OpenShift Container Platform 4.7

Windows Container Support for OpenShift

Red Hat OpenShift for Windows Containers Guide

Last Updated: 2021-04-28

OpenShift Container Platform 4.7 Windows Container Support for OpenShift

Red Hat OpenShift for Windows Containers Guide

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

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. WINDOWS CONTAINER SUPPORT FOR RED HAT OPENSHIFT RELEASE NOTES

1.1. ABOUT WINDOWS CONTAINER SUPPORT FOR RED HAT OPENSHIFT

Windows Container Support for Red Hat OpenShift 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 Windows nodes available, you can run Windows container workloads in OpenShift Container Platform.

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

1.2. GETTING SUPPORT

You must have a subscription to receive support for the Red Hat WMCO. Deploying Windows container workloads in production clusters is not supported without a subscription. If you do not have a subscription, you can use the community WMCO, a distribution that lacks official support. Request support through the Red Hat Customer Portal .

1.3. RELEASE NOTES FOR RED HAT WINDOWS MACHINE CONFIG OPERATOR 2.0.0

This release of the WMCO provides bug fixes and enhancements for running Windows compute nodes in an OpenShift Container Platform cluster. The components of the WMCO 2.0.0 were released in RHBA-2021:0440.

WMCO supports self-managed clusters built using installer-provisioned infrastructure running on the following cloud providers:

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

The following Windows Server operating systems are supported in this release of the WMCO, depending on which platform your cluster is installed on:

WMCO platform	Windows Server version
AWS	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
Azure	Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019
vSphere	Windows Server Semi-Annual Channel (SAC): Windows Server 1909 with Microsoft patch KB4565351

IMPORTANT

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

Version 2.x of the WMCO is only compatible with OpenShift Container Platform 4.7.

1.3.1. New features and improvements

This release adds the following new features and improvements.

1.3.1.1. Support for clusters running on VMware vSphere

You can now run Windows nodes on a cluster installed on VMware vSphere version 6.5, 6.7, or 7.0. You can create a Windows **MachineSet** object on vSphere to host Windows Server compute nodes. For more information, see Creating a Windows **MachineSet** object on vSphere.

1.3.1.2. Enhanced Windows node monitoring

Windows nodes are now fully integrated with most of the monitoring capabilities provided by the web console. However, it is not possible to view workload graphs for pods running on Windows nodes in this release.

1.3.2. Known issues

- The filesystem graphs available in the web console do not display for Windows nodes. This is caused by changes in the filesystem queries. This will be fixed in a future release of WMCO. (BZ#1930347)
- The Prometheus windows_exporter used by the WMCO currently collects metrics through HTTP, so it is considered unsafe. You must ensure that only trusted users can retrieve metrics from the endpoint. The windows_exporter feature recently added support for HTTPS configuration, but this configuration has not been implemented for WMCO. Support for HTTPS configuration in the WMCO will be added in a future release.
- The kube-proxy service terminates unexpectedly after the load balancer is created if you create the load balancer after the Windows pods begin running. To avoid this issue, you must create the load balancer before the Windows pods status transitions to **Running**. (BZ#1939968)

CHAPTER 2. UNDERSTANDING WINDOWS CONTAINER WORKLOADS

Windows Container Support for Red Hat OpenShift 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).

Windows container workloads are supported for clusters running on the following cloud providers:

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

The following Windows Server operating systems are supported for OpenShift Container Platform 4.7:

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

For more information, see Microsoft's documentation on Windows Server channels.

