



# OpenShift Container Platform 4.17

## Machine management

Adding and maintaining cluster machines



## OpenShift Container Platform 4.17 Machine management

---

Adding and maintaining cluster machines

## Legal Notice

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

Linux® 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

This document provides instructions for managing the machines that make up an OpenShift Container Platform cluster. Some tasks make use of the enhanced automatic machine management functions of an OpenShift Container Platform cluster and some tasks are manual. Not all tasks that are described in this document are available in all installation types.

## Table of Contents

<b>CHAPTER 1. OVERVIEW OF MACHINE MANAGEMENT .....</b>	<b>10</b>
1.1. MACHINE API OVERVIEW	10
1.2. MANAGING COMPUTE MACHINES	12
1.3. MANAGING CONTROL PLANE MACHINES	12
1.4. APPLYING AUTOSCALING TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER	13
1.5. ADDING COMPUTE MACHINES ON USER-PROVISIONED INFRASTRUCTURE	13
1.6. ADDING RHEL COMPUTE MACHINES TO YOUR CLUSTER	13
<b>CHAPTER 2. MANAGING COMPUTE MACHINES WITH THE MACHINE API .....</b>	<b>14</b>
2.1. CREATING A COMPUTE MACHINE SET ON AWS	14
2.1.1. Sample YAML for a compute machine set custom resource on AWS	14
2.1.2. Creating a compute machine set	16
2.1.3. Assigning machines to placement groups for Elastic Fabric Adapter instances by using machine sets	18
2.1.4. Machine set options for the Amazon EC2 Instance Metadata Service	19
2.1.4.1. Configuring IMDS by using machine sets	20
2.1.5. Machine sets that deploy machines as Dedicated Instances	20
2.1.5.1. Creating Dedicated Instances by using machine sets	20
2.1.6. Machine sets that deploy machines as Spot Instances	21
2.1.6.1. Creating Spot Instances by using compute machine sets	21
2.1.7. Adding a GPU node to an existing OpenShift Container Platform cluster	21
2.1.8. Deploying the Node Feature Discovery Operator	24
2.2. CREATING A COMPUTE MACHINE SET ON AZURE	26
2.2.1. Sample YAML for a compute machine set custom resource on Azure	26
2.2.2. Creating a compute machine set	28
2.2.3. Using the Azure Marketplace offering	30
2.2.4. Enabling Azure boot diagnostics	32
2.2.5. Machine sets that deploy machines as Spot VMs	33
2.2.5.1. Creating Spot VMs by using compute machine sets	34
2.2.6. Machine sets that deploy machines on Ephemeral OS disks	34
2.2.6.1. Creating machines on Ephemeral OS disks by using compute machine sets	34
2.2.7. Machine sets that deploy machines with ultra disks as data disks	35
2.2.7.1. Creating machines with ultra disks by using machine sets	36
2.2.7.2. Troubleshooting resources for machine sets that enable ultra disks	39
2.2.7.2.1. Incorrect ultra disk configuration	39
2.2.7.2.2. Unsupported disk parameters	39
2.2.7.2.3. Unable to delete disks	40
2.2.8. Enabling customer-managed encryption keys for a machine set	40
2.2.9. Configuring trusted launch for Azure virtual machines by using machine sets	41
2.2.10. Configuring Azure confidential virtual machines by using machine sets	42
2.2.11. Accelerated Networking for Microsoft Azure VMs	44
2.2.11.1. Limitations	44
2.2.12. Configuring Capacity Reservation by using machine sets	44
2.2.13. Adding a GPU node to an existing OpenShift Container Platform cluster	46
2.2.14. Deploying the Node Feature Discovery Operator	53
2.2.14.1. Enabling Accelerated Networking on an existing Microsoft Azure cluster	54
2.3. CREATING A COMPUTE MACHINE SET ON AZURE STACK HUB	55
2.3.1. Sample YAML for a compute machine set custom resource on Azure Stack Hub	55
2.3.2. Creating a compute machine set	57
2.3.3. Enabling Azure boot diagnostics	59
2.3.4. Enabling customer-managed encryption keys for a machine set	60
2.4. CREATING A COMPUTE MACHINE SET ON GCP	60

2.4.1. Sample YAML for a compute machine set custom resource on GCP	61
Values obtained by using the OpenShift CLI	61
2.4.2. Creating a compute machine set	63
2.4.3. Configuring persistent disk types by using machine sets	65
2.4.4. Configuring Confidential VM by using machine sets	66
2.4.5. Machine sets that deploy machines as preemptible VM instances	67
2.4.5.1. Creating preemptible VM instances by using compute machine sets	67
2.4.6. Configuring Shielded VM options by using machine sets	67
2.4.7. Enabling customer-managed encryption keys for a machine set	68
2.4.8. Enabling GPU support for a compute machine set	70
2.4.9. Adding a GPU node to an existing OpenShift Container Platform cluster	72
2.4.10. Deploying the Node Feature Discovery Operator	78
2.5. CREATING A COMPUTE MACHINE SET ON IBM CLOUD	79
2.5.1. Sample YAML for a compute machine set custom resource on IBM Cloud	79
2.5.2. Creating a compute machine set	81
2.6. CREATING A COMPUTE MACHINE SET ON IBM POWER VIRTUAL SERVER	83
2.6.1. Sample YAML for a compute machine set custom resource on IBM Power Virtual Server	83
2.6.2. Creating a compute machine set	85
2.7. CREATING A COMPUTE MACHINE SET ON NUTANIX	87
2.7.1. Sample YAML for a compute machine set custom resource on Nutanix	87
Values obtained by using the OpenShift CLI	87
2.7.2. Creating a compute machine set	89
2.7.3. Failure domains for Nutanix clusters	91
2.8. CREATING A COMPUTE MACHINE SET ON OPENSTACK	92
2.8.1. Sample YAML for a compute machine set custom resource on RHOSP	92
2.8.2. Sample YAML for a compute machine set custom resource that uses SR-IOV on RHOSP	93
2.8.3. Sample YAML for SR-IOV deployments where port security is disabled	96
2.8.4. Creating a compute machine set	98
2.9. CREATING A COMPUTE MACHINE SET ON VSphere	100
2.9.1. Sample YAML for a compute machine set custom resource on vSphere	100
2.9.2. Minimum required vCenter privileges for compute machine set management	102
2.9.3. Requirements for clusters with user-provisioned infrastructure to use compute machine sets	104
Obtaining the infrastructure ID	104
Satisfying vSphere credentials requirements	104
Satisfying Ignition configuration requirements	105
2.9.4. Creating a compute machine set	106
2.9.5. Adding tags to machines by using machine sets	109
2.10. CREATING A COMPUTE MACHINE SET ON BARE METAL	110
2.10.1. Sample YAML for a compute machine set custom resource on bare metal	110
2.10.2. Creating a compute machine set	111
<b>CHAPTER 3. MANUALLY SCALING A COMPUTE MACHINE SET .....</b>	<b>114</b>
3.1. PREREQUISITES	114
3.2. SCALING A COMPUTE MACHINE SET MANUALLY	114
3.3. THE COMPUTE MACHINE SET DELETION POLICY	115
3.4. ADDITIONAL RESOURCES	116
<b>CHAPTER 4. MODIFYING A COMPUTE MACHINE SET .....</b>	<b>117</b>
4.1. MODIFYING A COMPUTE MACHINE SET BY USING THE CLI	117
<b>CHAPTER 5. MACHINE PHASES AND LIFECYCLE .....</b>	<b>121</b>
5.1. MACHINE PHASES	121
5.2. THE MACHINE LIFECYCLE	121
5.3. DETERMINING THE PHASE OF A MACHINE	122

5.3.1. Determining the phase of a machine by using the CLI	122
5.3.2. Determining the phase of a machine by using the web console	123
5.4. ADDITIONAL RESOURCES	123
<b>CHAPTER 6. DELETING A MACHINE .....</b>	<b>124</b>
6.1. DELETING A SPECIFIC MACHINE	124
6.2. LIFECYCLE HOOKS FOR THE MACHINE DELETION PHASE	124
6.2.1. Terminology and definitions	125
6.2.2. Machine deletion processing order	125
6.2.3. Deletion lifecycle hook configuration	127
Example lifecycle hook configuration	128
6.2.4. Machine deletion lifecycle hook examples for Operator developers	129
Example use cases for preDrain lifecycle hooks	129
Example use cases for preTerminate lifecycle hooks	129
6.2.5. Quorum protection with machine lifecycle hooks	129
6.2.5.1. Control plane deletion with quorum protection processing order	130
6.3. ADDITIONAL RESOURCES	131
<b>CHAPTER 7. APPLYING AUTOSCALING TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER .....</b>	<b>132</b>
7.1. ABOUT THE CLUSTER AUTOSCALER	132
7.1.1. Configuring the cluster autoscaler	133
7.1.1.1. Cluster autoscaler resource definition	134
7.1.1.2. Configuring a priority expander for the cluster autoscaler	136
7.1.2. Deploying a cluster autoscaler	138
7.2. ABOUT THE MACHINE AUTOSCALER	138
7.2.1. Configuring machine autoscalers	138
7.2.1.1. Machine autoscaler resource definition	139
7.2.1.2. Deploying a machine autoscaler	140
7.3. DISABLING AUTOSCALING	140
7.3.1. Disabling a machine autoscaler	140
7.3.2. Disabling the cluster autoscaler	141
7.4. ADDITIONAL RESOURCES	142
<b>CHAPTER 8. CREATING INFRASTRUCTURE MACHINE SETS .....</b>	<b>143</b>
8.1. OPENSHIFT CONTAINER PLATFORM INFRASTRUCTURE COMPONENTS	143
8.2. CREATING INFRASTRUCTURE MACHINE SETS FOR PRODUCTION ENVIRONMENTS	144
8.2.1. Creating infrastructure machine sets for different clouds	144
8.2.1.1. Sample YAML for a compute machine set custom resource on AWS	144
8.2.1.2. Sample YAML for a compute machine set custom resource on Azure	146
8.2.1.3. Sample YAML for a compute machine set custom resource on Azure Stack Hub	149
8.2.1.4. Sample YAML for a compute machine set custom resource on IBM Cloud	151
8.2.1.5. Sample YAML for a compute machine set custom resource on GCP	153
Values obtained by using the OpenShift CLI	153
8.2.1.6. Sample YAML for a compute machine set custom resource on Nutanix	155
Values obtained by using the OpenShift CLI	155
8.2.1.7. Sample YAML for a compute machine set custom resource on RHOSP	158
8.2.1.8. Sample YAML for a compute machine set custom resource on vSphere	160
8.2.2. Creating a compute machine set	161
8.2.3. Creating an infrastructure node	163
8.2.4. Creating a machine config pool for infrastructure machines	165
8.3. ASSIGNING MACHINE SET RESOURCES TO INFRASTRUCTURE NODES	168
8.3.1. Binding infrastructure node workloads using taints and tolerations	168
8.4. MOVING RESOURCES TO INFRASTRUCTURE MACHINE SETS	171
8.4.1. Moving the router	172

8.4.2. Moving the default registry	173
8.4.3. Moving the monitoring solution	175
8.4.4. Moving the Vertical Pod Autoscaler Operator components	177
8.4.5. Moving the Cluster Resource Override Operator pods	181
<b>CHAPTER 9. ADDING RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER .</b>	<b>186</b>
9.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER	186
9.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES	186
9.2.1. Certificate signing requests management	188
9.3. PREPARING AN IMAGE FOR YOUR CLOUD	188
9.3.1. Listing latest available RHEL images on AWS	188
9.4. PREPARING THE MACHINE TO RUN THE PLAYBOOK	189
9.5. PREPARING A RHEL COMPUTE NODE	191
9.6. ATTACHING THE ROLE PERMISSIONS TO RHEL INSTANCE IN AWS	192
9.7. TAGGING A RHEL WORKER NODE AS OWNED OR SHARED	192
9.8. ADDING A RHEL COMPUTE MACHINE TO YOUR CLUSTER	192
9.9. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES	193
9.10. REQUIRED PARAMETERS FOR THE ANSIBLE HOSTS FILE	196
9.10.1. Optional: Removing RHCOS compute machines from a cluster	197
<b>CHAPTER 10. ADDING MORE RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER .....</b>	<b>198</b>
10.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER	198
10.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES	198
10.2.1. Certificate signing requests management	200
10.3. PREPARING AN IMAGE FOR YOUR CLOUD	200
10.3.1. Listing latest available RHEL images on AWS	200
10.4. PREPARING A RHEL COMPUTE NODE	201
10.5. ATTACHING THE ROLE PERMISSIONS TO RHEL INSTANCE IN AWS	202
10.6. TAGGING A RHEL WORKER NODE AS OWNED OR SHARED	203
10.7. ADDING MORE RHEL COMPUTE MACHINES TO YOUR CLUSTER	203
10.8. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES	204
10.9. REQUIRED PARAMETERS FOR THE ANSIBLE HOSTS FILE	207
<b>CHAPTER 11. MANAGING USER-PROVISIONED INFRASTRUCTURE MANUALLY .....</b>	<b>209</b>
11.1. ADDING COMPUTE MACHINES TO CLUSTERS WITH USER-PROVISIONED INFRASTRUCTURE MANUALLY	209
11.1.1. Adding compute machines to Amazon Web Services	209
11.1.2. Adding compute machines to Microsoft Azure	209
11.1.3. Adding compute machines to Azure Stack Hub	209
11.1.4. Adding compute machines to Google Cloud Platform	209
11.1.5. Adding compute machines to vSphere	209
11.1.6. Adding compute machines to bare metal	209
11.2. ADDING COMPUTE MACHINES TO AWS BY USING CLOUDFORMATION TEMPLATES	209
11.2.1. Prerequisites	209
11.2.2. Adding more compute machines to your AWS cluster by using CloudFormation templates	210
11.2.3. Approving the certificate signing requests for your machines	211
11.3. ADDING COMPUTE MACHINES TO VSphere MANUALLY	213
11.3.1. Prerequisites	214
11.3.2. Adding more compute machines to a cluster in vSphere	214
11.3.3. Approving the certificate signing requests for your machines	215
11.4. ADDING COMPUTE MACHINES TO BARE METAL	218
11.4.1. Prerequisites	218

11.4.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	218
11.4.2.1. Creating RHCOS machines using an ISO image	218
11.4.2.2. Creating RHCOS machines by PXE or iPXE booting	220
11.4.3. Approving the certificate signing requests for your machines	222
<b>CHAPTER 12. MANAGING CONTROL PLANE MACHINES .....</b>	<b>226</b>
12.1. ABOUT CONTROL PLANE MACHINE SETS	226
12.1.1. Control Plane Machine Set Operator overview	226
12.1.1.1. Control Plane Machine Set Operator limitations	226
12.1.2. Additional resources	227
12.2. GETTING STARTED WITH CONTROL PLANE MACHINE SETS	227
12.2.1. Supported cloud providers	228
12.2.2. Checking the control plane machine set custom resource state	228
12.2.3. Activating the control plane machine set custom resource	229
12.2.4. Creating a control plane machine set custom resource	230
12.3. MANAGING CONTROL PLANE MACHINES WITH CONTROL PLANE MACHINE SETS	231
12.3.1. Updating the control plane configuration	232
12.3.1.1. Automatic updates to the control plane configuration	232
12.3.1.2. Manual updates to the control plane configuration	232
12.3.2. Replacing a control plane machine	233
12.3.3. Additional resources	234
12.4. CONTROL PLANE MACHINE SET CONFIGURATION	234
12.4.1. Sample YAML for a control plane machine set custom resource	234
12.4.2. Provider-specific configuration options	235
12.5. CONFIGURATION OPTIONS FOR CONTROL PLANE MACHINES	236
12.5.1. Control plane configuration options for Amazon Web Services	236
12.5.1.1. Sample YAML for configuring Amazon Web Services clusters	236
12.5.1.1.1. Sample AWS provider specification	236
12.5.1.1.2. Sample AWS failure domain configuration	238
12.5.1.2. Enabling Amazon Web Services features for control plane machines	239
12.5.1.2.1. Restricting the API server to private	240
12.5.1.2.2. Changing the Amazon Web Services instance type by using a control plane machine set	240
12.5.1.2.3. Assigning machines to placement groups for Elastic Fabric Adapter instances by using machine sets	241
12.5.1.2.4. Machine set options for the Amazon EC2 Instance Metadata Service	242
12.5.1.2.4.1. Configuring IMDS by using machine sets	242
12.5.1.2.5. Machine sets that deploy machines as Dedicated Instances	243
12.5.1.2.5.1. Creating Dedicated Instances by using machine sets	243
12.5.1.2. Control plane configuration options for Microsoft Azure	243
12.5.2.1. Sample YAML for configuring Microsoft Azure clusters	243
12.5.2.1.1. Sample Azure provider specification	244
12.5.2.1.2. Sample Azure failure domain configuration	245
12.5.2.2. Enabling Microsoft Azure features for control plane machines	246
12.5.2.2.1. Restricting the API server to private	246
12.5.2.2.2. Using the Azure Marketplace offering	247
12.5.2.2.3. Enabling Azure boot diagnostics	249
12.5.2.2.4. Machine sets that deploy machines with ultra disks as data disks	250
12.5.2.2.4.1. Creating machines with ultra disks by using machine sets	250
12.5.2.2.4.2. Troubleshooting resources for machine sets that enable ultra disks	253
12.5.2.2.4.2.1. Incorrect ultra disk configuration	253
12.5.2.2.4.2.2. Unsupported disk parameters	254
12.5.2.2.4.2.3. Unable to delete disks	254
12.5.2.2.5. Enabling customer-managed encryption keys for a machine set	254

12.5.2.2.6. Configuring trusted launch for Azure virtual machines by using machine sets	255
12.5.2.2.7. Configuring Azure confidential virtual machines by using machine sets	256
12.5.2.2.8. Accelerated Networking for Microsoft Azure VMs	258
12.5.2.2.8.1. Limitations	258
12.5.2.2.9. Configuring Capacity Reservation by using machine sets	259
12.5.2.2.9.1. Enabling Accelerated Networking on an existing Microsoft Azure cluster	260
12.5.3. Control plane configuration options for Google Cloud Platform	260
12.5.3.1. Sample YAML for configuring Google Cloud Platform clusters	260
12.5.3.1.1. Sample GCP provider specification	260
Values obtained by using the OpenShift CLI	261
12.5.3.1.2. Sample GCP failure domain configuration	262
12.5.3.2. Enabling Google Cloud Platform features for control plane machines	263
12.5.3.2.1. Configuring persistent disk types by using machine sets	263
12.5.3.2.2. Configuring Confidential VM by using machine sets	264
12.5.3.2.3. Configuring Shielded VM options by using machine sets	265
12.5.3.2.4. Enabling customer-managed encryption keys for a machine set	266
12.5.4. Control plane configuration options for Nutanix	267
12.5.4.1. Sample YAML for configuring Nutanix clusters	267
12.5.4.1.1. Sample Nutanix provider specification	267
Values obtained by using the OpenShift CLI	267
12.5.4.1.2. Failure domains for Nutanix clusters	269
12.5.5. Control plane configuration options for Red Hat OpenStack Platform	270
12.5.5.1. Sample YAML for configuring Red Hat OpenStack Platform (RHOSP) clusters	270
12.5.5.1.1. Sample RHOSP provider specification	270
12.5.5.1.2. Sample RHOSP failure domain configuration	271
12.5.5.2. Enabling Red Hat OpenStack Platform (RHOSP) features for control plane machines	272
12.5.5.2.1. Changing the RHOSP compute flavor by using a control plane machine set	272
12.5.6. Control plane configuration options for VMware vSphere	272
12.5.6.1. Sample YAML for configuring VMware vSphere clusters	272
12.5.6.1.1. Sample VMware vSphere provider specification	273
12.5.6.1.2. Sample VMware vSphere failure domain configuration	274
12.5.6.2. Enabling VMware vSphere features for control plane machines	276
12.5.6.2.1. Adding tags to machines by using machine sets	276
12.6. CONTROL PLANE RESILIENCY AND RECOVERY	277
12.6.1. High availability and fault tolerance with failure domains	277
12.6.1.1. Failure domain platform support and configuration	277
12.6.1.2. Balancing control plane machines	278
12.6.2. Recovery of failed control plane machines	278
12.6.3. Quorum protection with machine lifecycle hooks	279
12.6.3.1. Control plane deletion with quorum protection processing order	279
12.7. TROUBLESHOOTING THE CONTROL PLANE MACHINE SET	280
12.7.1. Checking the control plane machine set custom resource state	280
12.7.2. Adding a missing Azure internal load balancer	281
12.7.3. Recovering a degraded etcd Operator	282
12.7.4. Upgrading clusters that run on RHOSP	282
12.7.4.1. Configuring RHOSP clusters that have machines with root volume availability zones after an upgrade	283
12.7.4.2. Configuring RHOSP clusters that have control plane machines with availability zones after an upgrade	284
12.8. DISABLING THE CONTROL PLANE MACHINE SET	286
12.8.1. Deleting the control plane machine set	286
12.8.2. Checking the control plane machine set custom resource state	287
12.8.3. Re-enabling the control plane machine set	287

