down. It takes several minutes for the new machines to be available.

Verification

- Verify the deletion of the intended machine:

```
$ oc get machines
```

CHAPTER 7. WINDOWS NODE UPGRADES

You can ensure your Windows nodes have the latest updates by upgrading the Windows Machine Config Operator (WMCO).

7.1. WINDOWS MACHINE CONFIG OPERATOR UPGRADES

When a new version of the Windows Machine Config Operator (WMCO) is released that is compatible with the current cluster version, the Operator is upgraded based on the upgrade channel and subscription approval strategy it was installed with when using the Operator Lifecycle Manager (OLM). The WMCO upgrade results in the Kubernetes components in the Windows machine being upgraded.

NOTE

If you are upgrading to a new version of the WMCO and want to use cluster monitoring, you must have the **openshift.io/cluster-monitoring=true** label present in the WMCO namespace. If you add the label to a pre-existing WMCO namespace, and there are already Windows nodes configured, restart the WMCO pod to allow monitoring graphs to display.

For a non-disruptive upgrade, the WMCO terminates the Windows machines configured by the previous version of the WMCO and recreates them using the current version. This is done by deleting the **Machine** object, which results in the drain and deletion of the Windows node. To facilitate an upgrade, the WMCO adds a version annotation to all the configured nodes. During an upgrade, a mismatch in version annotation results in the deletion and recreation of a Windows machine. To have minimal service disruptions during an upgrade, the WMCO only updates one Windows machine at a time.

IMPORTANT

The WMCO is only responsible for updating Kubernetes components, not for Windows operating system updates. You provide the Windows image when creating the VMs; therefore, you are responsible for providing an updated image. You can provide an updated Windows image by changing the image configuration in the **MachineSet** spec.

For more information on Operator upgrades using the Operator Lifecycle Manager (OLM), see Updating installed Operators.

CHAPTER 8. USING BRING-YOUR-OWN-HOST (BYOH) WINDOWS INSTANCES AS NODES

Bring-Your-Own-Host (BYOH) allows for users to repurpose Windows Server VMs and bring them to OpenShift Container Platform. BYOH Windows instances benefit users looking to mitigate major disruptions in the event that a Windows server goes offline.

8.1. CONFIGURING A BYOH WINDOWS INSTANCE

Creating a BYOH Windows instance requires creating a config map in the Windows Machine Config Operator (WMCO) namespace.

Prerequisites

Any Windows instances that are to be attached to the cluster as a node must fulfill the following requirements:

- The Docker container runtime must be installed on the instance.
- The instance must be on the same network as the Linux worker nodes in the cluster.
- Port 22 must be open and running an SSH server.
- The default shell for the SSH server must be the Windows Command shell, or **cmd.exe**.
- Port 10250 must be open for log collection.
- An administrator user is present with the private key used in the secret set as an authorized SSH key.
- If you are creating a BYOH Windows instance for an installer-provisioned infrastructure (IPI) AWS cluster, you must add a tag to the AWS instance that matches the **spec.template.spec.value.tag** value in the machine set for your worker nodes. For example, **kubernetes.io/cluster/<cluster_id>: owned** or **kubernetes.io/cluster/<cluster_id>: shared**.
- If you are creating a BYOH Windows instance on vSphere, communication with the internal API server must be enabled.
- The hostname of the instance must follow the RFC 1123 DNS label requirements, which include the following standards:
  - Contains only lowercase alphanumeric characters or '-'.
  - Starts with an alphanumeric character.
  - Ends with an alphanumeric character.

Procedure

1. Create a ConfigMap named **windows-instances** in the WMCO namespace that describes the Windows instances to be added.

NOTE

Format each entry in the config map's data section by using the address as the key while formatting the value as **username=<username>**.

Example config map

```
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) The address that the WMCO uses to reach the instance over SSH, either a DNS name or an IPv4 address. A DNS PTR record must exist for this address. It is recommended that you use a DNS name with your BYOH instance if your organization uses DHCP to assign IP addresses. If not, you need to update the **windows-instances** ConfigMap whenever the instance is assigned a new IP address.

(2) The name of the administrator user created in the prerequisites.

8.2. REMOVING BYOH WINDOWS INSTANCES

You can remove BYOH instances attached to the cluster by deleting the instance's entry in the config map. Deleting an instance reverts that instance back to its state prior to adding to the cluster. Any logs and container runtime artifacts are not added to these instances.

For an instance to be cleanly removed, it must be accessible with the current private key provided to WMCO. For example, to remove the **10.1.42.1** instance from the previous example, the config map would be changed to the following:

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

Deleting **windows-instances** is viewed as a request to deconstruct all Windows instances added as nodes.

CHAPTER 9. REMOVING WINDOWS NODES

You can remove a Windows node by deleting its host Windows machine.

9.1. DELETING A SPECIFIC MACHINE

You can delete a specific machine.

NOTE

You cannot delete a control plane machine.

Prerequisites

- Install an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.

Procedure

1. View the machines that are in the cluster and identify the one to delete:

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

   The command output contains a list of machines in the **<clusterid>-worker-<cloud_region>** format.

2. Delete the machine:

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

IMPORTANT

By default, the machine controller tries to drain the node that is backed by the machine until it succeeds. In some situations, such as with a misconfigured pod disruption budget, the drain operation might not be able to succeed in preventing the machine from being deleted. You can skip draining the node by annotating "machine.openshift.io/exclude-node-draining" in a specific machine. If the machine being deleted belongs to a machine set, a new machine is immediately created to satisfy the specified number of replicas.

CHAPTER 10. DISABLING WINDOWS CONTAINER WORKLOADS

You can disable the capability to run Windows container workloads by uninstalling the Windows Machine Config Operator (WMCO) and deleting the namespace that was added by default when you installed the WMCO.

10.1. UNINSTALLING THE WINDOWS MACHINE CONFIG OPERATOR

You can uninstall the Windows Machine Config Operator (WMCO) from your cluster.

Prerequisites

- Delete the Windows **Machine** objects hosting your Windows workloads.

Procedure

1. From the **Operators → OperatorHub** page, use the **Filter by keyword** box to search for **Red Hat Windows Machine Config Operator**.
2. Click the **Red Hat Windows Machine Config Operator**tile. The Operator tile indicates it is installed.
3. In the **Windows Machine Config Operator** descriptor page, click **Uninstall**.

10.2. DELETING THE WINDOWS MACHINE CONFIG OPERATOR NAMESPACE

You can delete the namespace that was generated for the Windows Machine Config Operator (WMCO) by default.

Prerequisites

- The WMCO is removed from your cluster.

Procedure

1. Remove all Windows workloads that were created in the **openshift-windows-machine-config-operator** namespace:

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

2. Verify that all pods in the **openshift-windows-machine-config-operator** namespace are deleted or are reporting a terminating state:

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

3. Delete the **openshift-windows-machine-config-operator** namespace:

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

Additional resources

- Deleting Operators from a cluster
- Removing Windows nodes