NOTE

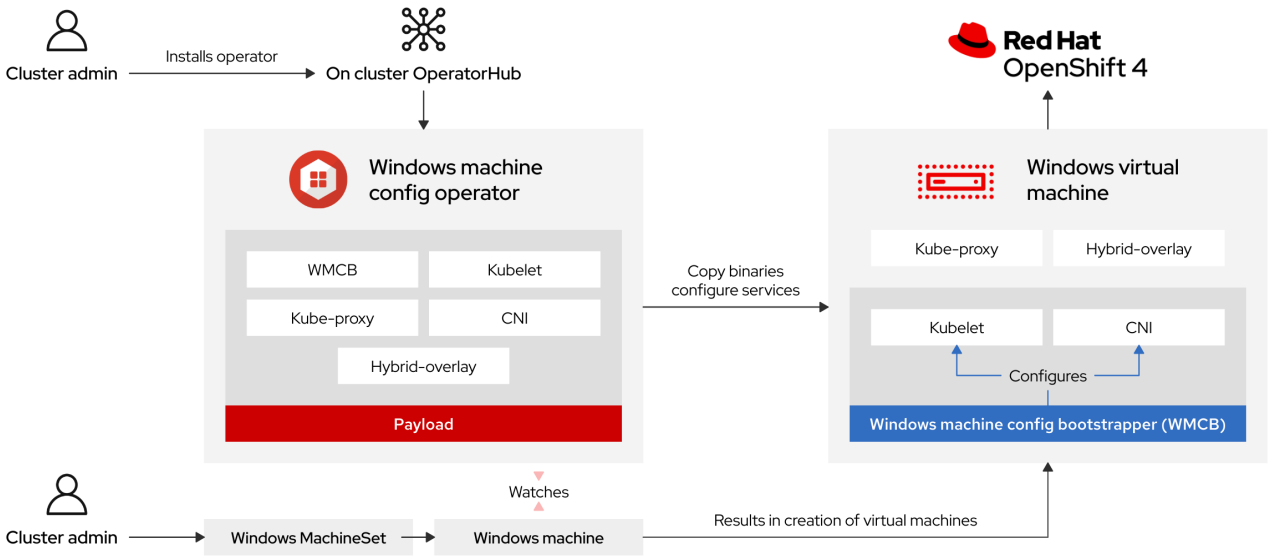
Multi-tenancy for clusters that have Windows nodes is not supported. Hostile multi-tenant usage introduces security concerns in all Kubernetes environments. Additional security features like pod security policies, or more fine-grained role-based access control (RBAC) for nodes, make exploits more difficult. 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.

2.1. 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 2.1. WMCO design



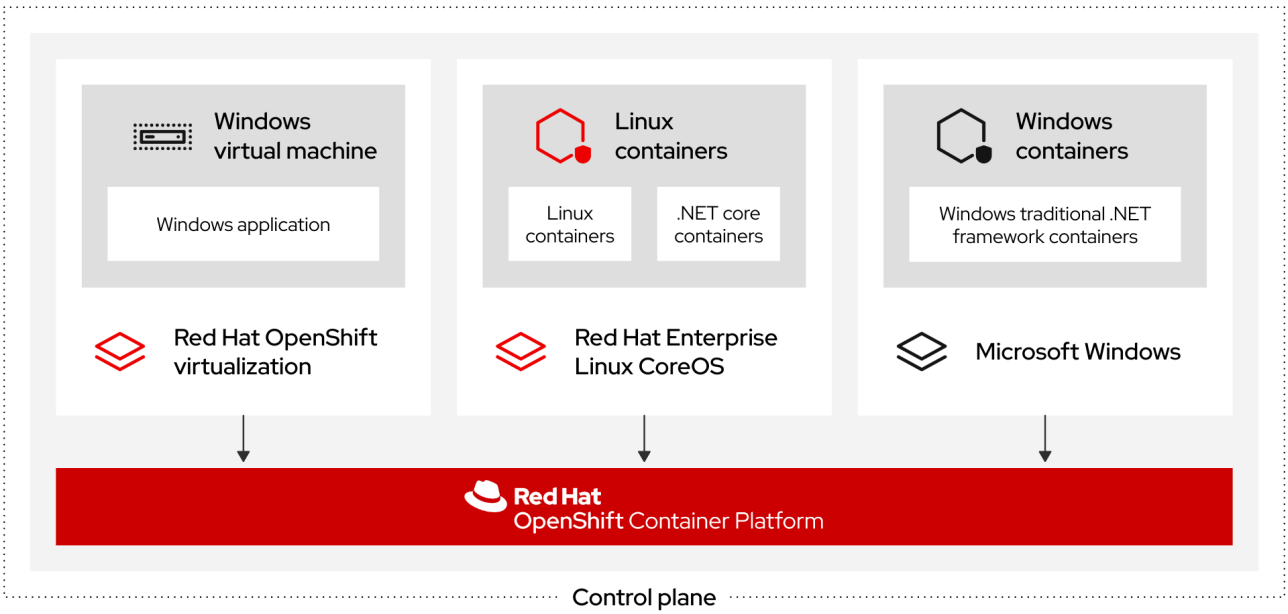
Before deploying Windows workloads, you must create a Windows compute node and have it join the cluster. The Windows node hosts the Windows workloads in a cluster, and can run alongside other Linux-based compute nodes. You can create a Windows compute node by creating a Windows machine set to host Windows 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.