<b>CHAPTER 13. MANAGING MACHINES WITH THE CLUSTER API .....</b>	<b>288</b>
13.1. ABOUT THE CLUSTER API	288
13.1.1. Cluster API overview	288
13.1.1.1. Cluster API benefits	288
13.1.1.2. Cluster API limitations	288
13.1.2. Cluster API architecture	289
13.1.2.1. The Cluster CAPI Operator	289
13.1.2.2. Cluster API primary resources	290
13.2. GETTING STARTED WITH THE CLUSTER API	290
13.2.1. Creating the Cluster API primary resources	290
13.2.1.1. Obtaining the cluster ID value	291
13.2.1.2. Creating the Cluster API cluster resource	291
13.2.1.3. Creating a Cluster API infrastructure resource	292
13.2.1.4. Creating a Cluster API machine template	294
13.2.1.5. Creating a Cluster API compute machine set	295
13.3. MANAGING MACHINES WITH THE CLUSTER API	297
13.3.1. Modifying a Cluster API machine template	297
13.3.2. Modifying a compute machine set by using the CLI	299
13.4. CLUSTER API CONFIGURATION	302
13.4.1. Sample YAML for a Cluster API cluster resource	303
13.4.2. Provider-specific configuration options	303
13.5. CONFIGURATION OPTIONS FOR CLUSTER API MACHINES	304
13.5.1. Cluster API configuration options for Amazon Web Services	304
13.5.1.1. Sample YAML for configuring Amazon Web Services clusters	304
13.5.1.1.1. Sample YAML for a Cluster API infrastructure resource on Amazon Web Services	304
13.5.1.1.2. Sample YAML for a Cluster API machine template resource on Amazon Web Services	305
13.5.1.1.3. Sample YAML for a Cluster API compute machine set resource on Amazon Web Services	305
13.5.2. Cluster API configuration options for Google Cloud Platform	306
13.5.2.1. Sample YAML for configuring Google Cloud Platform clusters	306
13.5.2.1.1. Sample YAML for a Cluster API infrastructure resource on Google Cloud Platform	306
13.5.2.1.2. Sample YAML for a Cluster API machine template resource on Google Cloud Platform	307
13.5.2.1.3. Sample YAML for a Cluster API compute machine set resource on Google Cloud Platform	308
13.5.3. Cluster API configuration options for VMware vSphere	308
13.5.3.1. Sample YAML for configuring VMware vSphere clusters	309
13.5.3.1.1. Sample YAML for a Cluster API infrastructure resource on VMware vSphere	309
13.5.3.1.2. Sample YAML for a Cluster API machine template resource on VMware vSphere	309
13.5.3.1.3. Sample YAML for a Cluster API compute machine set resource on VMware vSphere	311
13.6. TROUBLESHOOTING CLUSTERS THAT USE THE CLUSTER API	312
13.6.1. Referencing the intended objects when using the CLI	312
Cause	312
Consequences	312
Workaround	313
<b>CHAPTER 14. DEPLOYING MACHINE HEALTH CHECKS .....</b>	<b>314</b>
14.1. ABOUT MACHINE HEALTH CHECKS	314
14.1.1. Limitations when deploying machine health checks	315
14.2. SAMPLE MACHINEHEALTHCHECK RESOURCE	315
14.2.1. Short-circuiting machine health check remediation	316
14.2.1.1. Setting maxUnhealthy by using an absolute value	317
14.2.1.2. Setting maxUnhealthy by using percentages	317
14.3. CREATING A MACHINE HEALTH CHECK RESOURCE	317
14.4. ABOUT POWER-BASED REMEDIATION OF BARE METAL	317
14.4.1. MachineHealthChecks on bare metal	318

14.4.2. Understanding the annotation-based remediation process	318
14.4.3. Understanding the metal3-based remediation process	319
14.4.4. Creating a MachineHealthCheck resource for bare metal	319
14.4.5. Troubleshooting issues with power-based remediation	321



# CHAPTER 1. OVERVIEW OF MACHINE MANAGEMENT

You can use machine management to flexibly work with underlying infrastructure such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Red Hat OpenStack Platform (RHOSP), and VMware vSphere to manage the OpenShift Container Platform cluster. You can control the cluster and perform auto-scaling, such as scaling up and down the cluster based on specific workload policies.

It is important to have a cluster that adapts to changing workloads. The OpenShift Container Platform cluster can horizontally scale up and down when the load increases or decreases.

Machine management is implemented as a [custom resource definition](#) (CRD). A CRD object defines a new unique object **Kind** in the cluster and enables the Kubernetes API server to handle the object's entire lifecycle.

The Machine API Operator provisions the following resources:

- **MachineSet**
- **Machine**
- **ClusterAutoscaler**
- **MachineAutoscaler**
- **MachineHealthCheck**

## 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 4.17 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.17 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 compute node might define a specific machine type and required metadata.

### Machine sets

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



## WARNING

Control plane machines cannot be managed by compute machine sets.

Control plane machine sets provide management capabilities for supported control plane machines that are similar to what compute machine sets provide for compute machines.

For more information, see “[Managing control plane machines](#)”.

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 compute machine set API. You can use the cluster autoscaler to manage your cluster in the following ways:

- Set cluster-wide scaling limits for resources such as cores, nodes, memory, and GPU
- Set the priority so that the cluster prioritizes pods and new nodes are not brought online for less important pods
- 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 compute machine set is scoped to a single zone, so the installation program sends out compute 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

### Additional resources

- [Machine phases and lifecycle](#)

## 1.2. MANAGING COMPUTE MACHINES

As a cluster administrator, you can perform the following actions:

- Create a compute machine set for the following cloud providers:
  - [AWS](#)
  - [Azure](#)
  - [Azure Stack Hub](#)
  - [GCP](#)
  - [IBM Cloud](#)
  - [IBM Power Virtual Server](#)
  - [Nutanix](#)
  - [RHOSP](#)
  - [vSphere](#)
- Create a machine set for a bare metal deployment: [Creating a compute machine set on bare metal](#)
- [Manually scale a compute machine set](#) by adding or removing a machine from the compute machine set.
- [Modify a compute machine set](#) through the **MachineSet** YAML configuration file.
- [Delete a machine](#).
- [Create infrastructure compute machine sets](#).
- Configure and deploy a [machine health check](#) to automatically fix damaged machines in a machine pool.

## 1.3. MANAGING CONTROL PLANE MACHINES

As a cluster administrator, you can perform the following actions:

- [Update your control plane configuration](#) with a control plane machine set for the following cloud providers:
  - [Amazon Web Services](#)
  - [Google Cloud Platform](#)
  - [Microsoft Azure](#)
  - [Nutanix](#)
  - [Red Hat OpenStack Platform \(RHOSP\)](#)
  - [VMware vSphere](#)

- Configure and deploy a [machine health check](#) to automatically recover unhealthy control plane machines.

## 1.4. APPLYING AUTOSCALING TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

You can automatically scale your OpenShift Container Platform cluster to ensure flexibility for changing workloads. To [autoscale](#) your cluster, you must first deploy a cluster autoscaler, and then deploy a machine autoscaler for each compute machine set.

- The [\*cluster autoscaler\*](#) increases and decreases the size of the cluster based on deployment needs.
- The [\*machine autoscaler\*](#) adjusts the number of machines in the compute machine sets that you deploy in your OpenShift Container Platform cluster.

## 1.5. ADDING COMPUTE MACHINES ON USER-PROVISIONED INFRASTRUCTURE

User-provisioned infrastructure is an environment where you can deploy infrastructure such as compute, network, and storage resources that host the OpenShift Container Platform. You can [add compute machines](#) to a cluster on user-provisioned infrastructure during or after the installation process.

## 1.6. ADDING RHEL COMPUTE MACHINES TO YOUR CLUSTER

As a cluster administrator, you can perform the following actions:

- [Add Red Hat Enterprise Linux \(RHEL\) compute machines](#), also known as worker machines, to a user-provisioned infrastructure cluster or an installation-provisioned infrastructure cluster.
- [Add more Red Hat Enterprise Linux \(RHEL\) compute machines](#) to an existing cluster.

# CHAPTER 2. MANAGING COMPUTE MACHINES WITH THE MACHINE API

## 2.1. CREATING A COMPUTE MACHINE SET ON AWS

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



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.1.1. Sample YAML for a compute machine set custom resource on AWS

The sample YAML defines a compute machine set that runs in the **us-east-1a** Amazon Web Services (AWS) Local Zone and creates nodes that are labeled with **node-role.kubernetes.io/<role>: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ①
  name: <infrastructure_id>-<role>-<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>-<role>-<zone> ④
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ⑤
        machine.openshift.io/cluster-api-machine-role: <role> ⑥
        machine.openshift.io/cluster-api-machine-type: <role> ⑦
```

```

machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 8
spec:
  metadata:
    labels:
      node-role.kubernetes.io/<role>: "" 9
  providerSpec:
    value:
      ami:
        id: ami-046fe691f52a953f9 10
      apiVersion: machine.openshift.io/v1beta1
      blockDevices:
        - ebs:
            iops: 0
            volumeSize: 120
            volumeType: gp2
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile 11
      instanceType: m6i.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: <zone> 12
        region: <region> 13
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg 14
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-<zone> 15
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> 16
          value: owned
        - name: <custom_tag_name> 17
          value: <custom_tag_value> 18
      userDataSecret:
        name: worker-user-data

```

**1 3 5 11 14 16** Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

**2 4 8** Specify the infrastructure ID, role node label, and zone.

**6 7 9** Specify the role node label to add.

**10** Specify a valid Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) for your AWS zone for your OpenShift Container Platform nodes. If you want to use an AWS Marketplace

If you are using your OpenShift Container Platform cluster, or you want to use an AWS Marketplace image, you must complete the OpenShift Container Platform subscription from the [AWS Marketplace](#) to obtain an AMI ID for your region.

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}{"\n"}' \
get machineset/<infrastructure_id>-<role>-<zone>
```

- 17 18** Optional: Specify custom tag data for your cluster. For example, you might add an admin contact email address by specifying a **name:value** pair of **Email:admin-email@example.com**.



#### NOTE

Custom tags can also be specified during installation in the **install-config.yml** file. If the **install-config.yml** file and the machine set include a tag with the same **name** data, the value for the tag from the machine set takes priority over the value for the tag in the **install-config.yml** file.

- 12** Specify the zone, for example, **us-east-1a**.
- 13** Specify the region, for example, **us-east-1**.
- 15** Specify the infrastructure ID and zone.

### 2.1.2. Creating a compute 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 compute 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
```

- If you need compute machine sets in other availability zones, repeat this process to create more compute machine sets.

## 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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

### 2.1.3. Assigning machines to placement groups for Elastic Fabric Adapter instances by using machine sets

You can configure a machine set to deploy machines on [Elastic Fabric Adapter](#) (EFA) instances within an existing AWS placement group.

EFA instances do not require placement groups, and you can use placement groups for purposes other than configuring an EFA. This example uses both to demonstrate a configuration that can improve network performance for machines within the specified placement group.

#### Prerequisites

- You created a placement group in the AWS console.



#### NOTE

Ensure that the [rules and limitations](#) for the type of placement group that you create are compatible with your intended use case.

#### Procedure

- In a text editor, open the YAML file for an existing machine set or create a new one.
- Edit the following lines under the **providerSpec** field:

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

```

spec:
template:
spec:
providerSpec:
value:
  instanceType: <supported_instance_type> ①
  networkInterfaceType: EFA ②
placement:
  availabilityZone: <zone> ③
  region: <region> ④
placementGroupName: <placement_group> ⑤
placementGroupPartition: <placement_group_partition_number> ⑥
# ...

```

- ① Specify an instance type that [supports EFAs](#).
- ② Specify the **EFA** network interface type.
- ③ Specify the zone, for example, **us-east-1a**.
- ④ Specify the region, for example, **us-east-1**.
- ⑤ Specify the name of the existing AWS placement group to deploy machines in.
- ⑥ Optional: Specify the partition number of the existing AWS placement group to deploy machines in.

## Verification

- In the AWS console, find a machine that the machine set created and verify the following in the machine properties:
  - The placement group field has the value that you specified for the **placementGroupName** parameter in the machine set.
  - The partition number field has the value that you specified for the **placementGroupPartition** parameter in the machine set.
  - The interface type field indicates that it uses an EFA.

### 2.1.4. Machine set options for the Amazon EC2 Instance Metadata Service

You can use machine sets to create machines that use a specific version of the Amazon EC2 Instance Metadata Service (IMDS). Machine sets can create machines that allow the use of both IMDSv1 and [IMDSv2](#) or machines that require the use of IMDSv2.



#### NOTE

Using IMDSv2 is only supported on AWS clusters that were created with OpenShift Container Platform version 4.7 or later.

To deploy new compute machines with your preferred IMDS configuration, create a compute machine set YAML file with the appropriate values. You can also edit an existing machine set to create new machines with your preferred IMDS configuration when the machine set is scaled up.



## IMPORTANT

Before configuring a machine set to create machines that require IMDSv2, ensure that any workloads that interact with the AWS metadata service support IMDSv2.

### 2.1.4.1. Configuring IMDS by using machine sets

You can specify whether to require the use of IMDSv2 by adding or editing the value of **metadataServiceOptions.authentication** in the machine set YAML file for your machines.

#### Prerequisites

- To use IMDSv2, your AWS cluster must have been created with OpenShift Container Platform version 4.7 or later.

#### Procedure

- Add or edit the following lines under the **providerSpec** field:

```
providerSpec:  
  value:  
    metadataServiceOptions:  
      authentication: Required ①
```

- ① To require IMDSv2, set the parameter value to **Required**. To allow the use of both IMDSv1 and IMDSv2, set the parameter value to **Optional**. If no value is specified, both IMDSv1 and IMDSv2 are allowed.

### 2.1.5. Machine sets that deploy machines as Dedicated Instances

You can create a machine set running on AWS that deploys machines as Dedicated Instances. Dedicated Instances run in a virtual private cloud (VPC) on hardware that is dedicated to a single customer. These Amazon EC2 instances are physically isolated at the host hardware level. The isolation of Dedicated Instances occurs even if the instances belong to different AWS accounts that are linked to a single payer account. However, other instances that are not dedicated can share hardware with Dedicated Instances if they belong to the same AWS account.

Instances with either public or dedicated tenancy are supported by the Machine API. Instances with public tenancy run on shared hardware. Public tenancy is the default tenancy. Instances with dedicated tenancy run on single-tenant hardware.

### 2.1.5.1. Creating Dedicated Instances by using machine sets

You can run a machine that is backed by a Dedicated Instance by using Machine API integration. Set the **tenancy** field in your machine set YAML file to launch a Dedicated Instance on AWS.

#### Procedure

- Specify a dedicated tenancy under the **providerSpec** field:

```
providerSpec:  
  placement:  
    tenancy: dedicated
```

## 2.1.6. Machine sets that deploy machines as Spot Instances

You can save on costs by creating a compute machine set running on AWS that deploys machines as non-guaranteed Spot Instances. Spot Instances utilize unused AWS EC2 capacity and are less expensive than On-Demand Instances. You can use Spot Instances for workloads that can tolerate interruptions, such as batch or stateless, horizontally scalable workloads.

AWS EC2 can terminate a Spot Instance at any time. AWS gives a two-minute warning to the user when an interruption occurs. OpenShift Container Platform begins to remove the workloads from the affected instances when AWS issues the termination warning.

Interruptions can occur when using Spot Instances for the following reasons:

- The instance price exceeds your maximum price
- The demand for Spot Instances increases
- The supply of Spot Instances decreases

When AWS terminates an instance, a termination handler running on the Spot Instance node deletes the machine resource. To satisfy the compute machine set **replicas** quantity, the compute machine set creates a machine that requests a Spot Instance.

### 2.1.6.1. Creating Spot Instances by using compute machine sets

You can launch a Spot Instance on AWS by adding **spotMarketOptions** to your compute machine set YAML file.

#### Procedure

- Add the following line under the **providerSpec** field:

```
providerSpec:  
  value:  
    spotMarketOptions: {}
```

You can optionally set the **spotMarketOptions.maxPrice** field to limit the cost of the Spot Instance. For example you can set **maxPrice: '2.50'**.

If the **maxPrice** is set, this value is used as the hourly maximum spot price. If it is not set, the maximum price defaults to charge up to the On-Demand Instance price.



#### NOTE

It is strongly recommended to use the default On-Demand price as the **maxPrice** value and to not set the maximum price for Spot Instances.

## 2.1.7. Adding a GPU node to an existing OpenShift Container Platform cluster

You can copy and modify a default compute machine set configuration to create a GPU-enabled machine set and machines for the AWS EC2 cloud provider.

For more information about the supported instance types, see the following NVIDIA documentation:

- [NVIDIA GPU Operator Community support matrix](#)

- NVIDIA AI Enterprise support matrix

## Procedure

- View the existing nodes, machines, and machine sets by running the following command. Note that each node is an instance of a machine definition with a specific AWS region and OpenShift Container Platform role.

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-52-50.us-east-2.compute.internal	Ready	worker	3d17h	v1.30.3
ip-10-0-58-24.us-east-2.compute.internal	Ready	control-plane,master	3d17h	v1.30.3
ip-10-0-68-148.us-east-2.compute.internal	Ready	worker	3d17h	v1.30.3
ip-10-0-68-68.us-east-2.compute.internal	Ready	control-plane,master	3d17h	v1.30.3
ip-10-0-72-170.us-east-2.compute.internal	Ready	control-plane,master	3d17h	v1.30.3
ip-10-0-74-50.us-east-2.compute.internal	Ready	worker	3d17h	v1.30.3

- View the machines and machine sets that exist in the **openshift-machine-api** namespace by running the following command. Each compute machine set is associated with a different availability zone within the AWS region. The installer automatically load balances compute machines across availability zones.

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

### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
preserve-dsoc12r4-ktjfc-worker-us-east-2a	1	1	1	1	3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b	2	2	2	2	3d11h

- View the machines that exist in the **openshift-machine-api** namespace by running the following command. At this time, there is only one compute machine per machine set, though a compute machine set could be scaled to add a node in a particular region and zone.

```
$ oc get machines -n openshift-machine-api | grep worker
```

### Example output

preserve-dsoc12r4-ktjfc-worker-us-east-2a-dts8r	Running	m5.xlarge	us-east-2	us-east-2a	3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b-dkv7w	Running	m5.xlarge	us-east-2	us-east-2b	3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b-k58cw	Running	m5.xlarge	us-east-2	us-east-2b	3d11h

- Make a copy of one of the existing compute **MachineSet** definitions and output the result to a JSON file by running the following command. This will be the basis for the GPU-enabled compute machine set definition.

```
$ oc get machineset preserve-dsoc12r4-ktjfc-worker-us-east-2a -n openshift-machine-api -o json > <output_file.json>
```

5. Edit the JSON file and make the following changes to the new **MachineSet** definition:

- Replace **worker** with **gpu**. This will be the name of the new machine set.
- Change the instance type of the new **MachineSet** definition to **g4dn**, which includes an NVIDIA Tesla T4 GPU. To learn more about AWS **g4dn** instance types, see [Accelerated Computing](#).

```
$ jq .spec.template.spec.providerSpec.value.instanceType preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json  
"g4dn.xlarge"
```

The **<output\_file.json>** file is saved as **preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json**.

6. Update the following fields in **preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json**:

- **.metadata.name** to a name containing **gpu**.
- **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** to match the new **.metadata.name**.
- **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** to match the new **.metadata.name**.
- **.spec.template.spec.providerSpec.value.instanceType** to **g4dn.xlarge**.

7. To verify your changes, perform a **diff** of the original compute definition and the new GPU-enabled node definition by running the following command:

```
$ oc -n openshift-machine-api get preserve-dsoc12r4-ktjfc-worker-us-east-2a -o json | diff preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json -
```

### Example output

```
10c10  
  
< "name": "preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a",  
---  
> "name": "preserve-dsoc12r4-ktjfc-worker-us-east-2a",  
  
21c21  
  
< "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-gpu-us-  
east-2a"  
---  
> "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-us-east-2a"  
  
31c31  
  
< "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-gpu-us-
```

```

east-2a"
---
> "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-us-east-2a"

60c60

< "instanceType": "g4dn.xlarge",
---
> "instanceType": "m5.xlarge",

```

8. Create the GPU-enabled compute machine set from the definition by running the following command:

```
$ oc create -f preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json
```

#### Example output

```
machineset.machine.openshift.io/preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a created
```

#### Verification

1. View the machine set you created by running the following command:

```
$ oc -n openshift-machine-api get machinesets | grep gpu
```

The MachineSet replica count is set to **1** so a new **Machine** object is created automatically.

#### Example output

```
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a 1 1 1 1 4m21s
```

2. View the **Machine** object that the machine set created by running the following command:

```
$ oc -n openshift-machine-api get machines | grep gpu
```

#### Example output

```
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a  running  g4dn.xlarge  us-east-2  us-east-2a 4m36s
```

Note that there is no need to specify a namespace for the node. The node definition is cluster scoped.

### 2.1.8. Deploying the Node Feature Discovery Operator

After the GPU-enabled node is created, you need to discover the GPU-enabled node so it can be scheduled. To do this, install the Node Feature Discovery (NFD) Operator. The NFD Operator identifies hardware device features in nodes. It solves the general problem of identifying and cataloging hardware resources in the infrastructure nodes so they can be made available to OpenShift Container Platform.

#### Procedure

1. Install the Node Feature Discovery Operator from **OperatorHub** in the OpenShift Container Platform console.
2. After installing the NFD Operator into **OperatorHub**, select **Node Feature Discovery** from the installed Operators list and select **Create instance**. This installs the **nfd-master** and **nfd-worker** pods, one **nfd-worker** pod for each compute node, in the **openshift-nfd** namespace.
3. Verify that the Operator is installed and running by running the following command:

```
$ oc get pods -n openshift-nfd
```

#### Example output

NAME	READY	STATUS	RESTARTS	AGE
nfd-controller-manager-8646fcbb65-x5qgk	2/2	Running	7 (8h ago)	1d

4. Browse to the installed Operator in the console and select **Create Node Feature Discovery**.
5. Select **Create** to build a NFD custom resource. This creates NFD pods in the **openshift-nfd** namespace that poll the OpenShift Container Platform nodes for hardware resources and catalogue them.

#### Verification

1. After a successful build, verify that a NFD pod is running on each nodes by running the following command:

```
$ oc get pods -n openshift-nfd
```

#### Example output

NAME	READY	STATUS	RESTARTS	AGE
nfd-controller-manager-8646fcbb65-x5qgk	2/2	Running	7 (8h ago)	12d
nfd-master-769656c4cb-w9vrv	1/1	Running	0	12d
nfd-worker-qjxb2	1/1	Running	3 (3d14h ago)	12d
nfd-worker-xtz9b	1/1	Running	5 (3d14h ago)	12d

The NFD Operator uses vendor PCI IDs to identify hardware in a node. NVIDIA uses the PCI ID **10de**.

2. View the NVIDIA GPU discovered by the NFD Operator by running the following command:

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

#### Example output

```
Roles: worker
feature.node.kubernetes.io/pci-1013.present=true
```

```
feature.node.kubernetes.io/pci-10de.present=true
feature.node.kubernetes.io/pci-1d0f.present=true
```

**10de** appears in the node feature list for the GPU-enabled node. This means the NFD Operator correctly identified the node from the GPU-enabled MachineSet.

## 2.2. CREATING A COMPUTE MACHINE SET ON AZURE

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



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.2.1. Sample YAML for a compute machine set custom resource on Azure

This sample YAML defines a compute machine set that runs in the **1** Microsoft Azure zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/<role>: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ①
    machine.openshift.io/cluster-api-machine-role: <role> ②
    machine.openshift.io/cluster-api-machine-type: <role>
  name: <infrastructure_id>-<role>-<region> ③
  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>-<region>
  template:
    metadata:
```

```

creationTimestamp: null
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>-<region>
spec:
  metadata:
    creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-machineset: <machineset_name>
      node-role.kubernetes.io/<role>: ""
  providerSpec:
    value:
      apiVersion: azureproviderconfig.openshift.io/v1beta1
      credentialsSecret:
        name: azure-cloud-credentials
        namespace: openshift-machine-api
      image: ④
        offer: ""
        publisher: ""
        resourceId: /resourceGroups/<infrastructure_id>-
          rg/providers/Microsoft.Compute/galleries/gallery_<infrastructure_id>/images/<infrastructure_id>-
          gen2/versions/latest ⑤
        sku: ""
        version: ""
      internalLoadBalancer: ""
      kind: AzureMachineProviderSpec
      location: <region> ⑥
      managedIdentity: <infrastructure_id>-identity
      metadata:
        creationTimestamp: null
      natRule: null
      networkResourceGroup: ""
      osDisk:
        diskSizeGB: 128
        managedDisk:
          storageAccountType: Premium_LRS
          osType: Linux
      publicIP: false
      publicLoadBalancer: ""
      resourceGroup: <infrastructure_id>-rg
      sshPrivateKey: ""
      sshPublicKey: ""
      tags:
        - name: <custom_tag_name> ⑦
          value: <custom_tag_value>
      subnet: <infrastructure_id>-<role>-subnet
      userDataSecret:
        name: worker-user-data
      vmSize: Standard_D4s_v3
      vnet: <infrastructure_id>-vnet
      zone: "1" ⑧

```

- 1 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the

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

You can obtain the subnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.subnet}' \
get machineset/<infrastructure_id>-worker-centralus1
```

You can obtain the vnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.vnet}' \
get machineset/<infrastructure_id>-worker-centralus1
```

- 2 Specify the node label to add.
- 3 Specify the infrastructure ID, node label, and region.
- 4 Specify the image details for your compute machine set. If you want to use an Azure Marketplace image, see "Selecting an Azure Marketplace image".
- 5 Specify an image that is compatible with your instance type. The Hyper-V generation V2 images created by the installation program have a **-gen2** suffix, while V1 images have the same name without the suffix.
- 6 Specify the region to place machines on.
- 7 Optional: Specify custom tags in your machine set. Provide the tag name in **<custom\_tag\_name>** field and the corresponding tag value in **<custom\_tag\_value>** field.
- 8 Specify the zone within your region to place machines on. Be sure that your region supports the zone that you specify.

## 2.2.2. Creating a compute 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 compute 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**

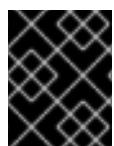
NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e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

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

### 2.2.3. Using the Azure Marketplace offering

You can create a machine set running on Azure that deploys machines that use the Azure Marketplace offering. To use this offering, you must first obtain the Azure Marketplace image. When obtaining your image, consider the following:

- While the images are the same, the Azure Marketplace publisher is different depending on your region. If you are located in North America, specify **redhat** as the publisher. If you are located in EMEA, specify **redhat-limited** as the publisher.
- The offer includes a **rh-ocp-worker** SKU and a **rh-ocp-worker-gen1** SKU. The **rh-ocp-worker** SKU represents a Hyper-V generation version 2 VM image. The default instance types used in OpenShift Container Platform are version 2 compatible. If you plan to use an instance type that is only version 1 compatible, use the image associated with the **rh-ocp-worker-gen1** SKU. The **rh-ocp-worker-gen1** SKU represents a Hyper-V version 1 VM image.

**IMPORTANT**

Installing images with the Azure marketplace is not supported on clusters with 64-bit ARM instances.

## Prerequisites

- You have installed the Azure CLI client (**az**).
- Your Azure account is entitled for the offer and you have logged into this account with the Azure CLI client.

## Procedure

1. Display all of the available OpenShift Container Platform images by running one of the following commands:

- North America:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
```

### Example output

Offer	Publisher	Sku	Urn	Version
rh-ocp-worker	RedHat	rh-ocp-worker	RedHat:rh-ocp-worker:rh-ocp-worker:4.15.2024072409	4.15.2024072409
rh-ocp-worker	RedHat	rh-ocp-worker-gen1	RedHat:rh-ocp-worker:rh-ocp-worker-gen1:4.15.2024072409	4.15.2024072409

- EMEA:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
```

### Example output

Offer	Publisher	Sku	Urn
Version			
rh-ocp-worker	redhat-limited	rh-ocp-worker	redhat-limited:rh-ocp-worker:rh-ocp-worker:4.15.2024072409
			4.15.2024072409
rh-ocp-worker	redhat-limited	rh-ocp-worker-gen1	redhat-limited:rh-ocp-worker:rh-ocp-worker-gen1:4.15.2024072409
			4.15.2024072409



### NOTE

Use the latest image that is available for compute and control plane nodes. If required, your VMs are automatically upgraded as part of the installation process.

2. Inspect the image for your offer by running one of the following commands:

- North America:

```
$ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

3. Review the terms of the offer by running one of the following commands:

- North America:

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. Accept the terms of the offering by running one of the following commands:

- North America:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. Record the image details of your offer, specifically the values for **publisher**, **offer**, **sku**, and **version**.

6. Add the following parameters to the **providerSpec** section of your machine set YAML file using the image details for your offer:

#### Sample providerSpec image values for Azure Marketplace machines

```
providerSpec:  
  value:  
    image:  
      offer: rh-ocp-worker  
      publisher: redhat  
      resourceId: ""  
      sku: rh-ocp-worker  
      type: MarketplaceWithPlan  
      version: 413.92.2023101700
```

### 2.2.4. Enabling Azure boot diagnostics

You can enable boot diagnostics on Azure machines that your machine set creates.

#### Prerequisites

- Have an existing Microsoft Azure cluster.

#### Procedure

- Add the **diagnostics** configuration that is applicable to your storage type to the **providerSpec** field in your machine set YAML file:

- For an Azure Managed storage account:

```
providerSpec:  
diagnostics:  
boot:  
storageAccountType: AzureManaged ①
```

- ① Specifies an Azure Managed storage account.

- For an Azure Unmanaged storage account:

```
providerSpec:  
diagnostics:  
boot:  
storageAccountType: CustomerManaged ①  
customerManaged:  
storageAccountURI: https://<storage-account>.blob.core.windows.net ②
```

- ① Specifies an Azure Unmanaged storage account.

- ② Replace **<storage-account>** with the name of your storage account.



#### NOTE

Only the Azure Blob Storage data service is supported.

#### Verification

- On the Microsoft Azure portal, review the **Boot diagnostics** page for a machine deployed by the machine set, and verify that you can see the serial logs for the machine.

#### 2.2.5. Machine sets that deploy machines as Spot VMs

You can save on costs by creating a compute machine set running on Azure that deploys machines as non-guaranteed Spot VMs. Spot VMs utilize unused Azure capacity and are less expensive than standard VMs. You can use Spot VMs for workloads that can tolerate interruptions, such as batch or stateless, horizontally scalable workloads.

Azure can terminate a Spot VM at any time. Azure gives a 30-second warning to the user when an interruption occurs. OpenShift Container Platform begins to remove the workloads from the affected instances when Azure issues the termination warning.

Interruptions can occur when using Spot VMs for the following reasons:

- The instance price exceeds your maximum price
- The supply of Spot VMs decreases
- Azure needs capacity back

When Azure terminates an instance, a termination handler running on the Spot VM node deletes the machine resource. To satisfy the compute machine set **replicas** quantity, the compute machine set creates a machine that requests a Spot VM.

### 2.2.5.1. Creating Spot VMs by using compute machine sets

You can launch a Spot VM on Azure by adding **spotVMOPTIONS** to your compute machine set YAML file.

#### Procedure

- Add the following line under the **providerSpec** field:

```
providerSpec:  
  value:  
    spotVMOPTIONS: {}
```

You can optionally set the **spotVMOPTIONS.maxPrice** field to limit the cost of the Spot VM. For example you can set **maxPrice: '0.98765'**. If the **maxPrice** is set, this value is used as the hourly maximum spot price. If it is not set, the maximum price defaults to **-1** and charges up to the standard VM price.

Azure caps Spot VM prices at the standard price. Azure will not evict an instance due to pricing if the instance is set with the default **maxPrice**. However, an instance can still be evicted due to capacity restrictions.



#### NOTE

It is strongly recommended to use the default standard VM price as the **maxPrice** value and to not set the maximum price for Spot VMs.

### 2.2.6. Machine sets that deploy machines on Ephemeral OS disks

You can create a compute machine set running on Azure that deploys machines on Ephemeral OS disks. Ephemeral OS disks use local VM capacity rather than remote Azure Storage. This configuration therefore incurs no additional cost and provides lower latency for reading, writing, and reimaging.

#### Additional resources

- For more information, see the Microsoft Azure documentation about [Ephemeral OS disks for Azure VMs](#).

### 2.2.6.1. Creating machines on Ephemeral OS disks by using compute machine sets

You can launch machines on Ephemeral OS disks on Azure by editing your compute machine set YAML file.

#### Prerequisites

- Have an existing Microsoft Azure cluster.

#### Procedure

1. Edit the custom resource (CR) by running the following command:

```
$ oc edit machineset <machine-set-name>
```

where **<machine-set-name>** is the compute machine set that you want to provision machines on Ephemeral OS disks.

2. Add the following to the **providerSpec** field:

```
providerSpec:
  value:
    ...
    osDisk:
      ...
      diskSettings: ①
        ephemeralStorageLocation: Local ②
        cachingType: ReadOnly ③
      managedDisk:
        storageAccountType: Standard_LRS ④
      ...
    
```

① ② ③ These lines enable the use of Ephemeral OS disks.

④ Ephemeral OS disks are only supported for VMs or scale set instances that use the Standard LRS storage account type.



### IMPORTANT

The implementation of Ephemeral OS disk support in OpenShift Container Platform only supports the **CacheDisk** placement type. Do not change the **placement** configuration setting.

3. Create a compute machine set using the updated configuration:

```
$ oc create -f <machine-set-config>.yaml
```

### Verification

- On the Microsoft Azure portal, review the **Overview** page for a machine deployed by the compute machine set, and verify that the **Ephemeral OS disk** field is set to **OS cache placement**.

### 2.2.7. Machine sets that deploy machines with ultra disks as data disks

You can create a machine set running on Azure that deploys machines with ultra disks. Ultra disks are high-performance storage that are intended for use with the most demanding data workloads.

You can also create a persistent volume claim (PVC) that dynamically binds to a storage class backed by Azure ultra disks and mounts them to pods.



### NOTE

Data disks do not support the ability to specify disk throughput or disk IOPS. You can configure these properties by using PVCs.

### Additional resources

- [Microsoft Azure ultra disks documentation](#)

- Machine sets that deploy machines on ultra disks using CSI PVCs
- Machine sets that deploy machines on ultra disks using in-tree PVCs

### 2.2.7.1. Creating machines with ultra disks by using machine sets

You can deploy machines with ultra disks on Azure by editing your machine set YAML file.

#### Prerequisites

- Have an existing Microsoft Azure cluster.

#### Procedure

1. Create a custom secret in the **openshift-machine-api** namespace using the **worker** data secret by running the following command:

```
$ oc -n openshift-machine-api \
get secret <role>-user-data \ ①
--template='{{index .data.userData | base64decode}}' | jq > userData.txt ②
```

- 1 Replace **<role>** with **worker**.
- 2 Specify **userData.txt** as the name of the new custom secret.

2. In a text editor, open the **userData.txt** file and locate the final **}** character in the file.

- a. On the immediately preceding line, add a **,**
- b. Create a new line after the **,** and add the following configuration details:

```
"storage": {
  "disks": [ ①
    {
      "device": "/dev/disk/azure/scsi1/lun0", ②
      "partitions": [ ③
        {
          "label": "lun0p1", ④
          "sizeMiB": 1024, ⑤
          "startMiB": 0
        }
      ]
    }
  ],
  "filesystems": [ ⑥
    {
      "device": "/dev/disk/by-partlabel/lun0p1",
      "format": "xfs",
      "path": "/var/lib/lun0p1"
    }
  ],
  "systemd": {
```

```

"units": [ 7
  {
    "contents": "[Unit]\nBefore=local-
fs.target\n[Mount]\nWhere=/var/lib/lun0p1\nWhat=/dev/disk/by-
partlabel/lun0p1\nOptions=defaults,pquota\n[Install]\nWantedBy=local-fs.target\n", 8
    "enabled": true,
    "name": "var-lib-lun0p1.mount"
  }
]
}

```

- 1 The configuration details for the disk that you want to attach to a node as an ultra disk.
- 2 Specify the **lun** value that is defined in the **dataDisks** stanza of the machine set you are using. For example, if the machine set contains **lun: 0**, specify **lun0**. You can initialize multiple data disks by specifying multiple "**disks**" entries in this configuration file. If you specify multiple "**disks**" entries, ensure that the **lun** value for each matches the value in the machine set.
- 3 The configuration details for a new partition on the disk.
- 4 Specify a label for the partition. You might find it helpful to use hierarchical names, such as **lun0p1** for the first partition of **lun0**.
- 5 Specify the total size in MiB of the partition.
- 6 Specify the filesystem to use when formatting a partition. Use the partition label to specify the partition.
- 7 Specify a **systemd** unit to mount the partition at boot. Use the partition label to specify the partition. You can create multiple partitions by specifying multiple "**partitions**" entries in this configuration file. If you specify multiple "**partitions**" entries, you must specify a **systemd** unit for each.
- 8 For **Where**, specify the value of **storage.filesystems.path**. For **What**, specify the value of **storage.filesystems.device**.

- 3 Extract the disabling template value to a file called **disableTemplating.txt** by running the following command:

```
$ oc -n openshift-machine-api get secret <role>-user-data \ 1
--template='{{index .data.disableTemplating | base64decode}}' | jq > disableTemplating.txt
```

- 1 Replace **<role>** with **worker**.

- 4 Combine the **userData.txt** file and **disableTemplating.txt** file to create a data secret file by running the following command:

```
$ oc -n openshift-machine-api create secret generic <role>-user-data-x5 \ 1
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt
```

- 1 For **<role>-user-data-x5**, specify the name of the secret. Replace **<role>** with **worker**.

5. Copy an existing Azure **MachineSet** custom resource (CR) and edit it by running the following command:

```
$ oc edit machineset <machine-set-name>
```

where **<machine-set-name>** is the machine set that you want to provision machines with ultra disks.

6. Add the following lines in the positions indicated:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd 1
      providerSpec:
        value:
          ultraSSDCapability: Enabled 2
          dataDisks: 3
            - nameSuffix: ultrassd
              lun: 0
              diskSizeGB: 4
            deletionPolicy: Delete
            cachingType: None
            managedDisk:
              storageAccountType: UltraSSD_LRS
          userDataSecret:
            name: <role>-user-data-x5 4
```

**1** Specify a label to use to select a node that is created by this machine set. This procedure uses **disk.ultrassd** for this value.

**2** **3** These lines enable the use of ultra disks. For **dataDisks**, include the entire stanza.

**4** Specify the user data secret created earlier. Replace **<role>** with **worker**.

7. Create a machine set using the updated configuration by running the following command:

```
$ oc create -f <machine-set-name>.yaml
```

## Verification

1. Validate that the machines are created by running the following command:

```
$ oc get machines
```

The machines should be in the **Running** state.

2. For a machine that is running and has a node attached, validate the partition by running the following command:

```
$ oc debug node/<node-name> -- chroot /host lsblk
```

In this command, **oc debug node/<node-name>** starts a debugging shell on the node **<node-name>** and passes a command with **--**. The passed command **chroot /host** provides access to the underlying host OS binaries, and **lsblk** shows the block devices that are attached to the host OS machine.

## Next steps

- To use an ultra disk from within a pod, create a workload that uses the mount point. Create a YAML file similar to the following example:

```
apiVersion: v1
kind: Pod
metadata:
  name: ssd-benchmark1
spec:
  containers:
    - name: ssd-benchmark1
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - name: lun0p1
          mountPath: "/tmp"
  volumes:
    - name: lun0p1
      hostPath:
        path: /var/lib/lun0p1
        type: DirectoryOrCreate
  nodeSelector:
    disktype: ultrassd
```

### 2.2.7.2. Troubleshooting resources for machine sets that enable ultra disks

Use the information in this section to understand and recover from issues you might encounter.

#### 2.2.7.2.1. Incorrect ultra disk configuration

If an incorrect configuration of the **ultraSSDCapability** parameter is specified in the machine set, the machine provisioning fails.

For example, if the **ultraSSDCapability** parameter is set to **Disabled**, but an ultra disk is specified in the **dataDisks** parameter, the following error message appears:

StorageAccountType UltraSSD\_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.

- To resolve this issue, verify that your machine set configuration is correct.

#### 2.2.7.2.2. Unsupported disk parameters

If a region, availability zone, or instance size that is not compatible with ultra disks is specified in the machine set, the machine provisioning fails. Check the logs for the following error message:

```
failed to create vm <machine_name>; failure sending request for machine <machine_name>; cannot
create vm: compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request:
StatusCode=400 -- Original Error: Code="BadRequest" Message="Storage Account type
'UltraSSD_LRS' is not supported <more_information_about_why>."
```

- To resolve this issue, verify that you are using this feature in a supported environment and that your machine set configuration is correct.

#### 2.2.7.2.3. Unable to delete disks

If the deletion of ultra disks as data disks is not working as expected, the machines are deleted and the data disks are orphaned. You must delete the orphaned disks manually if desired.

### 2.2.8. Enabling customer-managed encryption keys for a machine set

You can supply an encryption key to Azure to encrypt data on managed disks at rest. You can enable server-side encryption with customer-managed keys by using the Machine API.