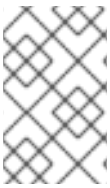
The WMCO watches for machines with the Windows label. After a Windows machine set is detected and its respective machines are provisioned, the WMCO configures the underlying Windows virtual machine (VM) so that it can join the cluster as a compute node.

Figure 2.2. Mixed Windows and Linux workloads



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.

2.2. 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) plug-ins	Exposes networking for Windows nodes.
Windows Machine Config Bootstrapper (WMCB)	Configures the kubelet and CNI plug-ins.
hybrid-overlay	Creates the OpenShift Container Platform Host Network Service (HNS).
kube-proxy	Maintains network rules on nodes allowing outside communication.

CHAPTER 3. 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.

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. Clusters installed with user-provisioned infrastructure are not supported for Windows container workloads.
- 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.

3.1. INSTALLING THE WINDOWS MACHINE CONFIG OPERATOR

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

3.1.1. 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).

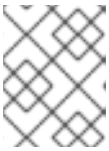
Procedure

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.

3.1.2. Installing the Windows Machine Config Operator using the CLI

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

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
```

3.2. 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.

Additional Resources

- Generating an SSH private key and adding it to the agent
- Adding Operators to a cluster .

CHAPTER 4. CREATING WINDOWS MACHINESET OBJECTS

4.1. 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.

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.

4.1.1. 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 4.7 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.7 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.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified 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.

4.1.2. 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.

4.1.3. Creating a machine set

In addition to the ones created by the installation program, you can create your own machine sets 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, as shown, and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.

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

   ```
   $ 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. Check values of a specific machine set:

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

   Example output

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

   (1) The cluster ID.

   (2) A default node label.

2. Create the new **MachineSet** CR:

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

3. View the list of machine sets:

```
$ 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.

4. After the new machine set is available, check status of the machine and the node that it references:

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

For example:

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

Example output

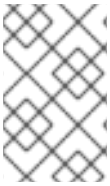
```
status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
    name: ip-10-0-133-18.ec2.internal
    uid: 71fb8d75-6d8f-11e9-9ff3-0e3f103c7cd8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

```
    reason: MachineCreationSucceeded
    status: "True"
    type: MachineCreation
  instanceId: i-09ca0701454124294
  instanceState: running
  kind: AWSMachineProviderStatus
```

5. View the new node and confirm that the new node has the label that you specified:

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

Review the command output and confirm that **node-role.kubernetes.io/<your_label>** is in the **LABELS** list.



NOTE

Any change to a machine set is not applied to existing machines owned by the machine set. For example, labels edited or added to an existing machine set are not propagated to existing machines and nodes associated with the machine set.

4.1.4. Additional resources

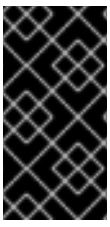
- For more information on managing machine sets, see the *Machine management* section.

4.2. 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.

4.2.1. 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 4.7 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.7 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.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified 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.

4.2.2. 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

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.

4.2.3. Creating a machine set

In addition to the ones created by the installation program, you can create your own machine sets 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, as shown, and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.

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

   ```
   $ 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. Check values of a specific machine set:

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

Example output

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

(1) The cluster ID.

(2) A default node label.

2. Create the new **MachineSet** CR:

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

3. View the list of machine sets:

```
$ 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.

4. After the new machine set is available, check status of the machine and the node that it references:

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

For example:

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

Example output

```
-
```

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

5. View the new node and confirm that the new node has the label that you specified:

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

Review the command output and confirm that **node-role.kubernetes.io/<your_label>** is in the **LABELS** list.

NOTE

Any change to a machine set is not applied to existing machines owned by the machine set. For example, labels edited or added to an existing machine set are not propagated to existing machines and nodes associated with the machine set.

4.2.4. Additional resources

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

4.3. 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.

4.3.1. 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 4.7 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.7 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.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified 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.