An Azure Key Vault, a disk encryption set, and an encryption key are required to use a customer-managed key. The disk encryption set must be in a resource group where the Cloud Credential Operator (CCO) has granted permissions. If not, an additional reader role is required to be granted on the disk encryption set.

#### Prerequisites

- [Create an Azure Key Vault instance](#).
- [Create an instance of a disk encryption set](#).
- [Grant the disk encryption set access to key vault](#).

#### Procedure

- Configure the disk encryption set under the **providerSpec** field in your machine set YAML file. For example:

```
providerSpec:
  value:
    osDisk:
      diskSizeGB: 128
    managedDisk:
      diskEncryptionSet:
        id:
          /subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.
          Compute/diskEncryptionSets/<disk_encryption_set_name>
        storageAccountType: Premium_LRS
```

#### Additional resources

- [Azure documentation about customer-managed keys](#)

## 2.2.9. Configuring trusted launch for Azure virtual machines by using machine sets



### IMPORTANT

Using trusted launch for Azure virtual machines is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

OpenShift Container Platform 4.17 supports trusted launch for Azure virtual machines (VMs). By editing the machine set YAML file, you can configure the trusted launch options that a machine set uses for machines that it deploys. For example, you can configure these machines to use UEFI security features such as Secure Boot or a dedicated virtual Trusted Platform Module (vTPM) instance.



### NOTE

Some feature combinations result in an invalid configuration.

**Table 2.1. UEFI feature combination compatibility**

Secure Boot[1]	vTPM[2]	Valid configuration
Enabled	Enabled	Yes
Enabled	Disabled	Yes
Enabled	Omitted	Yes
Disabled	Enabled	Yes
Omitted	Enabled	Yes
Disabled	Disabled	No
Omitted	Disabled	No
Omitted	Omitted	No

1. Using the **secureBoot** field.
2. Using the **virtualizedTrustedPlatformModule** field.

For more information about related features and functionality, see the Microsoft Azure documentation about [Trusted launch for Azure virtual machines](#).

## Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following section under the **providerSpec** field to provide a valid configuration:

### Sample valid configuration with UEFI Secure Boot and vTPM enabled

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    machines_v1beta1_machine_openshift_io:
      spec:
        providerSpec:
          value:
            securityProfile:
              settings:
                securityType: TrustedLaunch 1
                trustedLaunch:
                  uefiSettings: 2
                    secureBoot: Enabled 3
                    virtualizedTrustedPlatformModule: Enabled 4
# ...
```

- 1** Enables the use of trusted launch for Azure virtual machines. This value is required for all valid configurations.
- 2** Specifies which UEFI security features to use. This section is required for all valid configurations.
- 3** Enables UEFI Secure Boot.
- 4** Enables the use of a vTPM.

## Verification

- On the Azure portal, review the details for a machine deployed by the machine set and verify that the trusted launch options match the values that you configured.

### 2.2.10. Configuring Azure confidential virtual machines by using machine sets



## IMPORTANT

Using Azure confidential virtual machines is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

OpenShift Container Platform 4.17 supports Azure confidential virtual machines (VMs).



## NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

By editing the machine set YAML file, you can configure the confidential VM options that a machine set uses for machines that it deploys. For example, you can configure these machines to use UEFI security features such as Secure Boot or a dedicated virtual Trusted Platform Module (vTPM) instance.

For more information about related features and functionality, see the Microsoft Azure documentation about [Confidential virtual machines](#).

## Procedure

- 1 In a text editor, open the YAML file for an existing machine set or create a new one.
- 2 Edit the following section under the **providerSpec** field:

### Sample configuration

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          osDisk:
            # ...
            managedDisk:
              securityProfile: ①
                securityEncryptionType: VMGuestStateOnly ②
              # ...
              securityProfile: ③
                settings:
                  securityType: ConfidentialVM ④
                  confidentialVM:
                    uefiSettings: ⑤
                    secureBoot: Disabled ⑥
```

```

virtualizedTrustedPlatformModule: Enabled 7
vmSize: Standard_DC16ads_v5 8
# ...

```

- 1 Specifies security profile settings for the managed disk when using a confidential VM.
- 2 Enables encryption of the Azure VM Guest State (VMGS) blob. This setting requires the use of vTPM.
- 3 Specifies security profile settings for the confidential VM.
- 4 Enables the use of confidential VMs. This value is required for all valid configurations.
- 5 Specifies which UEFI security features to use. This section is required for all valid configurations.
- 6 Disables UEFI Secure Boot.
- 7 Enables the use of a vTPM.
- 8 Specifies an instance type that supports confidential VMs.

## Verification

- On the Azure portal, review the details for a machine deployed by the machine set and verify that the confidential VM options match the values that you configured.

### 2.2.11. Accelerated Networking for Microsoft Azure VMs

Accelerated Networking uses single root I/O virtualization (SR-IOV) to provide Microsoft Azure VMs with a more direct path to the switch. This enhances network performance. This feature can be enabled during or after installation.

#### 2.2.11.1. Limitations

Consider the following limitations when deciding whether to use Accelerated Networking:

- Accelerated Networking is only supported on clusters where the Machine API is operational.
- Although the minimum requirement for an Azure worker node is two vCPUs, Accelerated Networking requires an Azure VM size that includes at least four vCPUs. To satisfy this requirement, you can change the value of **vmSize** in your machine set. For information about Azure VM sizes, see [Microsoft Azure documentation](#).
- When this feature is enabled on an existing Azure cluster, only newly provisioned nodes are affected. Currently running nodes are not reconciled. To enable the feature on all nodes, you must replace each existing machine. This can be done for each machine individually, or by scaling the replicas down to zero, and then scaling back up to your desired number of replicas.

### 2.2.12. Configuring Capacity Reservation by using machine sets

OpenShift Container Platform version 4.17 and later supports on-demand Capacity Reservation with Capacity Reservation groups on Microsoft Azure clusters.

You can configure a machine set to deploy machines on any available resources that match the parameters of a capacity request that you define. These parameters specify the VM size, region, and number of instances that you want to reserve. If your Azure subscription quota can accommodate the capacity request, the deployment succeeds.

For more information, including limitations and suggested use cases for this Azure instance type, see the Microsoft Azure documentation about [On-demand Capacity Reservation](#).



## NOTE

You cannot change an existing Capacity Reservation configuration for a machine set. To use a different Capacity Reservation group, you must replace the machine set and the machines that the previous machine set deployed.

## Prerequisites

- You have access to the cluster with **cluster-admin** privileges.
- You installed the OpenShift CLI (**oc**).
- You created a Capacity Reservation group.  
For more information, see the Microsoft Azure documentation [Create a Capacity Reservation](#).

## Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following section under the **providerSpec** field:

### Sample configuration

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          capacityReservationGroupID: <capacity_reservation_group> ①
# ...
```

- 1 Specify the ID of the Capacity Reservation group that you want the machine set to deploy machines on.

## Verification

- To verify machine deployment, list the machines that the machine set created by running the following command:

```
$ oc get machines.machine.openshift.io \
-n openshift-machine-api \
-l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

where <machine\_set\_name> is the name of the compute machine set.

In the output, verify that the characteristics of the listed machines match the parameters of your Capacity Reservation.

### 2.2.13. Adding a GPU node to an existing OpenShift Container Platform cluster

You can copy and modify a default compute machine set configuration to create a GPU-enabled machine set and machines for the Azure cloud provider.

The following table lists the validated instance types:

vmSize	NVIDIA GPU accelerator	Maximum number of GPUs	Architecture
<b>Standard_NC24s_v3</b>	V100	4	x86
<b>Standard_NC4as_T4_v3</b>	T4	1	x86
<b>ND A100 v4</b>	A100	8	x86



#### NOTE

By default, Azure subscriptions do not have a quota for the Azure instance types with GPU. Customers have to request a quota increase for the Azure instance families listed above.

#### Procedure

- View the machines and machine sets that exist in the **openshift-machine-api** namespace by running the following command. Each compute machine set is associated with a different availability zone within the Azure region. The installer automatically load balances compute machines across availability zones.

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

#### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
myclustername-worker-centralus1	1	1	1	1	6h9m
myclustername-worker-centralus2	1	1	1	1	6h9m
myclustername-worker-centralus3	1	1	1	1	6h9m

- Make a copy of one of the existing compute **MachineSet** definitions and output the result to a YAML file by running the following command. This will be the basis for the GPU-enabled compute machine set definition.

```
$ oc get machineset -n openshift-machine-api myclustername-worker-centralus1 -o yaml > machineset-azure.yaml
```

- View the content of the machineset:

```
$ cat machineset-azure.yaml
```

### Example machineset-azure.yaml file

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  annotations:
    machine.openshift.io/GPU: "0"
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
  creationTimestamp: "2023-02-06T14:08:19Z"
  generation: 1
  labels:
    machine.openshift.io/cluster-api-cluster: myclusternamespace
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: myclusternamespace-worker-centralus1
  namespace: openshift-machine-api
  resourceVersion: "23601"
  uid: acd56e0c-7612-473a-ae37-8704f34b80de
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: myclusternamespace
      machine.openshift.io/cluster-api-machineset: myclusternamespace-worker-centralus1
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: myclusternamespace
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: myclusternamespace-worker-centralus1
    spec:
      lifecycleHooks: {}
      metadata: {}
      providerSpec:
        value:
          acceleratedNetworking: true
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          diagnostics: {}
          image:
            offer: ""
            publisher: ""
            resourceId: /resourceGroups/myclusternamespace-rg/providers/Microsoft.Compute/galleries/gallery_myclusternamespace_n6n4r/images/myclusternamespace-gen2/versions/latest
            sku: ""
            version: ""

```

```

kind: AzureMachineProviderSpec
location: centralus
managedIdentity: myclustername-identity
metadata:
  creationTimestamp: null
networkResourceGroup: myclustername-rg
osDisk:
  diskSettings: {}
  diskSizeGB: 128
  managedDisk:
    storageAccountType: Premium_LRS
  osType: Linux
  publicIP: false
publicLoadBalancer: myclustername
resourceGroup: myclustername-rg
spotVMOptions: {}
subnet: myclustername-worker-subnet
userDataSecret:
  name: worker-user-data
vmSize: Standard_D4s_v3
vnet: myclustername-vnet
zone: "1"
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1

```

4. Make a copy of the **machineset-azure.yaml** file by running the following command:

```
$ cp machineset-azure.yaml machineset-azure-gpu.yaml
```

5. Update the following fields in **machineset-azure-gpu.yaml**:

- Change **.metadata.name** to a name containing **gpu**.
- Change **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** to match the new **.metadata.name**.
- Change **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** to match the new **.metadata.name**.
- Change **.spec.template.spec.providerSpec.value.vmSize** to **Standard\_NC4as\_T4\_v3**.

#### Example machineset-azure-gpu.yaml file

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  annotations:
    machine.openshift.io/GPU: "1"
    machine.openshift.io/memoryMb: "28672"
    machine.openshift.io/vCPU: "4"
  creationTimestamp: "2023-02-06T20:27:12Z"
  generation: 1

```

```

labels:
  machine.openshift.io/cluster-api-cluster: myclusternamespace
  machine.openshift.io/cluster-api-machine-role: worker
  machine.openshift.io/cluster-api-machine-type: worker
  name: myclusternamespace-nc4ast4-gpu-worker-centralus1
  namespace: openshift-machine-api
  resourceVersion: "166285"
  uid: 4eedce7f-6a57-4abe-b529-031140f02ffa
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: myclusternamespace
      machine.openshift.io/cluster-api-machineset: myclusternamespace-nc4ast4-gpu-worker-centralus1
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: myclusternamespace
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: myclusternamespace-nc4ast4-gpu-worker-centralus1
    spec:
      lifecycleHooks: {}
      metadata: {}
      providerSpec:
        value:
          acceleratedNetworking: true
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          diagnostics: {}
          image:
            offer: ""
            publisher: ""
            resourceId: /resourceGroups/myclusternamespace-rg/providers/Microsoft.Compute/galleries/gallery_myclusternamespace_n6n4r/images/myclusternamespace-gen2/versions/latest
            sku: ""
            version: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: myclusternamespace-identity
          metadata:
            creationTimestamp: null
          networkResourceGroup: myclusternamespace-rg
          osDisk:
            diskSettings: {}
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
              osType: Linux
            publicIP: false
            publicLoadBalancer: myclusternamespace

```

```

resourceGroup: myclustername-rg
spotVMOptions: {}
subnet: myclustername-worker-subnet
userDataSecret:
  name: worker-user-data
vmSize: Standard_NC4as_T4_v3
vnet: myclustername-vnet
zone: "1"
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1

```

- To verify your changes, perform a **diff** of the original compute definition and the new GPU-enabled node definition by running the following command:

```
$ diff machineset-azure.yaml machineset-azure-gpu.yaml
```

#### Example output

```

14c14
<   name: myclustername-worker-centralus1
---
>   name: myclustername-nc4ast4-gpu-worker-centralus1
23c23
<     machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
---
>     machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
30c30
<       machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
---
>       machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
67c67
<         vmSize: Standard_D4s_v3
---
>         vmSize: Standard_NC4as_T4_v3

```

- Create the GPU-enabled compute machine set from the definition file by running the following command:

```
$ oc create -f machineset-azure-gpu.yaml
```

#### Example output

```
machineset.machine.openshift.io/myclustername-nc4ast4-gpu-worker-centralus1 created
```

- View the machines and machine sets that exist in the **openshift-machine-api** namespace by running the following command. Each compute machine set is associated with a different availability zone within the Azure region. The installer automatically load balances compute machines across availability zones.

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

### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
clustername-n6n4r-nc4ast4-gpu-worker-centralus1	1	1	1	1	122m
clustername-n6n4r-worker-centralus1	1	1	1	1	8h
clustername-n6n4r-worker-centralus2	1	1	1	1	8h
clustername-n6n4r-worker-centralus3	1	1	1	1	8h

9. View the machines that exist in the **openshift-machine-api** namespace by running the following command. You can only configure one compute machine per set, although you can scale a compute machine set to add a node in a particular region and zone.

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

### Example output

NAME	PHASE	TYPE	REGION	ZONE	AGE
myclustername-master-0	Running	Standard_D8s_v3	centralus	2	6h40m
myclustername-master-1	Running	Standard_D8s_v3	centralus	1	6h40m
myclustername-master-2	Running	Standard_D8s_v3	centralus	3	6h40m
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn	Running	centralus	1	21m	myclustername-worker-centralus1-rbh6b
1	Running	Standard_D4s_v3	centralus	6h38m	myclustername-worker-centralus2-dbz7w
2	Running	Standard_D4s_v3	centralus	6h38m	myclustername-worker-centralus3-p9b8c
3	Running	Standard_D4s_v3	centralus	6h38m	

10. View the existing nodes, machines, and machine sets by running the following command. Note that each node is an instance of a machine definition with a specific Azure region and OpenShift Container Platform role.

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
myclustername-master-0	Ready	control-plane,master	6h39m	v1.30.3
myclustername-master-1	Ready	control-plane,master	6h41m	v1.30.3
myclustername-master-2	Ready	control-plane,master	6h39m	v1.30.3
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn	Ready	worker	14m	v1.30.3
myclustername-worker-centralus1-rbh6b	Ready	worker	6h29m	v1.30.3
myclustername-worker-centralus2-dbz7w	Ready	worker	6h29m	v1.30.3
myclustername-worker-centralus3-p9b8c	Ready	worker	6h31m	v1.30.3

11. View the list of compute machine sets:

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

### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
myclustername-worker-centralus1	1	1	1	1	8h
myclustername-worker-centralus2	1	1	1	1	8h
myclustername-worker-centralus3	1	1	1	1	8h

12. Create the GPU-enabled compute machine set from the definition file by running the following command:

```
$ oc create -f machineset-azure-gpu.yaml
```

13. View the list of compute machine sets:

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

### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
myclustername-nc4ast4-gpu-worker-centralus1	1	1	1	1	121m
myclustername-worker-centralus1	1	1	1	1	8h
myclustername-worker-centralus2	1	1	1	1	8h
myclustername-worker-centralus3	1	1	1	1	8h

## Verification

1. View the machine set you created by running the following command:

```
$ oc get machineset -n openshift-machine-api | grep gpu
```

The MachineSet replica count is set to **1** so a new **Machine** object is created automatically.

### Example output

```
myclustername-nc4ast4-gpu-worker-centralus1 1 1 1 1 121m
```

2. View the **Machine** object that the machine set created by running the following command:

```
$ oc -n openshift-machine-api get machines | grep gpu
```

### Example output

```
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn Running Standard_NC4as_T4_v3
centralus 1 21m
```



### NOTE

There is no need to specify a namespace for the node. The node definition is cluster scoped.

## 2.2.14. Deploying the Node Feature Discovery Operator

After the GPU-enabled node is created, you need to discover the GPU-enabled node so it can be scheduled. To do this, install the Node Feature Discovery (NFD) Operator. The NFD Operator identifies hardware device features in nodes. It solves the general problem of identifying and cataloging hardware resources in the infrastructure nodes so they can be made available to OpenShift Container Platform.

### Procedure

1. Install the Node Feature Discovery Operator from **OperatorHub** in the OpenShift Container Platform console.
2. After installing the NFD Operator into **OperatorHub**, select **Node Feature Discovery** from the installed Operators list and select **Create instance**. This installs the **nfd-master** and **nfd-worker** pods, one **nfd-worker** pod for each compute node, in the **openshift-nfd** namespace.
3. Verify that the Operator is installed and running by running the following command:

```
$ oc get pods -n openshift-nfd
```

### Example output

NAME	READY	STATUS	RESTARTS	AGE
nfd-controller-manager-8646fcbb65-x5qgk	2/2	Running	7 (8h ago)	1d

4. Browse to the installed Operator in the console and select **Create Node Feature Discovery**.
5. Select **Create** to build a NFD custom resource. This creates NFD pods in the **openshift-nfd** namespace that poll the OpenShift Container Platform nodes for hardware resources and catalogue them.

### Verification

1. After a successful build, verify that a NFD pod is running on each nodes by running the following command:

```
$ oc get pods -n openshift-nfd
```

### Example output

NAME	READY	STATUS	RESTARTS	AGE
nfd-controller-manager-8646fcbb65-x5qgk	2/2	Running	7 (8h ago)	12d
nfd-master-769656c4cb-w9vrv	1/1	Running	0	12d
nfd-worker-qjxb2	1/1	Running	3 (3d14h ago)	12d
nfd-worker-xtz9b	1/1	Running	5 (3d14h ago)	12d

The NFD Operator uses vendor PCI IDs to identify hardware in a node. NVIDIA uses the PCI ID **10de**.

2. View the NVIDIA GPU discovered by the NFD Operator by running the following command:

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

## Example output

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

**10de** appears in the node feature list for the GPU-enabled node. This mean the NFD Operator correctly identified the node from the GPU-enabled MachineSet.

## Additional resources

- [Enabling Accelerated Networking during installation](#)

### 2.2.14.1. Enabling Accelerated Networking on an existing Microsoft Azure cluster

You can enable Accelerated Networking on Azure by adding **acceleratedNetworking** to your machine set YAML file.

## Prerequisites

- Have an existing Microsoft Azure cluster where the Machine API is operational.

## Procedure

- Add the following to the **providerSpec** field:

```
providerSpec:
  value:
    acceleratedNetworking: true 1
    vmSize: <azure-vm-size> 2
```

- 1** This line enables Accelerated Networking.
- 2** Specify an Azure VM size that includes at least four vCPUs. For information about VM sizes, see [Microsoft Azure documentation](#).

## Next steps

- To enable the feature on currently running nodes, you must replace each existing machine. This can be done for each machine individually, or by scaling the replicas down to zero, and then scaling back up to your desired number of replicas.

## Verification

- On the Microsoft Azure portal, review the **Networking** settings page for a machine provisioned by the machine set, and verify that the **Accelerated networking** field is set to **Enabled**.

## Additional resources

- Manually scaling a compute machine set

## 2.3. CREATING A COMPUTE MACHINE SET ON AZURE STACK HUB

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



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.3.1. Sample YAML for a compute machine set custom resource on Azure Stack Hub

This sample YAML defines a compute machine set that runs in the **1** Microsoft Azure zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/<role>**: "".