4.3.2. 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.

4.3.2.1. Creating the vSphere Windows VM golden image

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

Prerequisites

- You have installed a cluster on vSphere configured with hybrid networking using OVN-Kubernetes.
- You have defined a custom VXLAN port in your hybrid networking configuration to work around the pod-to-pod connectivity issue between hosts .

Procedure

1. Create the VM from an updated version of the Windows Server 1909 VM 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. This patch is not available for the **Windows Server 2019** VM image.
2. Create the **C:\Users\Administrator\.ssh\authorized_keys** file in the Windows VM containing the public key that corresponds to the private key that resides in the secret you created in the **openshift-windows-machine-config-operator** namespace. The private key of the secret was created when first installing the Windows Machine Config Operator (WMCO) to give OpenShift Container Platform access to Windows VMs. The **authorized_keys** file is used to configure SSH in the Windows VM.
3. Configure SSH on the Windows VM by running the following Powershell script:

```
# Powershell script to configure SSH on vSphere Windows VMs
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
$firewallRuleName = "ContainerLogsPort"
$containerLogsPort = "10250"
New-NetFirewallRule -DisplayName $firewallRuleName -Direction Inbound -Action Allow -
Protocol TCP -LocalPort $containerLogsPort -EdgeTraversalPolicy Allow
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
Install-Module -Force OpenSSHUtils
Set-Service -Name ssh-agent -StartupType 'Automatic'
Set-Service -Name sshd -StartupType 'Automatic'
Start-Service ssh-agent
Start-Service sshd
$pubKeyConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'#PubkeyAuthentication yes','PubkeyAuthentication yes'
$pubKeyConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
$passwordConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'#PasswordAuthentication yes','PasswordAuthentication yes'
```

```
$passwordConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
$authFileConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'AuthorizedKeysFile
__PROGRAMDATA__/ssh/administrators_authorized_keys','#AuthorizedKeysFile
__PROGRAMDATA__/ssh/administrators_authorized_keys'
$authFileConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
$pubKeyLocationConf = (Get-Content -path C:\ProgramData\ssh\sshd_config) -replace
'Match Group administrators','#Match Group administrators'
$pubKeyLocationConf | Set-Content -Path C:\ProgramData\ssh\sshd_config
Restart-Service sshd
New-item -Path $env:USERPROFILE -Name .ssh -ItemType Directory -force
```

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

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

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

   ```
   exclude-nics=
   ```

   This entry ensures the following:

   - The cloned vNIC generated on the Windows VM by the hybrid-overlay is not ignored.
   - The VM has an IP address in vCenter.

   b. The VMTools Windows service is running.

6. Pull all of the required Windows container base images needed for your applications. The images you pull are dependent on the Windows kernel you are using. See Microsoft's documentation on pulling Windows container base images for more information.

7. Run the Windows Sysprep tool on the Windows VM:

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

   An example **unattend.xml** is provided, which maintains all the changes needed for the WMCO. For example, the **unattend.xml** file must ensure the Administrator's home directory stays intact with the SSH public key. You must customize the example to fit your needs.

   Example 4.1. Example unattend.xml

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!--A sample unattend.xml which helps in setting admin password and running scripts on
   first boot-->
   <unattend xmlns="urn:schemas-microsoft-com:unattend">
     <settings pass="specialize">
       <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
   xmlns:xsi="http:// www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
   International-Core" processorArchitecture="amd64"
   publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
         <InputLocale>0409:00000409</InputLocale>
         <SystemLocale>en-US</SystemLocale>
         <UILanguage>en-US</UILanguage>
         <UILanguageFallback>en-US</UILanguageFallback>
   ```