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ①
    machine.openshift.io/cluster-api-machine-role: <role> ②
    machine.openshift.io/cluster-api-machine-type: <role> ③
  name: <infrastructure_id>-<role>-<region> ④
  namespace: openshift-machine-api
spec:
  replicas: ⑤
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ⑥
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> ⑦
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ⑧
        machine.openshift.io/cluster-api-machine-role: <role> ⑨
        machine.openshift.io/cluster-api-machine-type: <role>
```

```

machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
spec:
  metadata:
    creationTimestamp: null
  labels:
    node-role.kubernetes.io/<role>: "" 11
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1beta1
      availabilitySet: <availability_set> 12
      credentialsSecret:
        name: azure-cloud-credentials
        namespace: openshift-machine-api
      image:
        offer: ""
        publisher: ""
        resourceId: /resourceGroups/<infrastructure_id>-
          rg/providers/Microsoft.Compute/images/<infrastructure_id> 13
        sku: ""
        version: ""
      internalLoadBalancer: ""
      kind: AzureMachineProviderSpec
      location: <region> 14
      managedIdentity: <infrastructure_id>-identity 15
      metadata:
        creationTimestamp: null
      natRule: null
      networkResourceGroup: ""
      osDisk:
        diskSizeGB: 128
        managedDisk:
          storageAccountType: Premium_LRS
          osType: Linux
      publicIP: false
      publicLoadBalancer: ""
      resourceGroup: <infrastructure_id>-rg 16
      sshPrivateKey: ""
      sshPublicKey: ""
      subnet: <infrastructure_id>-<role>-subnet 17 18
      userDataSecret:
        name: worker-user-data 19
      vmSize: Standard_DS4_v2
      vnet: <infrastructure_id>-vnet 20
      zone: "1" 21

```

**1 5 7 13 15 16 17 20** Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

You can obtain the subnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.subnet}{"\n"}' \
get machineset/<infrastructure_id>-worker-centralus1
```

You can obtain the vnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.vnet}{"\n"}' \
get machineset/<infrastructure_id>-worker-centralus1
```

**2 3 8 9 11 18 19** Specify the node label to add.

**4 6 10** Specify the infrastructure ID, node label, and region.

**14** Specify the region to place machines on.

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

**12** Specify the availability set for the cluster.

### 2.3.2. Creating a compute 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.
- Create an availability set in which to deploy Azure Stack Hub compute machines.

#### Procedure

1. Create a new YAML file that contains the compute machine set custom resource (CR) sample and is named **<file\_name>.yaml**.  
Ensure that you set the **<availabilitySet>**, **<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-vplvk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplvk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplvk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplvk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplvk-worker-us-east-1d	0	0			55m
agl030519-vplvk-worker-us-east-1e	0	0			55m
agl030519-vplvk-worker-us-east-1f	0	0			55m

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

### 2.3.3. Enabling Azure boot diagnostics

You can enable boot diagnostics on Azure machines that your machine set creates.

#### Prerequisites

- Have an existing Microsoft Azure Stack Hub cluster.

#### Procedure

- Add the **diagnostics** configuration that is applicable to your storage type to the **providerSpec** field in your machine set YAML file:
  - For an Azure Managed storage account:

```
providerSpec:  
  diagnostics:  
    boot:  
      storageAccountType: AzureManaged ①
```

① Specifies an Azure Managed storage account.

- For an Azure Unmanaged storage account:

```
providerSpec:  
  diagnostics:  
    boot:  
      storageAccountType: CustomerManaged ①  
      customerManaged:  
        storageAccountURI: https://<storage-account>.blob.core.windows.net ②
```

- 1 Specifies an Azure Unmanaged storage account.
- 2 Replace <storage-account> with the name of your storage account.

**NOTE**

Only the Azure Blob Storage data service is supported.

**Verification**

- On the Microsoft Azure portal, review the **Boot diagnostics** page for a machine deployed by the machine set, and verify that you can see the serial logs for the machine.

**2.3.4. Enabling customer-managed encryption keys for a machine set**

You can supply an encryption key to Azure to encrypt data on managed disks at rest. You can enable server-side encryption with customer-managed keys by using the Machine API.

An Azure Key Vault, a disk encryption set, and an encryption key are required to use a customer-managed key. The disk encryption set must be in a resource group where the Cloud Credential Operator (CCO) has granted permissions. If not, an additional reader role is required to be granted on the disk encryption set.

**Prerequisites**

- [Create an Azure Key Vault instance](#).
- [Create an instance of a disk encryption set](#).
- [Grant the disk encryption set access to key vault](#).

**Procedure**

- Configure the disk encryption set under the **providerSpec** field in your machine set YAML file. For example:

```
providerSpec:
  value:
    osDisk:
      diskSizeGB: 128
      managedDisk:
        diskEncryptionSet:
          id:
            /subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.Compute/diskEncryptionSets/<disk_encryption_set_name>
          storageAccountType: Premium_LRS
```

**Additional resources**

- [Azure documentation about customer-managed keys](#)

**2.4. CREATING A COMPUTE MACHINE SET ON GCP**

You can create a different compute machine set to serve a specific purpose in your OpenShift Container Platform cluster on Google Cloud Platform (GCP). For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.



## IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.4.1. Sample YAML for a compute machine set custom resource on GCP

This sample YAML defines a compute machine set that runs in Google Cloud Platform (GCP) and creates nodes that are labeled with **node-role.kubernetes.io/<role>: ""**, where **<role>** is the node label to add.

#### Values obtained by using the OpenShift CLI

In the following example, you can obtain some of the values for your cluster by using the OpenShift CLI.

#### Infrastructure ID

The **<infrastructure\_id>** string is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

#### Image path

The **<path\_to\_image>** string is the path to the image that was used to create the disk. If you have the OpenShift CLI installed, you can obtain the path to the image by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}{"\n"}' \
get machineset/<infrastructure_id>-worker-a
```

#### Sample GCP MachineSet values

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ①
  name: <infrastructure_id>-w-a
```

```

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>-w-a
  template:
    metadata:
      creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <role> 2
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: ""
    providerSpec:
      value:
        apiVersion: gcpprovider.openshift.io/v1beta1
        canIPForward: false
        credentialsSecret:
          name: gcp-cloud-credentials
        deletionProtection: false
        disks:
          - autoDelete: true
            boot: true
            image: <path_to_image> 3
            labels: null
            sizeGb: 128
            type: pd-ssd
        gcpMetadata: 4
          - key: <custom_metadata_key>
            value: <custom_metadata_value>
        kind: GCPMachineProviderSpec
        machineType: n1-standard-4
        metadata:
          creationTimestamp: null
        networkInterfaces:
          - network: <infrastructure_id>-network
            subnetwork: <infrastructure_id>-worker-subnet
        projectID: <project_name> 5
        region: us-central1
        serviceAccounts: 6
          - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com
            scopes:
              - https://www.googleapis.com/auth/cloud-platform
        tags:
          - <infrastructure_id>-worker
        userDataSecret:
          name: worker-user-data
        zone: us-central1-a

```

- 1 For <**infrastructure\_id**>, specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.
- 2 For <**node**>, specify the node label to add.
- 3 Specify the path to the image that is used in current compute machine sets.

To use a GCP Marketplace image, specify the offer to use:

- OpenShift Container Platform:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736>
- OpenShift Kubernetes Engine:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736>

- 4 Optional: Specify custom metadata in the form of a **key:value** pair. For example use cases, see the GCP documentation for [setting custom metadata](#).
- 5 For <**project\_name**>, specify the name of the GCP project that you use for your cluster.
- 6 Specifies a single service account. Multiple service accounts are not supported.

## 2.4.2. Creating a compute 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 compute machine set custom resource (CR) sample and is named <**file\_name**>.yaml.  
 Ensure that you set the <**clusterID**> and <**role**> parameter values.
  - a. To list the compute machine sets in your cluster, run the following command:
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.

```
$ 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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

### 2.4.3. Configuring persistent disk types by using machine sets

You can configure the type of persistent disk that a machine set deploys machines on by editing the machine set YAML file.

For more information about persistent disk types, compatibility, regional availability, and limitations, see the GCP Compute Engine documentation about [persistent disks](#).

#### Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following line under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            type: <pd-disk-type> ①
```

- 1 Specify the persistent disk type. Valid values are **pd-ssd**, **pd-standard**, and **pd-balanced**. The default value is **pd-standard**.

## Verification

- Using the Google Cloud console, review the details for a machine deployed by the machine set and verify that the **Type** field matches the configured disk type.

### 2.4.4. Configuring Confidential VM by using machine sets

By editing the machine set YAML file, you can configure the Confidential VM options that a machine set uses for machines that it deploys.

For more information about Confidential VM features, functions, and compatibility, see the GCP Compute Engine documentation about [Confidential VM](#).



#### NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.



#### IMPORTANT

OpenShift Container Platform 4.17 does not support some Confidential Compute features, such as Confidential VMs with AMD Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP).

## Procedure

- In a text editor, open the YAML file for an existing machine set or create a new one.
- Edit the following section under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          confidentialCompute: Enabled 1
          onHostMaintenance: Terminate 2
          machineType: n2d-standard-8 3
...

```

- Specify whether Confidential VM is enabled. Valid values are **Disabled** or **Enabled**.
- Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VM does not support live VM migration.
- Specify a machine type that supports Confidential VM. Confidential VM supports the N2D and C2D series of machine types.

## Verification

- On the Google Cloud console, review the details for a machine deployed by the machine set and verify that the Confidential VM options match the values that you configured.

## 2.4.5. Machine sets that deploy machines as preemptible VM instances

You can save on costs by creating a compute machine set running on GCP that deploys machines as non-guaranteed preemptible VM instances. Preemptible VM instances utilize excess Compute Engine capacity and are less expensive than normal instances. You can use preemptible VM instances for workloads that can tolerate interruptions, such as batch or stateless, horizontally scalable workloads.

GCP Compute Engine can terminate a preemptible VM instance at any time. Compute Engine sends a preemption notice to the user indicating that an interruption will occur in 30 seconds. OpenShift Container Platform begins to remove the workloads from the affected instances when Compute Engine issues the preemption notice. An ACPI G3 Mechanical Off signal is sent to the operating system after 30 seconds if the instance is not stopped. The preemptible VM instance is then transitioned to a **TERMINATED** state by Compute Engine.

Interruptions can occur when using preemptible VM instances for the following reasons:

- There is a system or maintenance event
- The supply of preemptible VM instances decreases
- The instance reaches the end of the allotted 24-hour period for preemptible VM instances

When GCP terminates an instance, a termination handler running on the preemptible VM instance node deletes the machine resource. To satisfy the compute machine set **replicas** quantity, the compute machine set creates a machine that requests a preemptible VM instance.

### 2.4.5.1. Creating preemptible VM instances by using compute machine sets

You can launch a preemptible VM instance on GCP by adding **preemptible** to your compute machine set YAML file.

#### Procedure

- Add the following line under the **providerSpec** field:

```
providerSpec:  
  value:  
    preemptible: true
```

If **preemptible** is set to **true**, the machine is labelled as an **interruptable-instance** after the instance is launched.

## 2.4.6. Configuring Shielded VM options by using machine sets

By editing the machine set YAML file, you can configure the Shielded VM options that a machine set uses for machines that it deploys.

For more information about Shielded VM features and functionality, see the GCP Compute Engine documentation about [Shielded VM](#).

#### Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following section under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          shieldedInstanceConfig: ①
          integrityMonitoring: Enabled ②
          secureBoot: Disabled ③
          virtualizedTrustedPlatformModule: Enabled ④
# ...
```

- ① In this section, specify any Shielded VM options that you want.
- ② Specify whether integrity monitoring is enabled. Valid values are **Disabled** or **Enabled**.



#### NOTE

When integrity monitoring is enabled, you must not disable virtual trusted platform module (vTPM).

- ③ Specify whether UEFI Secure Boot is enabled. Valid values are **Disabled** or **Enabled**.
- ④ Specify whether vTPM is enabled. Valid values are **Disabled** or **Enabled**.

## Verification

- Using the Google Cloud console, review the details for a machine deployed by the machine set and verify that the Shielded VM options match the values that you configured.

## Additional resources

- [What is Shielded VM?](#)
  - [Secure Boot](#)
  - [Virtual Trusted Platform Module \(vTPM\)](#)
  - [Integrity monitoring](#)

### 2.4.7. Enabling customer-managed encryption keys for a machine set

Google Cloud Platform (GCP) Compute Engine allows users to supply an encryption key to encrypt data on disks at rest. The key is used to encrypt the data encryption key, not to encrypt the customer's data. By default, Compute Engine encrypts this data by using Compute Engine keys.

You can enable encryption with a customer-managed key in clusters that use the Machine API. You must first [create a KMS key](#) and assign the correct permissions to a service account. The KMS key name, key ring name, and location are required to allow a service account to use your key.



## NOTE

If you do not want to use a dedicated service account for the KMS encryption, the Compute Engine default service account is used instead. You must grant the default service account permission to access the keys if you do not use a dedicated service account. The Compute Engine default service account name follows the **service-@compute-system.iam.gserviceaccount.com** pattern.

## Procedure

1. To allow a specific service account to use your KMS key and to grant the service account the correct IAM role, run the following command with your KMS key name, key ring name, and location:

```
$ gcloud kms keys add-iam-policy-binding <key_name> \
    --keyring <key_ring_name> \
    --location <key_ring_location> \
    --member "serviceAccount:service-<project_number>@compute-
    system.iam.gserviceaccount.com" \
    --role roles/cloudkms.cryptoKeyEncrypterDecrypter
```

2. Configure the encryption key under the **providerSpec** field in your machine set YAML file. For example:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            - type:
                encryptionKey:
                  kmsKey:
                    name: machine-encryption-key ①
                    keyRing: openshift-encrpytion-ring ②
                    location: global ③
                    projectID: openshift-gcp-project ④
                    kmsKeyServiceAccount: openshift-service-account@openshift-gcp-
                    project.iam.gserviceaccount.com ⑤
```

- ① The name of the customer-managed encryption key that is used for the disk encryption.
- ② The name of the KMS key ring that the KMS key belongs to.
- ③ The GCP location in which the KMS key ring exists.
- ④ Optional: The ID of the project in which the KMS key ring exists. If a project ID is not set, the machine set **projectId** in which the machine set was created is used.

- 5 Optional: The service account that is used for the encryption request for the given KMS key. If a service account is not set, the Compute Engine default service account is used.

When a new machine is created by using the updated **providerSpec** object configuration, the disk encryption key is encrypted with the KMS key.

#### 2.4.8. Enabling GPU support for a compute machine set

Google Cloud Platform (GCP) Compute Engine enables users to add GPUs to VM instances. Workloads that benefit from access to GPU resources can perform better on compute machines with this feature enabled. OpenShift Container Platform on GCP supports NVIDIA GPU models in the A2 and N1 machine series.

Table 2.2. Supported GPU configurations

Model name	GPU type	Machine types <sup>[1]</sup>
NVIDIA A100	<b>nvidia-tesla-a100</b>	<ul style="list-style-type: none"><li>• <b>a2-highgpu-1g</b></li><li>• <b>a2-highgpu-2g</b></li><li>• <b>a2-highgpu-4g</b></li><li>• <b>a2-highgpu-8g</b></li><li>• <b>a2-megagpu-16g</b></li></ul>

Model name	GPU type	Machine types <sup>[1]</sup>
NVIDIA K80	<b>nvidia-tesla-k80</b>	<ul style="list-style-type: none"> <li>• n1-standard-1</li> </ul>
NVIDIA P100	<b>nvidia-tesla-p100</b>	<ul style="list-style-type: none"> <li>• n1-standard-2</li> </ul>
NVIDIA P4	<b>nvidia-tesla-p4</b>	<ul style="list-style-type: none"> <li>• n1-standard-4</li> </ul>
NVIDIA T4	<b>nvidia-tesla-t4</b>	<ul style="list-style-type: none"> <li>• n1-standard-8</li> </ul>
NVIDIA V100	<b>nvidia-tesla-v100</b>	<ul style="list-style-type: none"> <li>• n1-standard-16</li> <li>• n1-standard-32</li> <li>• n1-standard-64</li> <li>• n1-standard-96</li> <li>• n1-highmem-2</li> <li>• n1-highmem-4</li> <li>• n1-highmem-8</li> <li>• n1-highmem-16</li> <li>• n1-highmem-32</li> <li>• n1-highmem-64</li> <li>• n1-highmem-96</li> <li>• n1-highcpu-2</li> <li>• n1-highcpu-4</li> <li>• n1-highcpu-8</li> <li>• n1-highcpu-16</li> <li>• n1-highcpu-32</li> <li>• n1-highcpu-64</li> <li>• n1-highcpu-96</li> </ul>

1. For more information about machine types, including specifications, compatibility, regional availability, and limitations, see the GCP Compute Engine documentation about [N1 machine series](#), [A2 machine series](#), and [GPU regions and zones availability](#).

You can define which supported GPU to use for an instance by using the Machine API.

You can configure machines in the N1 machine series to deploy with one of the supported GPU types. Machines in the A2 machine series come with associated GPUs, and cannot use guest accelerators.



#### NOTE

GPUs for graphics workloads are not supported.

## Procedure

1. In a text editor, open the YAML file for an existing compute machine set or create a new one.
2. Specify a GPU configuration under the **providerSpec** field in your compute machine set YAML file. See the following examples of valid configurations:

### Example configuration for the A2 machine series

```
providerSpec:
  value:
    machineType: a2-highgpu-1g ①
    onHostMaintenance: Terminate ②
    restartPolicy: Always ③
```

- ① Specify the machine type. Ensure that the machine type is included in the A2 machine series.
- ② When using GPU support, you must set **onHostMaintenance** to **Terminate**.
- ③ Specify the restart policy for machines deployed by the compute machine set. Allowed values are **Always** or **Never**.

### Example configuration for the N1 machine series

```
providerSpec:
  value:
    gpus:
      - count: 1 ①
        type: nvidia-tesla-p100 ②
    machineType: n1-standard-1 ③
    onHostMaintenance: Terminate ④
    restartPolicy: Always ⑤
```

- ① Specify the number of GPUs to attach to the machine.
- ② Specify the type of GPUs to attach to the machine. Ensure that the machine type and GPU type are compatible.
- ③ Specify the machine type. Ensure that the machine type and GPU type are compatible.
- ④ When using GPU support, you must set **onHostMaintenance** to **Terminate**.
- ⑤ Specify the restart policy for machines deployed by the compute machine set. Allowed values are **Always** or **Never**.

## 2.4.9. Adding a GPU node to an existing OpenShift Container Platform cluster

You can copy and modify a default compute machine set configuration to create a GPU-enabled machine set and machines for the GCP cloud provider.

The following table lists the validated instance types:

Instance type	NVIDIA GPU accelerator	Maximum number of GPUs	Architecture
a2-highgpu-1g	A100	1	x86
n1-standard-4	T4	1	x86

## Procedure

1. Make a copy of an existing **MachineSet**.
2. In the new copy, change the machine set **name** in **metadata.name** and in both instances of **machine.openshift.io/cluster-api-machineset**.
3. Change the instance type to add the following two lines to the newly copied **MachineSet**:

```
machineType: a2-highgpu-1g
onHostMaintenance: Terminate
```

### Example a2-highgpu-1g.json file

```
{
  "apiVersion": "machine.openshift.io/v1beta1",
  "kind": "MachineSet",
  "metadata": {
    "annotations": {
      "machine.openshift.io/GPU": "0",
      "machine.openshift.io/memoryMb": "16384",
      "machine.openshift.io/vCPU": "4"
    },
    "creationTimestamp": "2023-01-13T17:11:02Z",
    "generation": 1,
    "labels": {
      "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p"
    },
    "name": "myclustername-2pt9p-worker-gpu-a",
    "namespace": "openshift-machine-api",
    "resourceVersion": "20185",
    "uid": "2daf4712-733e-4399-b4b4-d43cb1ed32bd"
  },
  "spec": {
    "replicas": 1,
    "selector": {
      "matchLabels": {
        "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p",
        "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-gpu-
a"
      }
    },
    "template": {
      "metadata": {
        "labels": {
          "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p",
          "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-gpu-
a"
        }
      }
    }
  }
}
```

```

        "machine.openshift.io/cluster-api-machine-role": "worker",
        "machine.openshift.io/cluster-api-machine-type": "worker",
        "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
    }
},
"spec": {
    "lifecycleHooks": {},
    "metadata": {},
    "providerSpec": {
        "value": {
            "apiVersion": "machine.openshift.io/v1beta1",
            "canIPForward": false,
            "credentialsSecret": {
                "name": "gcp-cloud-credentials"
            },
            "deletionProtection": false,
            "disks": [
                {
                    "autoDelete": true,
                    "boot": true,
                    "image": "projects/rhcos-cloud/global/images/rhcos-412-86-
202212081411-0-gcp-x86-64",
                    "labels": null,
                    "sizeGb": 128,
                    "type": "pd-ssd"
                }
            ],
            "kind": "GCPMachineProviderSpec",
            "machineType": "a2-highgpu-1g",
            "onHostMaintenance": "Terminate",
            "metadata": {
                "creationTimestamp": null
            },
            "networkInterfaces": [
                {
                    "network": "myclustername-2pt9p-network",
                    "subnetwork": "myclustername-2pt9p-worker-subnet"
                }
            ],
            "preemptible": true,
            "projectId": "myteam",
            "region": "us-central1",
            "serviceAccounts": [
                {
                    "email": "myclustername-2pt9p-w@myteam.iam.gserviceaccount.com",
                    "scopes": [
                        "https://www.googleapis.com/auth/cloud-platform"
                    ]
                }
            ],
            "tags": [
                "myclustername-2pt9p-worker"
            ],
            "userDataSecret": {
                "name": "worker-user-data"
            }
        }
    }
}

```

```

        },
        "zone": "us-central1-a"
    }
}
},
{
"status": {
    "availableReplicas": 1,
    "fullyLabeledReplicas": 1,
    "observedGeneration": 1,
    "readyReplicas": 1,
    "replicas": 1
}
}
}

```

- View the existing nodes, machines, and machine sets by running the following command. Note that each node is an instance of a machine definition with a specific GCP region and OpenShift Container Platform role.

```
$ oc get nodes
```

#### Example output

NAME	STATUS	ROLES	AGE	VERSION
myclustername-2pt9p-master-0.c.openshift-qe.internal 8h v1.30.3		Ready		control-plane,master
myclustername-2pt9p-master-1.c.openshift-qe.internal 8h v1.30.3		Ready		control-plane,master
myclustername-2pt9p-master-2.c.openshift-qe.internal 8h v1.30.3		Ready		control-plane,master
myclustername-2pt9p-worker-a-mxtnz.c.openshift-qe.internal 8h v1.30.3		Ready		worker
myclustername-2pt9p-worker-b-9pzzn.c.openshift-qe.internal 8h v1.30.3		Ready		worker
myclustername-2pt9p-worker-c-6pb6.c.openshift-qe.internal 8h v1.30.3		Ready		worker
myclustername-2pt9p-worker-gpu-a-wxcr6.c.openshift-qe.internal 4h35m v1.30.3	Ready			worker

- View the machines and machine sets that exist in the **openshift-machine-api** namespace by running the following command. Each compute machine set is associated with a different availability zone within the GCP region. The installer automatically load balances compute machines across availability zones.

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

#### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
myclustername-2pt9p-worker-a	1	1	1	1	8h
myclustername-2pt9p-worker-b	1	1	1	1	8h
myclustername-2pt9p-worker-c	1	1			8h
myclustername-2pt9p-worker-f	0	0			8h

- View the machines that exist in the **openshift-machine-api** namespace by running the following command. You can only configure one compute machine per set, although you can scale a compute machine set to add a node in a particular region and zone.

```
$ oc get machines -n openshift-machine-api | grep worker
```

### Example output

myclustername-2pt9p-worker-a-mxtnz	Running	n2-standard-4	us-central1	us-central1-a	8h
myclustername-2pt9p-worker-b-9pzzn	Running	n2-standard-4	us-central1	us-central1-b	8h
myclustername-2pt9p-worker-c-6pb6	Running	n2-standard-4	us-central1	us-central1-c	8h

- Make a copy of one of the existing compute **MachineSet** definitions and output the result to a JSON file by running the following command. This will be the basis for the GPU-enabled compute machine set definition.

```
$ oc get machineset myclustername-2pt9p-worker-a -n openshift-machine-api -o json > <output_file.json>
```

- Edit the JSON file to make the following changes to the new **MachineSet** definition:

- Rename the machine set **name** by inserting the substring **gpu** in **metadata.name** and in both instances of **machine.openshift.io/cluster-api-machineset**.
- Change the **machineType** of the new **MachineSet** definition to **a2-highgpu-1g**, which includes an NVIDIA A100 GPU.

```
jq .spec.template.spec.providerSpec.value.machineType ocp_4.17_machineset-a2-highgpu-1g.json  
"a2-highgpu-1g"
```

The **<output\_file.json>** file is saved as **ocp\_4.17\_machineset-a2-highgpu-1g.json**.

- Update the following fields in **ocp\_4.17\_machineset-a2-highgpu-1g.json**:

- Change **.metadata.name** to a name containing **gpu**.
- Change **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** to match the new **.metadata.name**.
- Change **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** to match the new **.metadata.name**.
- Change **.spec.template.spec.providerSpec.value.MachineType** to **a2-highgpu-1g**.
- Add the following line under **machineType**: `onHostMaintenance": "Terminate"`. For example:

```
"machineType": "a2-highgpu-1g",  
"onHostMaintenance": "Terminate",
```

- To verify your changes, perform a **diff** of the original compute definition and the new GPU-enabled node definition by running the following command:

```
$ oc get machineset/myclustername-2pt9p-worker-a -n openshift-machine-api -o json | diff ocp_4.17_machineset-a2-highgpu-1g.json -
```

### Example output

```
15c15
<      "name": "myclustername-2pt9p-worker-gpu-a",
---
>      "name": "myclustername-2pt9p-worker-a",
25c25
<          "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
---
>          "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-a"
34c34
<              "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
---
>              "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
a"
59,60c59
<          "machineType": "a2-highgpu-1g",
<          "onHostMaintenance": "Terminate",
---
>          "machineType": "n2-standard-4",
```

- Create the GPU-enabled compute machine set from the definition file by running the following command:

```
$ oc create -f ocp_4.17_machineset-a2-highgpu-1g.json
```

### Example output

```
machineset.machine.openshift.io/myclustername-2pt9p-worker-gpu-a created
```

## Verification

- View the machine set you created by running the following command:

```
$ oc -n openshift-machine-api get machinesets | grep gpu
```

The MachineSet replica count is set to **1** so a new **Machine** object is created automatically.

### Example output

```
myclustername-2pt9p-worker-gpu-a 1 1 1 5h24m
```

- View the **Machine** object that the machine set created by running the following command:

```
$ oc -n openshift-machine-api get machines | grep gpu
```

## Example output

```
myclustername-2pt9p-worker-gpu-a-wxcr6 Running a2-highgpu-1g us-central1 us-central1-a 5h25m
```



### NOTE

Note that there is no need to specify a namespace for the node. The node definition is cluster scoped.

## 2.4.10. Deploying the Node Feature Discovery Operator

After the GPU-enabled node is created, you need to discover the GPU-enabled node so it can be scheduled. To do this, install the Node Feature Discovery (NFD) Operator. The NFD Operator identifies hardware device features in nodes. It solves the general problem of identifying and cataloging hardware resources in the infrastructure nodes so they can be made available to OpenShift Container Platform.

### Procedure

1. Install the Node Feature Discovery Operator from **OperatorHub** in the OpenShift Container Platform console.
2. After installing the NFD Operator into **OperatorHub**, select **Node Feature Discovery** from the installed Operators list and select **Create instance**. This installs the **nfd-master** and **nfd-worker** pods, one **nfd-worker** pod for each compute node, in the **openshift-nfd** namespace.
3. Verify that the Operator is installed and running by running the following command:

```
$ oc get pods -n openshift-nfd
```

## Example output

NAME	READY	STATUS	RESTARTS	AGE
nfd-controller-manager-8646fcbb65-x5qgk	2/2	Running	7 (8h ago)	1d

4. Browse to the installed Operator in the console and select **Create Node Feature Discovery**.
5. Select **Create** to build a NFD custom resource. This creates NFD pods in the **openshift-nfd** namespace that poll the OpenShift Container Platform nodes for hardware resources and catalogue them.

### Verification

1. After a successful build, verify that a NFD pod is running on each nodes by running the following command:

```
$ oc get pods -n openshift-nfd
```

## Example output

NAME	READY	STATUS	RESTARTS	AGE
------	-------	--------	----------	-----

nfd-controller-manager-8646fcbb65-x5qgk	2/2	Running	7 (8h ago)	12d
nfd-master-769656c4cb-w9vrv	1/1	Running	0	12d
nfd-worker-qjxb2	1/1	Running	3 (3d14h ago)	12d
nfd-worker-xtz9b	1/1	Running	5 (3d14h ago)	12d

The NFD Operator uses vendor PCI IDs to identify hardware in a node. NVIDIA uses the PCI ID **10de**.

- View the NVIDIA GPU discovered by the NFD Operator by running the following command:

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

### Example output

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

**10de** appears in the node feature list for the GPU-enabled node. This mean the NFD Operator correctly identified the node from the GPU-enabled MachineSet.

## 2.5. CREATING A COMPUTE MACHINE SET ON IBM CLOUD

You can create a different compute machine set to serve a specific purpose in your OpenShift Container Platform cluster on IBM Cloud®. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.5.1. Sample YAML for a compute machine set custom resource on IBM Cloud

This sample YAML defines a compute machine set that runs in a specified IBM Cloud® zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/<role>: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machine-role: <role> 8
      machine.openshift.io/cluster-api-machine-type: <role> 9
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
spec:
  metadata:
    labels:
      node-role.kubernetes.io/<role>: ""
  providerSpec:
    value:
      apiVersion: ibmcloudproviderconfig.openshift.io/v1beta1
      credentialsSecret:
        name: ibmcloud-credentials
      image: <infrastructure_id>-rhcos 11
      kind: IBMCloudMachineProviderSpec
      primaryNetworkInterface:
        securityGroups:
          - <infrastructure_id>-sg-cluster-wide
          - <infrastructure_id>-sg-openshift-net
        subnet: <infrastructure_id>-subnet-compute-<zone> 12
      profile: <instance_profile> 13
      region: <region> 14
      resourceGroup: <resource_group> 15
      userDataSecret:
        name: <role>-user-data 16
      vpc: <vpc_name> 17
      zone: <zone> 18

```

**1 | 5 | 7** The infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

**2 3 8 9 16** The node label to add.

**4 6 10** The infrastructure ID, node label, and region.

**11** The custom Red Hat Enterprise Linux CoreOS (RHCOS) image that was used for cluster installation.

**12** The infrastructure ID and zone within your region to place machines on. Be sure that your region supports the zone that you specify.

**13** Specify the [IBM Cloud® instance profile](#).

**14** Specify the region to place machines on.

**15** The resource group that machine resources are placed in. This is either an existing resource group specified at installation time, or an installer-created resource group named based on the infrastructure ID.

**17** The VPC name.

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

### 2.5.2. Creating a compute 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 compute 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
```

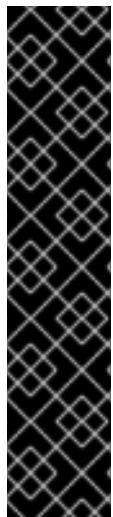
### Example output

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

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

## 2.6. CREATING A COMPUTE MACHINE SET ON IBM POWER VIRTUAL SERVER

You can create a different compute machine set to serve a specific purpose in your OpenShift Container Platform cluster on IBM Power® Virtual Server. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.6.1. Sample YAML for a compute machine set custom resource on IBM Power Virtual Server

This sample YAML file defines a compute machine set that runs in a specified IBM Power® Virtual Server zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/<role>**: "".

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

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

```

metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
    name: <infrastructure_id>-<role>-<region> 4
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machine-role: <role> 8
      machine.openshift.io/cluster-api-machine-type: <role> 9
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
spec:
  metadata:
    labels:
      node-role.kubernetes.io/<role>: ""
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1
      credentialsSecret:
        name: powervs-credentials
      image:
        name: rhcos-<infrastructure_id> 11
        type: Name
      keyPairName: <infrastructure_id>-key
      kind: PowerVSMachineProviderConfig
      memoryGiB: 32
      network:
        regex: ^DHCP SERVER[0-9a-z]{32}_Private$ 12
        type: RegEx
      processorType: Shared
      processors: "0.5"
      serviceInstance:
        id: <ibm_power_vs_service_instance_id>
        type: ID 12
      systemType: s922
      userDataSecret:
        name: <role>-user-data

```

**1 5 7** The infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

**2 3 8 9** The node label to add.

**4 6 10** The infrastructure ID, node label, and region.

- 11** The custom Red Hat Enterprise Linux CoreOS (RHCOS) image that was used for cluster installation.
- 12** The infrastructure ID within your region to place machines on.

## 2.6.2. Creating a compute 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 compute 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-vplvk-infra-us-east-1a	1	1	1	1	11m
agl030519-vplvk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplvk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplvk-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 compute machine set is available, the **DESIRED** and **CURRENT** values match. If the compute machine set is not available, wait a few minutes and run the command again.

## 2.7. CREATING A COMPUTE MACHINE SET ON NUTANIX

You can create a different compute machine set to serve a specific purpose in your OpenShift Container Platform cluster on Nutanix. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.7.1. Sample YAML for a compute machine set custom resource on Nutanix

This sample YAML defines a Nutanix compute machine set that creates nodes that are labeled with **node-role.kubernetes.io/<role>**: "".

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

#### Values obtained by using the OpenShift CLI

In the following example, you can obtain some of the values for your cluster by using the OpenShift CLI (**oc**).

#### Infrastructure ID

The **<infrastructure\_id>** string is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ①
```

```

machine.openshift.io/cluster-api-machine-role: <role> ②
machine.openshift.io/cluster-api-machine-type: <role>
name: <infrastructure_id>-<role>-<zone> ③
namespace: openshift-machine-api
annotations: ④
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
replicas: 3
selector:
matchLabels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone>
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>-<zone>
spec:
metadata:
labels:
    node-role.kubernetes.io/<role>: ""
providerSpec:
value:
apiVersion: machine.openshift.io/v1
bootType: "" ⑤
categories: ⑥
- key: <category_name>
  value: <category_value>
cluster: ⑦
type: uuid
uuid: <cluster_uuid>
credentialsSecret:
name: nutanix-credentials
image:
name: <infrastructure_id>-rhcos ⑧
type: name
kind: NutanixMachineProviderConfig
memorySize: 16Gi ⑨
project: ⑩
type: name
name: <project_name>
subnets:
- type: uuid
  uuid: <subnet_uuid>
systemDiskSize: 120Gi ⑪
userDataSecret:
name: <user_data_secret> ⑫
vcpuSockets: 4 ⑬
vcpusPerSocket: 1 ⑭

```

- 1 For <**infrastructure\_id**>, specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.
- 2 Specify the node label to add.
- 3 Specify the infrastructure ID, node label, and zone.
- 4 Annotations for the cluster autoscaler.
- 5 Specifies the boot type that the compute machines use. For more information about boot types, see [Understanding UEFI, Secure Boot, and TPM in the Virtualized Environment](#). Valid values are **Legacy**, **SecureBoot**, or **UEFI**. The default is **Legacy**.



#### NOTE

You must use the **Legacy** boot type in OpenShift Container Platform 4.17.

- 6 Specify one or more Nutanix Prism categories to apply to compute machines. This stanza requires **key** and **value** parameters for a category key-value pair that exists in Prism Central. For more information about categories, see [Category management](#).
- 7 Specify a Nutanix Prism Element cluster configuration. In this example, the cluster type is **uuid**, so there is a **uuid** stanza.
- 8 Specify the image to use. Use an image from an existing default compute machine set for the cluster.
- 9 Specify the amount of memory for the cluster in Gi.
- 10 Specify the Nutanix project that you use for your cluster. In this example, the project type is **name**, so there is a **name** stanza.
- 11 Specify the size of the system disk in Gi.
- 12 Specify the name of the secret in the user data YAML file that is in the **openshift-machine-api** namespace. Use the value that installation program populates in the default compute machine set.
- 13 Specify the number of vCPU sockets.
- 14 Specify the number of vCPUs per socket.

### 2.7.2. Creating a compute 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 compute 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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

### 2.7.3. Failure domains for Nutanix clusters

To add or update the failure domain configuration on a Nutanix cluster, you must make coordinated changes to several resources. The following actions are required:

1. Modify the cluster infrastructure custom resource (CR).
2. Modify the cluster control plane machine set CR.
3. Modify or replace the compute machine set CRs.

For more information, see "Adding failure domains to an existing Nutanix cluster" in the *Post-installation configuration* content.

### Additional resources

- [Adding failure domains to an existing Nutanix cluster](#)

## 2.8. CREATING A COMPUTE MACHINE SET ON OPENSTACK

You can create a different compute machine set to serve a specific purpose in your OpenShift Container Platform cluster on Red Hat OpenStack Platform (RHOSP). For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.8.1. Sample YAML for a compute machine set custom resource on RHOSP

This sample YAML defines a compute machine set that runs on Red Hat OpenStack Platform (RHOSP) and creates nodes that are labeled with **node-role.kubernetes.io/<role>**: "".

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 10
    spec:
```

```

providerSpec:
  value:
    apiVersion: machine.openshift.io/v1alpha1
    cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: <nova_flavor>
    image: <glance_image_name_or_location>
    serverGroupID: <optional_UUID_of_server_group> ⑪
    kind: OpenstackProviderSpec
  networks: ⑫
    - filter: {}
      subnets:
        - filter:
          name: <subnet_name>
          tags: openshiftClusterID=<infrastructure_id> ⑬
    primarySubnet: <rhosp_subnet_UUID> ⑭
    securityGroups:
      - filter: {}
        name: <infrastructure_id>-worker ⑮
    serverMetadata:
      Name: <infrastructure_id>-worker ⑯
      openshiftClusterID: <infrastructure_id> ⑰
    tags:
      - openshiftClusterID=<infrastructure_id> ⑱
    trunk: true
    userDataSecret:
      name: worker-user-data ⑲
    availabilityZone: <optional_openstack_availability_zone>

```

① ⑤ ⑦ ⑬ ⑮ ⑯ ⑰ ⑱ Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

② ③ ⑧ ⑨ ⑲ Specify the node label to add.

④ ⑥ ⑩ Specify the infrastructure ID and node label.

⑪ To set a server group policy for the MachineSet, enter the value that is returned from [creating a server group](#). For most deployments, **anti-affinity** or **soft-anti-affinity** policies are recommended.

⑫ Required for deployments to multiple networks. To specify multiple networks, add another entry in the networks array. Also, you must include the network that is used as the **primarySubnet** value.

⑬ Specify the RHOSP subnet that you want the endpoints of nodes to be published on. Usually, this is the same subnet that is used as the value of **machinesSubnet** in the **install-config.yaml** file.

## 2.8.2. Sample YAML for a compute machine set custom resource that uses SR-IOV on RHOSP

If you configured your cluster for single-root I/O virtualization (SR-IOV), you can create compute machine sets that use that technology.

This sample YAML defines a compute machine set that uses SR-IOV networks. The nodes that it creates are labeled with **node-role.openshift.io/<node\_role>**: ""

In this sample, **infrastructure\_id** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **node\_role** is the node label to add.

The sample assumes two SR-IOV networks that are named "radio" and "uplink". The networks are used in port definitions in the **spec.template.spec.providerSpec.value.ports** list.



## NOTE

Only parameters that are specific to SR-IOV deployments are described in this sample. To review a more general sample, see "Sample YAML for a compute machine set custom resource on RHOSP".

### An example compute machine set that uses SR-IOV networks

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
    spec:
      metadata:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1alpha1
            cloudName: openstack
            cloudsSecret:
              name: openstack-cloud-credentials
              namespace: openshift-machine-api
            flavor: <nova_flavor>
            image: <glance_image_name_or_location>
            serverGroupID: <optional_UUID_of_server_group>
            kind: OpenstackProviderSpec

```

```

networks:
  - subnets:
    - UUID: <machines_subnet_UUID>
ports:
  - networkID: <radio_network_UUID> ①
    nameSuffix: radio
    fixedIPs:
      - subnetID: <radio_subnet_UUID> ②
tags:
  - sriov
  - radio
vnicType: direct ③
portSecurity: false ④
  - networkID: <uplink_network_UUID> ⑤
    nameSuffix: uplink
    fixedIPs:
      - subnetID: <uplink_subnet_UUID> ⑥
tags:
  - sriov
  - uplink
vnicType: direct ⑦
portSecurity: false ⑧
primarySubnet: <machines_subnet_UUID>
securityGroups:
  - filter: {}
    name: <infrastructure_id>-<node_role>
serverMetadata:
  Name: <infrastructure_id>-<node_role>
  openshiftClusterID: <infrastructure_id>
tags:
  - openshiftClusterID=<infrastructure_id>
trunk: true
userDataSecret:
  name: <node_role>-user-data
availabilityZone: <optional_openstack_availability_zone>

```

① ⑤ Enter a network UUID for each port.

② ⑥ Enter a subnet UUID for each port.

③ ⑦ The value of the **vnicType** parameter must be **direct** for each port.

④ ⑧ The value of the **portSecurity** parameter must be **false** for each port.

You cannot set security groups and allowed address pairs for ports when port security is disabled. Setting security groups on the instance applies the groups to all ports that are attached to it.



## IMPORTANT

After you deploy compute machines that are SR-IOV-capable, you must label them as such. For example, from a command line, enter:

```
$ oc label node <NODE_NAME> feature.node.kubernetes.io/network-sriov.capable="true"
```



## NOTE

Trunking is enabled for ports that are created by entries in the networks and subnets lists. The names of ports that are created from these lists follow the pattern **<machine\_name>-<nameSuffix>**. The **nameSuffix** field is required in port definitions.

You can enable trunking for each port.

Optionally, you can add tags to ports as part of their **tags** lists.

## Additional resources

- [Preparing to install a cluster that uses SR-IOV or OVS-DPDK on OpenStack](#)

### 2.8.3. Sample YAML for SR-IOV deployments where port security is disabled

To create single-root I/O virtualization (SR-IOV) ports on a network that has port security disabled, define a compute machine set that includes the ports as items in the **spec.template.spec.providerSpec.value.ports** list. This difference from the standard SR-IOV compute machine set is due to the automatic security group and allowed address pair configuration that occurs for ports that are created by using the network and subnet interfaces.

Ports that you define for machines subnets require:

- Allowed address pairs for the API and ingress virtual IP ports
- The compute security group
- Attachment to the machines network and subnet



## NOTE

Only parameters that are specific to SR-IOV deployments where port security is disabled are described in this sample. To review a more general sample, see [Sample YAML for a compute machine set custom resource that uses SR-IOV on RHOSP](#).

### An example compute machine set that uses SR-IOV networks and has port security disabled

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
```

```

name: <infrastructure_id>-<node_role>
namespace: openshift-machine-api
spec:
replicas: <number_of_replicas>
selector:
matchLabels:
  machine.openshift.io/cluster-api-cluster: <infrastructure_id>
  machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
template:
metadata:
labels:
  machine.openshift.io/cluster-api-cluster: <infrastructure_id>
  machine.openshift.io/cluster-api-machine-role: <node_role>
  machine.openshift.io/cluster-api-machine-type: <node_role>
  machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
spec:
metadata: {}
providerSpec:
value:
apiVersion: machine.openshift.io/v1alpha1
cloudName: openstack
cloudsSecret:
name: openstack-cloud-credentials
namespace: openshift-machine-api
flavor: <nova_flavor>
image: <glance_image_name_or_location>
kind: OpenstackProviderSpec
ports:
- allowedAddressPairs: ①
  - ipAddress: <API_VIP_port_IP>
  - ipAddress: <ingress_VIP_port_IP>
fixedIPs:
  - subnetID: <machines_subnet_UUID> ②
nameSuffix: nodes
networkID: <machines_network_UUID> ③
securityGroups:
  - <compute_security_group_UUID> ④
- networkID: <SRIOV_network_UUID>
nameSuffix: sriov
fixedIPs:
  - subnetID: <SRIOV_subnet_UUID>
tags:
  - sriov
vnicType: direct
portSecurity: False
primarySubnet: <machines_subnet_UUID>
serverMetadata:
  Name: <infrastructure_ID>-<node_role>
  openshiftClusterID: <infrastructure_id>
tags:
  - openshiftClusterID=<infrastructure_id>
trunk: false
userDataSecret:
  name: worker-user-data

```

- 1** Specify allowed address pairs for the API and ingress ports.
- 2** **3** Specify the machines network and subnet.
- 4** Specify the compute machines security group.