```
        <UserLocale>en-US</UserLocale>
      </component>
      <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Security-SPP-UX" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
        <SkipAutoActivation>true</SkipAutoActivation>
      </component>
      <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
SQMApi" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
        <CEIPEnabled>0</CEIPEnabled>
      </component>
      <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
        <ComputerName>windows-host</ComputerName>
        <ProductKey>My_Product_key</ProductKey>
      </component>
    </settings>
    <settings pass="oobeSystem">
      <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
        <AutoLogon>
          <Password>
            <Value>MyPassword</Value>
            <PlainText>true</PlainText>
          </Password>
          <Enabled>true</Enabled>
          <Username>Administrator</Username>
        </AutoLogon>
        <OOBE>
          <HideEULAPage>true</HideEULAPage>
          <HideLocalAccountScreen>true</HideLocalAccountScreen>
          <HideOEMRegistrationScreen>true</HideOEMRegistrationScreen>
          <HideOnlineAccountScreens>true</HideOnlineAccountScreens>
          <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
          <NetworkLocation>Work</NetworkLocation>
          <ProtectYourPC>1</ProtectYourPC>
          <SkipMachineOOBE>true</SkipMachineOOBE>
          <SkipUserOOBE>true</SkipUserOOBE>
        </OOBE>
        <RegisteredOrganization>Organization</RegisteredOrganization>
        <RegisteredOwner>Owner</RegisteredOwner>
        <DisableAutoDaylightTimeSet>false</DisableAutoDaylightTimeSet>
        <TimeZone>Eastern Standard Time</TimeZone>
        <UserAccounts>
          <AdministratorPassword>
            <Value>MyPassword</Value>
            <PlainText>true</PlainText>
          </AdministratorPassword>
          <LocalAccounts>
```

```
            <LocalAccount wcm:action="add">
              <Description>Administrator</Description>
              <DisplayName>Administrator</DisplayName>
              <Group>Administrators</Group>
              <Name>Administrator</Name>
            </LocalAccount>
          </LocalAccounts>
        </UserAccounts>
      </component>
    </settings>
  </unattend>
```

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

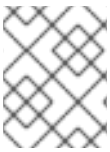
The Windows Machine Config Operator (WMCO) downloads the Ignition config files from the internal API server endpoint. You must enable communication with the internal API server so that your Windows virtual machine (VM) can download the Ignition config files, and the kubelet on the configured VM can only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

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

1 3 5 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 vSphere VM network to deploy the machine set to.

10 Specify the full path of the Windows vSphere VM template to use, such as **/Datacenter/vm/ocp4-llplx/windows-golden-image**. The name must be unique.

11

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.

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

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

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

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

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

4.3.4. Creating a machine set

In addition to the ones created by the installation program, you can create your own machine sets 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, as shown, and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.

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

   ```
   $ 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. Check values of a specific machine set:

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

   Example output

   ```
   -
   ```

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

1. The cluster ID.
2. A default node label.

2. Create the new **MachineSet** CR:

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

3. View the list of machine sets:

```
$ 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.

4. After the new machine set is available, check status of the machine and the node that it references:

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

For example:

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

Example output

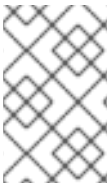
```
status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
```

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

5. View the new node and confirm that the new node has the label that you specified:

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

Review the command output and confirm that **node-role.kubernetes.io/<your_label>** is in the **LABELS** list.



NOTE

Any change to a machine set is not applied to existing machines owned by the machine set. For example, labels edited or added to an existing machine set are not propagated to existing machines and nodes associated with the machine set.

4.3.5. Additional resources

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

CHAPTER 5. SCHEDULING WINDOWS CONTAINER WORKLOADS

You can schedule Windows workloads to Windows compute nodes.

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.

5.1. 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**. 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.

Additional resources

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

5.2. 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.

5.3. 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:
          windowsOptions:
            runAsUserName: "ContainerAdministrator"
      nodeSelector:
        beta.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.

5.4. SCALING A MACHINE SET MANUALLY

If you must 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 does 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. Scale the machine set:

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

   Or:

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

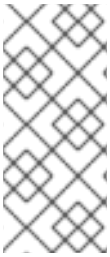
   You can scale the machine set up or down. It takes several minutes for the new machines to be available.

CHAPTER 6. WINDOWS NODE UPGRADES

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

6.1. 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 Upgrading installed Operators.

CHAPTER 7. REMOVING WINDOWS NODES

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

7.1. DELETING A SPECIFIC MACHINE

You can delete a specific machine.

Prerequisites

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

Procedure

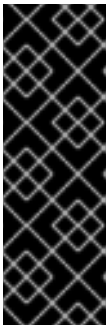
1. View the machines that are in the cluster and identify the one to delete:

```
$ 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 8. 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.

8.1. 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**.

8.2. 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.