#### NOTE

Trunking is enabled for ports that are created by entries in the networks and subnets lists. The names of ports that are created from these lists follow the pattern **<machine\_name>-<nameSuffix>**. The **nameSuffix** field is required in port definitions.

You can enable trunking for each port.

Optionally, you can add tags to ports as part of their **tags** lists.

### 2.8.4. Creating a compute 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 compute 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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

## 2.9. CREATING A COMPUTE MACHINE SET ON VSphere

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



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.9.1. Sample YAML for a compute machine set custom resource on vSphere

This sample YAML defines a compute machine set that runs on VMware vSphere and creates nodes that are labeled with **node-role.kubernetes.io/<role>**: "".

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  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> ③
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> ④
template:
  metadata:
    creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ⑤
    machine.openshift.io/cluster-api-machine-role: <role> ⑥
    machine.openshift.io/cluster-api-machine-type: <role> ⑦
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> ⑧
spec:
  metadata:
    creationTimestamp: null
  labels:
    node-role.kubernetes.io/<role>: "" ⑨
providerSpec:
  value:
    apiVersion: vsphereprovider.openshift.io/v1beta1
    credentialsSecret:
      name: vsphere-cloud-credentials
    diskGiB: 120
    kind: VSphereMachineProviderSpec
    memoryMiB: 8192
    metadata:
      creationTimestamp: null
    network:
      devices:
        - networkName: "<vm_network_name>" ⑩
    numCPUs: 4
    numCoresPerSocket: 1
    snapshot: ""
    template: <vm_template_name> ⑪
    userDataSecret:
      name: worker-user-data
    workspace:
      datacenter: <vcenter_data_center_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. If you have the OpenShift CLI (**oc**) installed, you can obtain the infrastructure ID by running the following command:

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

**2 4 8** Specify the infrastructure ID and node label.

**6 7 9** Specify the node label to add.

**10**

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

- 11 Specify the vSphere VM template to use, such as **user-5ddjd-rhcos**.
- 12 Specify the vCenter data center to deploy the compute machine set on.
- 13 Specify the vCenter datastore to deploy the compute machine set on.
- 14 Specify the path to the vSphere VM folder in vCenter, such as **/dc1/vm/user-inst-5ddjd**.
- 15 Specify the vSphere resource pool for your VMs.
- 16 Specify the vCenter server IP or fully qualified domain name.

### 2.9.2. Minimum required vCenter privileges for compute machine set management

To manage compute machine sets in an OpenShift Container Platform cluster on vCenter, you must use an account with privileges to read, create, and delete the required resources. Using an account that has global administrative privileges is the simplest way to access all of the necessary permissions.

If you cannot use an account with global administrative privileges, you must create roles to grant the minimum required privileges. The following table lists the minimum vCenter roles and privileges that are required to create, scale, and delete compute machine sets and to delete machines in your OpenShift Container Platform cluster.

**Example 2.1. Minimum vCenter roles and privileges required for compute machine set management**

vSphere object for role	When required	Required privileges
vSphere vCenter	Always	<b>InventoryService.Tagging.AttachTag</b> <b>InventoryService.Tagging.CreateCategory</b> <b>InventoryService.Tagging.CreateTag</b> <b>InventoryService.Tagging.DeleteCategory</b> <b>InventoryService.Tagging.DeleteTag</b> <b>InventoryService.Tagging.EditCategory</b> <b>InventoryService.Tagging.EditTag</b> <b>Sessions.ValidateSession</b> <b>StorageProfile.Update<sup>1</sup></b> <b>StorageProfile.View<sup>1</sup></b>
vSphere vCenter Cluster	Always	<b>Resource.AssignVMToPool</b>
vSphere datastore	Always	<b>Datastore.AllocateSpace</b> <b>Datastore.Browse</b>

vSphere object for role	When required	Required privileges
vSphere Port Group	Always	<b>Network.Assign</b>
Virtual Machine Folder	Always	<b>VirtualMachine.Config.Add</b> <b>RemoveDevice</b> <b>VirtualMachine.Config.AdvancedConfig</b> <b>VirtualMachine.Config.Annotation</b> <b>VirtualMachine.Config.CPUCount</b> <b>VirtualMachine.Config.DiskExtend</b> <b>VirtualMachine.Config.Memory</b> <b>VirtualMachine.Config.Settings</b> <b>VirtualMachine.Interact.PowerOff</b> <b>VirtualMachine.Interact.PowerOn</b> <b>VirtualMachine.Inventory.CreateFromExisting</b> <b>VirtualMachine.Inventory.Delete</b> <b>VirtualMachine.Provisioning.Clone</b>
vSphere vCenter data center	If the installation program creates the virtual machine folder	<b>Resource.AssignVMTToPool</b> <b>VirtualMachine.Provisioning.DeployTemplate</b>

<sup>1</sup>The **StorageProfile.Update** and **StorageProfile.View** permissions are required only for storage backends that use the Container Storage Interface (CSI).

The following table details the permissions and propagation settings that are required for compute machine set management.

#### Example 2.2. Required permissions and propagation settings

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter	Always	Not required	Listed required privileges
vSphere vCenter data center	Existing folder	Not required	<b>ReadOnly</b> permission

vSphere object	Folder type	Propagate to children	Permissions required
	Installation program creates the folder	Required	Listed required privileges
vSphere vCenter Cluster	Always	Required	Listed required privileges
vSphere vCenter datastore	Always	Not required	Listed required privileges
vSphere Switch	Always	Not required	<b>ReadOnly</b> permission
vSphere Port Group	Always	Not required	Listed required privileges
vSphere vCenter Virtual Machine Folder	Existing folder	Required	Listed required privileges

For more information about creating an account with only the required privileges, see [vSphere Permissions and User Management Tasks](#) in the vSphere documentation.

### 2.9.3. Requirements for clusters with user-provisioned infrastructure to use compute machine sets

To use compute machine sets on clusters that have user-provisioned infrastructure, you must ensure that your cluster configuration supports using the Machine API.

#### Obtaining the infrastructure ID

To create compute machine sets, you must be able to supply the infrastructure ID for your cluster.

#### Procedure

- To obtain the infrastructure ID for your cluster, run the following command:

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

#### Satisfying vSphere credentials requirements

To use compute machine sets, the Machine API must be able to interact with vCenter. Credentials that authorize the Machine API components to interact with vCenter must exist in a secret in the **openshift-machine-api** namespace.

#### Procedure

1. To determine whether the required credentials exist, run the following command:

```
$ oc get secret \
-n openshift-machine-api vsphere-cloud-credentials \
-o go-template='{{range $k,$v := .data}}{{printf "%s: " $k}}{{if not $v}}{{$v}}{{else}}{{$v}}|'
```

```
base64decode}}}}{{end}}}}{{"\n" }}}}{{end}}}'
```

### Sample output

```
<vcenter-server>.password=<openshift-user-password>
<vcenter-server>.username=<openshift-user>
```

where **<vcenter-server>** is the IP address or fully qualified domain name (FQDN) of the vCenter server and **<openshift-user>** and **<openshift-user-password>** are the OpenShift Container Platform administrator credentials to use.

- If the secret does not exist, create it by running the following command:

```
$ oc create secret generic vsphere-cloud-credentials \
-n openshift-machine-api \
--from-literal=<vcenter-server>.username=<openshift-user> --from-literal=<vcenter-
server>.password=<openshift-user-password>
```

### Satisfying Ignition configuration requirements

Provisioning virtual machines (VMs) requires a valid Ignition configuration. The Ignition configuration contains the **machine-config-server** address and a system trust bundle for obtaining further Ignition configurations from the Machine Config Operator.

By default, this configuration is stored in the **worker-user-data** secret in the **machine-api-operator** namespace. Compute machine sets reference the secret during the machine creation process.

### Procedure

- To determine whether the required secret exists, run the following command:

```
$ oc get secret \
-n openshift-machine-api worker-user-data \
-o go-template='{{range $k,$v := .data}}{{printf "%s: " $k}}{{if not $v}}{{($v)}}{{else}}{{($v)}}{{if $v | base64decode}}{{end}}{{"\n" }}{{end}}'
```

### Sample output

```
disableTemplating: false
userData: ①
{
  "ignition": {
    ...
  },
  ...
}
```

① The full output is omitted here, but should have this format.

- If the secret does not exist, create it by running the following command:

```
$ oc create secret generic worker-user-data \
-n openshift-machine-api \
--from-file=<installation_directory>/worker.ign
```

■ where **<installation\_directory>** is the directory that was used to store your installation assets during cluster installation.

## Additional resources

- [Understanding the Machine Config Operator](#)
- [Installing RHCOS and starting the OpenShift Container Platform bootstrap process](#)

### 2.9.4. Creating a compute 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.



#### NOTE

Clusters that are installed with user-provisioned infrastructure have a different networking stack than clusters with infrastructure that is provisioned by the installation program. As a result of this difference, automatic load balancer management is unsupported on clusters that have user-provisioned infrastructure. For these clusters, a compute machine set can only create **worker** and **infra** type machines.

#### Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.
- Have the necessary permissions to deploy VMs in your vCenter instance and have the required access to the datastore specified.
- If your cluster uses user-provisioned infrastructure, you have satisfied the specific Machine API requirements for that configuration.

#### Procedure

1. Create a new YAML file that contains the compute 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.

- c. If you are creating a compute machine set for a cluster that has user-provisioned infrastructure, note the following important values:

## Example vSphere providerSpec values

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
template:
...
spec:
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1beta1
      credentialsSecret:
        name: vsphere-cloud-credentials 1
      diskGiB: 120
      kind: VSphereMachineProviderSpec
      memoryMiB: 16384
      network:
        devices:
          - networkName: "<vm_network_name>"
      numCPUs: 4
      numCoresPerSocket: 4
      snapshot: ""
      template: <vm_template_name> 2
      userDataSecret:
        name: worker-user-data 3
      workspace:
        datacenter: <vcenter_data_center_name>
        datastore: <vcenter_datastore_name>
        folder: <vcenter_vm_folder_path>
        resourcepool: <vsphere_resource_pool>
        server: <vcenter_server_address> 4

```

- 1** The name of the secret in the **openshift-machine-api** namespace that contains the required vCenter credentials.
- 2** The name of the RHCOS VM template for your cluster that was created during installation.
- 3** The name of the secret in the **openshift-machine-api** namespace that contains the required Ignition configuration credentials.
- 4** The IP address or fully qualified domain name (FQDN) of the vCenter server.

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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

### 2.9.5. Adding tags to machines by using machine sets

OpenShift Container Platform adds a cluster-specific tag to each virtual machine (VM) that it creates. The installation program uses these tags to select the VMs to delete when uninstalling a cluster.

In addition to the cluster-specific tags assigned to VMs, you can configure a machine set to add up to 10 additional vSphere tags to the VMs it provisions.

#### Prerequisites

- You have access to an OpenShift Container Platform cluster installed on vSphere using an account with **cluster-admin** permissions.
- You have access to the VMware vCenter console associated with your cluster.
- You have created a tag in the vCenter console.
- You have installed the OpenShift CLI (**oc**).

#### Procedure

1. Use the vCenter console to find the tag ID for any tag that you want to add to your machines:
  - a. Log in to the vCenter console.
  - b. From the **Home** menu, click **Tags & Custom Attributes**.
  - c. Select a tag that you want to add to your machines.
  - d. Use the browser URL for the tag that you select to identify the tag ID.

#### Example tag URL

`https://vcenter.example.com/ui/app/tags/tag/urn:vmomi:InventoryServiceTag:208e713cae3-4b7f-918e-4051ca7d1f97:GLOBAL/permissions`

#### Example tag ID

`urn:vmomi:InventoryServiceTag:208e713cae3-4b7f-918e-4051ca7d1f97:GLOBAL`

2. In a text editor, open the YAML file for an existing machine set or create a new one.
3. Edit the following lines under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          tagIDs: ①
          - <tag_id_value> ②
# ...
```

- ① Specify a list of up to 10 tags to add to the machines that this machine set provisions.
- ② Specify the value of the tag that you want to add to your machines. For example, **urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL**.

## 2.10. CREATING A COMPUTE MACHINE SET ON BARE METAL

You can create a different compute machine set to serve a specific purpose in your OpenShift Container Platform cluster on bare metal. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.



### IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

### 2.10.1. Sample YAML for a compute machine set custom resource on bare metal

This sample YAML defines a compute machine set that runs on bare metal and creates nodes that are labeled with **node-role.kubernetes.io/<role>: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

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

```

creationTimestamp: null
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> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 4
template:
  metadata:
    creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
    machine.openshift.io/cluster-api-machine-role: <role> 6
    machine.openshift.io/cluster-api-machine-type: <role> 7
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
spec:
  metadata:
    creationTimestamp: null
  labels:
    node-role.kubernetes.io/<role>: "" 9
providerSpec:
  value:
    apiVersion: baremetal.cluster.k8s.io/v1alpha1
    hostSelector: {}
    image:
      checksum: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2.<md5sum>
  10
      url: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2 11
    kind: BareMetalMachineProviderSpec
    metadata:
      creationTimestamp: null
    userData:
      name: worker-user-data

```

**1 3 5** Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI (**oc**) installed, you can obtain the infrastructure ID by running the following command:

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

**2 4 8** Specify the infrastructure ID and node label.

**6 7 9** Specify the node label to add.

**10** Edit the **checksum** URL to use the API VIP address.

**11** Edit the **url** URL to use the API VIP address.

## 2.10.2. Creating a compute 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 compute 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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

# CHAPTER 3. MANUALLY SCALING A COMPUTE MACHINE SET

You can add or remove an instance of a machine in a compute machine set.



## NOTE

If you need to modify aspects of a compute machine set outside of scaling, see [Modifying a compute machine set](#).

## 3.1. PREREQUISITES

- If you enabled the cluster-wide proxy and scale up compute machines not included in `networking.machineNetwork[].cidr` from the installation configuration, you must [add the compute machines to the Proxy object's noProxy field](#) to prevent connection issues.



## IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

## 3.2. SCALING A COMPUTE MACHINE SET MANUALLY

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

This guidance is relevant to fully automated, installer-provisioned infrastructure installations. Customized, user-provisioned infrastructure installations do not have compute 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 compute machine sets that are in the cluster by running the following command:

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

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

2. View the compute machines that are in the cluster by running the following command:

```
$ oc get machines.machine.openshift.io -n openshift-machine-api
```

3. Set the annotation on the compute machine that you want to delete by running the following command:

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

4. Scale the compute machine set by running one of the following commands:

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

Or:

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

### TIP

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

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

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



### 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. If the drain operation fails, the machine controller cannot proceed removing the machine.

You can skip draining the node by annotating **machine.openshift.io/exclude-node-draining** in a specific machine.

### Verification

- Verify the deletion of the intended machine by running the following command:

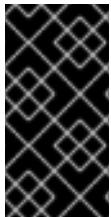
```
$ oc get machines.machine.openshift.io
```

## 3.3. THE COMPUTE MACHINE SET DELETION POLICY

**Random**, **Newest**, and **Oldest** are the three supported deletion options. The default is **Random**, meaning that random machines are chosen and deleted when scaling compute machine sets down. The deletion policy can be set according to the use case by modifying the particular compute machine set:

```
spec:  
  deletePolicy: <delete_policy>  
  replicas: <desired_replica_count>
```

Specific machines can also be prioritized for deletion by adding the annotation **machine.openshift.io/delete-machine=true** to the machine of interest, regardless of the deletion policy.



### IMPORTANT

By default, the OpenShift Container Platform router pods are deployed on workers. Because the router is required to access some cluster resources, including the web console, do not scale the worker compute machine set to **0** unless you first relocate the router pods.



### NOTE

Custom compute machine sets can be used for use cases requiring that services run on specific nodes and that those services are ignored by the controller when the worker compute machine sets are scaling down. This prevents service disruption.

## 3.4. ADDITIONAL RESOURCES

- [Lifecycle hooks for the machine deletion phase](#)

# CHAPTER 4. MODIFYING A COMPUTE MACHINE SET

You can modify a compute machine set, such as adding labels, changing the instance type, or changing block storage.



## NOTE

If you need to scale a compute machine set without making other changes, see [Manually scaling a compute machine set](#).

## 4.1. MODIFYING A COMPUTE MACHINE SET BY USING THE CLI

You can modify the configuration of a compute machine set, and then propagate the changes to the machines in your cluster by using the CLI.

By updating the compute machine set configuration, you can enable features or change the properties of the machines it creates. When you modify a compute machine set, your changes only apply to compute machines that are created after you save the updated **MachineSet** custom resource (CR). The changes do not affect existing machines.



## NOTE

Changes made in the underlying cloud provider are not reflected in the **Machine** or **MachineSet** CRs. To adjust instance configuration in cluster-managed infrastructure, use the cluster-side resources.

You can replace the existing machines with new ones that reflect the updated configuration by scaling the compute machine set to create twice the number of replicas and then scaling it down to the original number of replicas.

If you need to scale a compute machine set without making other changes, you do not need to delete the machines.



## NOTE

By default, the OpenShift Container Platform router pods are deployed on compute machines. Because the router is required to access some cluster resources, including the web console, do not scale the compute machine set to **0** unless you first relocate the router pods.

The output examples in this procedure use the values for an AWS cluster.

### Prerequisites

- Your OpenShift Container Platform cluster uses the Machine API.
- You are logged in to the cluster as an administrator by using the OpenShift CLI (**oc**).

### Procedure

1. List the compute machine sets in your cluster by running the following command:

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

## Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
<compute_machine_set_name_1>	1	1	1	1	55m
<compute_machine_set_name_2>	1	1	1	1	55m

2. Edit a compute machine set by running the following command:

```
$ oc edit machinesets.machine.openshift.io <machine_set_name> \
-n openshift-machine-api
```

3. Note the value of the **spec.replicas** field, because you need it when scaling the machine set to apply the changes.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name>
  namespace: openshift-machine-api
spec:
  replicas: 2 1
# ...
```

- 1** The examples in this procedure show a compute machine set that has a **replicas** value of **2**.

4. Update the compute machine set CR with the configuration options that you want and save your changes.
5. List the machines that are managed by the updated compute machine set by running the following command:

```
$ oc get machines.machine.openshift.io \
-n openshift-machine-api \
-l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

## Example output for an AWS cluster

NAME	PHASE	TYPE	REGION	ZONE	AGE
<machine_name_original_1>	Running	m6i.xlarge	us-west-1	us-west-1a	4h
<machine_name_original_2>	Running	m6i.xlarge	us-west-1	us-west-1a	4h

6. For each machine that is managed by the updated compute machine set, set the **delete** annotation by running the following command:

```
$ oc annotate machine.machine.openshift.io/<machine_name_original_1> \
-n openshift-machine-api \
machine.openshift.io/delete-machine="true"
```

7. To create replacement machines with the new configuration, scale the compute machine set to twice the number of replicas by running the following command:

```
$ oc scale --replicas=4 \①
machineset.machine.openshift.io <machine_set_name> \
-n openshift-machine-api
```

- ① The original example value of **2** is doubled to **4**.

8. List the machines that are managed by the updated compute machine set by running the following command:

```
$ oc get machines.machine.openshift.io \
-n openshift-machine-api \
-l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

#### Example output for an AWS cluster

NAME	PHASE	TYPE	REGION	ZONE	AGE
<machine_name_original_1>	Running	m6i.xlarge	us-west-1	us-west-1a	4h
<machine_name_original_2>	Running	m6i.xlarge	us-west-1	us-west-1a	4h
<machine_name_updated_1>	Provisioned	m6i.xlarge	us-west-1	us-west-1a	55s
<machine_name_updated_2>	Provisioning	m6i.xlarge	us-west-1	us-west-1a	55s

When the new machines are in the **Running** phase, you can scale the compute machine set to the original number of replicas.

9. To remove the machines that were created with the old configuration, scale the compute machine set to the original number of replicas by running the following command:

```
$ oc scale --replicas=2 \①
machineset.machine.openshift.io <machine_set_name> \
-n openshift-machine-api
```

- ① The original example value of **2**.

#### Verification

- To verify that a machine created by the updated machine set has the correct configuration, examine the relevant fields in the CR for one of the new machines by running the following command:

```
$ oc describe machine.machine.openshift.io <machine_name_updated_1> \
-n openshift-machine-api
```

- To verify that the compute machines without the updated configuration are deleted, list the machines that are managed by the updated compute machine set by running the following command:

```
$ oc get machines.machine.openshift.io \
-n openshift-machine-api \
-l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

#### Example output while deletion is in progress for an AWS cluster

NAME	PHASE	TYPE	REGION	ZONE	AGE
<machine_name_original_1>	Deleting	m6i.xlarge	us-west-1	us-west-1a	4h
<machine_name_original_2>	Deleting	m6i.xlarge	us-west-1	us-west-1a	4h
<machine_name_updated_1>	Running	m6i.xlarge	us-west-1	us-west-1a	5m41s
<machine_name_updated_2>	Running	m6i.xlarge	us-west-1	us-west-1a	5m41s

Example output when deletion is complete for an AWS cluster

NAME	PHASE	TYPE	REGION	ZONE	AGE
<machine_name_updated_1>	Running	m6i.xlarge	us-west-1	us-west-1a	6m30s
<machine_name_updated_2>	Running	m6i.xlarge	us-west-1	us-west-1a	6m30s

## Additional resources

- [Lifecycle hooks for the machine deletion phase](#)
- [Scaling a compute machine set manually](#)
- [Controlling pod placement using the scheduler](#)

# CHAPTER 5. MACHINE PHASES AND LIFECYCLE

Machines move through a *lifecycle* that has several defined phases. Understanding the machine lifecycle and its phases can help you verify whether a procedure is complete or troubleshoot undesired behavior. In OpenShift Container Platform, the machine lifecycle is consistent across all supported cloud providers.

## 5.1. MACHINE PHASES

As a machine moves through its lifecycle, it passes through different phases. Each phase is a basic representation of the state of the machine.

### Provisioning

There is a request to provision a new machine. The machine does not yet exist and does not have an instance, a provider ID, or an address.

### Provisioned

The machine exists and has a provider ID or an address. The cloud provider has created an instance for the machine. The machine has not yet become a node and the **status.nodeRef** section of the machine object is not yet populated.

### Running

The machine exists and has a provider ID or address. Ignition has run successfully and the cluster machine approver has approved a certificate signing request (CSR). The machine has become a node and the **status.nodeRef** section of the machine object contains node details.

### Deleting

There is a request to delete the machine. The machine object has a **DeletionTimestamp** field that indicates the time of the deletion request.

### Failed

There is an unrecoverable problem with the machine. This can happen, for example, if the cloud provider deletes the instance for the machine.

## 5.2. THE MACHINE LIFECYCLE

The lifecycle begins with the request to provision a machine and continues until the machine no longer exists.

The machine lifecycle proceeds in the following order. Interruptions due to errors or lifecycle hooks are not included in this overview.

1. There is a request to provision a new machine for one of the following reasons:
  - A cluster administrator scales a machine set such that it requires additional machines.
  - An autoscaling policy scales machine set such that it requires additional machines.
  - A machine that is managed by a machine set fails or is deleted and the machine set creates a replacement to maintain the required number of machines.
2. The machine enters the **Provisioning** phase.
3. The infrastructure provider creates an instance for the machine.
4. The machine has a provider ID or address and enters the **Provisioned** phase.

5. The Ignition configuration file is processed.
6. The kubelet issues a certificate signing request (CSR).
7. The cluster machine approver approves the CSR.
8. The machine becomes a node and enters the **Running** phase.
9. An existing machine is slated for deletion for one of the following reasons:
  - A user with **cluster-admin** permissions uses the **oc delete machine** command.
  - The machine gets a **machine.openshift.io/delete-machine** annotation.
  - The machine set that manages the machine marks it for deletion to reduce the replica count as part of reconciliation.
  - The cluster autoscaler identifies a node that is unnecessary to meet the deployment needs of the cluster.
  - A machine health check is configured to replace an unhealthy machine.
10. The machine enters the **Deleting** phase, in which it is marked for deletion but is still present in the API.
11. The machine controller removes the instance from the infrastructure provider.
12. The machine controller deletes the **Node** object.

## 5.3. DETERMINING THE PHASE OF A MACHINE

You can find the phase of a machine by using the OpenShift CLI (**oc**) or by using the web console. You can use this information to verify whether a procedure is complete or to troubleshoot undesired behavior.

### 5.3.1. Determining the phase of a machine by using the CLI

You can find the phase of a machine by using the OpenShift CLI (**oc**).

#### Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- You have installed the **oc** CLI.

#### Procedure

- List the machines on the cluster by running the following command:

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

#### Example output

NAME	PHASE	TYPE	REGION	ZONE	AGE
------	-------	------	--------	------	-----

```

mycluster-5kbpsp-master-0      Running m6i.xlarge us-west-1 us-west-1a 4h55m
mycluster-5kbpsp-master-1      Running m6i.xlarge us-west-1 us-west-1b 4h55m
mycluster-5kbpsp-master-2      Running m6i.xlarge us-west-1 us-west-1a 4h55m
mycluster-5kbpsp-worker-us-west-1a-fmx8t Running m6i.xlarge us-west-1 us-west-1a
4h51m
mycluster-5kbpsp-worker-us-west-1a-m889l Running m6i.xlarge us-west-1 us-west-1a
4h51m
mycluster-5kbpsp-worker-us-west-1b-c8qzm Running m6i.xlarge us-west-1 us-west-1b
4h51m

```

The **PHASE** column of the output contains the phase of each machine.

### 5.3.2. Determining the phase of a machine by using the web console

You can find the phase of a machine by using the OpenShift Container Platform web console.

#### Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.

#### Procedure

1. Log in to the web console as a user with the **cluster-admin** role.
2. Navigate to **Compute → Machines**.
3. On the **Machines** page, select the name of the machine that you want to find the phase of.
4. On the **Machine details** page, select the **YAML** tab.
5. In the YAML block, find the value of the **status.phase** field.

#### Example YAML snippet

```

apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  name: mycluster-5kbpsp-worker-us-west-1a-fmx8t
# ...
status:
  phase: Running ①

```

- ① In this example, the phase is **Running**.

## 5.4. ADDITIONAL RESOURCES

- [Lifecycle hooks for the machine deletion phase](#)

# CHAPTER 6. DELETING A MACHINE

You can delete a specific machine.

## 6.1. DELETING A SPECIFIC MACHINE

You can delete a specific machine.



### IMPORTANT

Do not delete a control plane machine unless your cluster uses a control plane machine set.

#### 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 by running the following command:

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

The command output contains a list of machines in the <clusterid>-<role>-<cloud\_region> format.

2. Identify the machine that you want to delete.
3. Delete the machine by running the following command:

```
$ 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. If the drain operation fails, the machine controller cannot proceed removing the machine.

You can skip draining the node by annotating **machine.openshift.io/exclude-node-draining** in a specific machine.

If the machine that you delete belongs to a machine set, a new machine is immediately created to satisfy the specified number of replicas.

## 6.2. LIFECYCLE HOOKS FOR THE MACHINE DELETION PHASE

Machine lifecycle hooks are points in the reconciliation lifecycle of a machine where the normal lifecycle process can be interrupted. In the machine **Deleting** phase, these interruptions provide the opportunity for components to modify the machine deletion process.

### 6.2.1. Terminology and definitions

To understand the behavior of lifecycle hooks for the machine deletion phase, you must understand the following concepts:

#### Reconciliation

Reconciliation is the process by which a controller attempts to make the real state of the cluster and the objects that it comprises match the requirements in an object specification.

#### Machine controller

The machine controller manages the reconciliation lifecycle for a machine. For machines on cloud platforms, the machine controller is the combination of an OpenShift Container Platform controller and a platform-specific actuator from the cloud provider.

In the context of machine deletion, the machine controller performs the following actions:

- Drain the node that is backed by the machine.
- Delete the machine instance from the cloud provider.
- Delete the **Node** object.

#### Lifecycle hook

A lifecycle hook is a defined point in the reconciliation lifecycle of an object where the normal lifecycle process can be interrupted. Components can use a lifecycle hook to inject changes into the process to accomplish a desired outcome.

There are two lifecycle hooks in the machine **Deleting** phase:

- **preDrain** lifecycle hooks must be resolved before the node that is backed by the machine can be drained.
- **preTerminate** lifecycle hooks must be resolved before the instance can be removed from the infrastructure provider.

#### Hook-implementing controller

A hook-implementing controller is a controller, other than the machine controller, that can interact with a lifecycle hook. A hook-implementing controller can do one or more of the following actions:

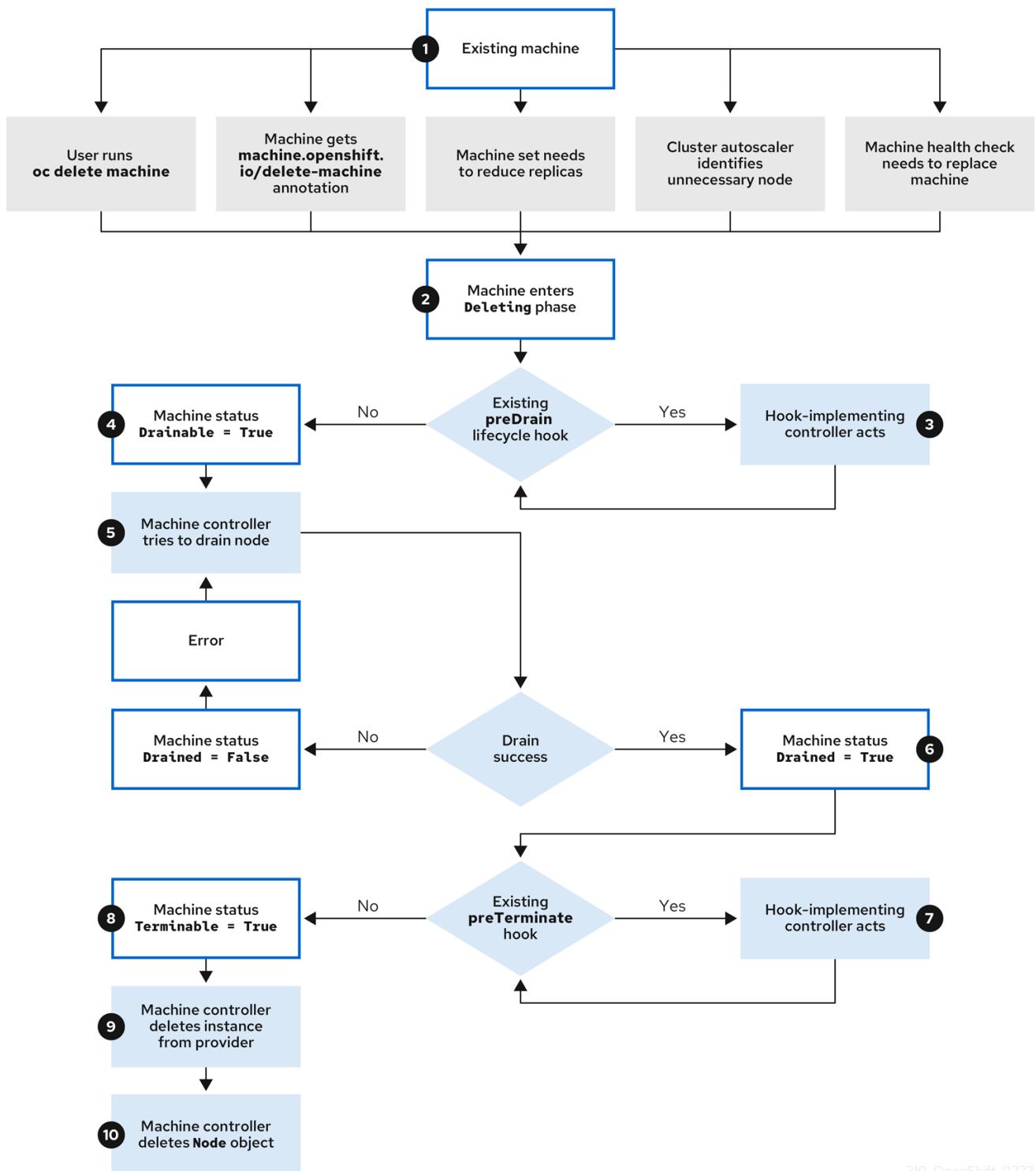
- Add a lifecycle hook.
- Respond to a lifecycle hook.
- Remove a lifecycle hook.

Each lifecycle hook has a single hook-implementing controller, but a hook-implementing controller can manage one or more hooks.

### 6.2.2. Machine deletion processing order

In OpenShift Container Platform 4.17, there are two lifecycle hooks for the machine deletion phase: **preDrain** and **preTerminate**. When all hooks for a given lifecycle point are removed, reconciliation continues as normal.

Figure 6.1. Machine deletion flow



310\_OpenShift\_0223

The machine **Deleting** phase proceeds in the following order:

1. An existing machine is slated for deletion for one of the following reasons:
  - A user with **cluster-admin** permissions uses the `oc delete machine` command.
  - The machine gets a **machine.openshift.io/delete-machine** annotation.

- The machine set that manages the machine marks it for deletion to reduce the replica count as part of reconciliation.
  - The cluster autoscaler identifies a node that is unnecessary to meet the deployment needs of the cluster.
  - A machine health check is configured to replace an unhealthy machine.
2. The machine enters the **Deleting** phase, in which it is marked for deletion but is still present in the API.
  3. If a **preDrain** lifecycle hook exists, the hook-implementing controller that manages it does a specified action.  
Until all **preDrain** lifecycle hooks are satisfied, the machine status condition **Drainable** is set to **False**.
  4. There are no unresolved **preDrain** lifecycle hooks and the machine status condition **Drainable** is set to **True**.
  5. The machine controller attempts to drain the node that is backed by the machine.
    - If draining fails, **Drained** is set to **False** and the machine controller attempts to drain the node again.
    - If draining succeeds, **Drained** is set to **True**.
  6. The machine status condition **Drained** is set to **True**.
  7. If a **preTerminate** lifecycle hook exists, the hook-implementing controller that manages it does a specified action.  
Until all **preTerminate** lifecycle hooks are satisfied, the machine status condition **Terminable** is set to **False**.
  8. There are no unresolved **preTerminate** lifecycle hooks and the machine status condition **Terminable** is set to **True**.
  9. The machine controller removes the instance from the infrastructure provider.
  10. The machine controller deletes the **Node** object.

### 6.2.3. Deletion lifecycle hook configuration

The following YAML snippets demonstrate the format and placement of deletion lifecycle hook configurations within a machine set:

#### YAML snippet demonstrating a **preDrain** lifecycle hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
...
spec:
lifecycleHooks:
preDrain:
```

```

- name: <hook_name> ①
  owner: <hook_owner> ②
...

```

- ① The name of the **preDrain** lifecycle hook.
- ② The hook-implementing controller that manages the **preDrain** lifecycle hook.

### YAML snippet demonstrating a **preTerminate** lifecycle hook

```

apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
...
spec:
  lifecycleHooks:
    preTerminate:
      - name: <hook_name> ①
        owner: <hook_owner> ②
...

```

- ① The name of the **preTerminate** lifecycle hook.
- ② The hook-implementing controller that manages the **preTerminate** lifecycle hook.

### Example lifecycle hook configuration

The following example demonstrates the implementation of multiple fictional lifecycle hooks that interrupt the machine deletion process:

### Example configuration for lifecycle hooks

```

apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
...
spec:
  lifecycleHooks:
    preDrain: ①
      - name: MigrateImportantApp
        owner: my-app-migration-controller
    preTerminate: ②
      - name: BackupFileSystem
        owner: my-backup-controller
      - name: CloudProviderSpecialCase
        owner: my-custom-storage-detach-controller ③
      - name: WaitForStorageDetach
        owner: my-custom-storage-detach-controller
...

```

- ① A **preDrain** lifecycle hook stanza that contains a single lifecycle hook.
- ② A **preTerminate** lifecycle hook stanza that contains three lifecycle hooks.

- 3 A hook-implementing controller that manages two **preTerminate** lifecycle hooks: **CloudProviderSpecialCase** and **WaitForStorageDetach**.

## 6.2.4. Machine deletion lifecycle hook examples for Operator developers

Operators can use lifecycle hooks for the machine deletion phase to modify the machine deletion process. The following examples demonstrate possible ways that an Operator can use this functionality.

### Example use cases for **preDrain** lifecycle hooks

#### Proactively replacing machines

An Operator can use a **preDrain** lifecycle hook to ensure that a replacement machine is successfully created and joined to the cluster before removing the instance of a deleted machine. This can mitigate the impact of disruptions during machine replacement or of replacement instances that do not initialize promptly.

#### Implementing custom draining logic

An Operator can use a **preDrain** lifecycle hook to replace the machine controller draining logic with a different draining controller. By replacing the draining logic, the Operator would have more flexibility and control over the lifecycle of the workloads on each node.

For example, the machine controller drain libraries do not support ordering, but a custom drain provider could provide this functionality. By using a custom drain provider, an Operator could prioritize moving mission-critical applications before draining the node to ensure that service interruptions are minimized in cases where cluster capacity is limited.

### Example use cases for **preTerminate** lifecycle hooks

#### Verifying storage detachment

An Operator can use a **preTerminate** lifecycle hook to ensure that storage that is attached to a machine is detached before the machine is removed from the infrastructure provider.

#### Improving log reliability

After a node is drained, the log exporter daemon requires some time to synchronize logs to the centralized logging system.

A logging Operator can use a **preTerminate** lifecycle hook to add a delay between when the node drains and when the machine is removed from the infrastructure provider. This delay would provide time for the Operator to ensure that the main workloads are removed and no longer adding to the log backlog. When no new data is being added to the log backlog, the log exporter can catch up on the synchronization process, thus ensuring that all application logs are captured.

## 6.2.5. Quorum protection with machine lifecycle hooks

For OpenShift Container Platform clusters that use the Machine API Operator, the etcd Operator uses lifecycle hooks for the machine deletion phase to implement a quorum protection mechanism.

By using a **preDrain** lifecycle hook, the etcd Operator can control when the pods on a control plane machine are drained and removed. To protect etcd quorum, the etcd Operator prevents the removal of an etcd member until it migrates that member onto a new node within the cluster.

This mechanism allows the etcd Operator precise control over the members of the etcd quorum and allows the Machine API Operator to safely create and remove control plane machines without specific operational knowledge of the etcd cluster.

### 6.2.5.1. Control plane deletion with quorum protection processing order

When a control plane machine is replaced on a cluster that uses a control plane machine set, the cluster temporarily has four control plane machines. When the fourth control plane node joins the cluster, the etcd Operator starts a new etcd member on the replacement node. When the etcd Operator observes that the old control plane machine is marked for deletion, it stops the etcd member on the old node and promotes the replacement etcd member to join the quorum of the cluster.

The control plane machine **Deleting** phase proceeds in the following order:

1. A control plane machine is slated for deletion.
2. The control plane machine enters the **Deleting** phase.
3. To satisfy the **preDrain** lifecycle hook, the etcd Operator takes the following actions:
  - a. The etcd Operator waits until a fourth control plane machine is added to the cluster as an etcd member. This new etcd member has a state of **Running** but not **ready** until it receives the full database update from the etcd leader.
  - b. When the new etcd member receives the full database update, the etcd Operator promotes the new etcd member to a voting member and removes the old etcd member from the cluster.

After this transition is complete, it is safe for the old etcd pod and its data to be removed, so the **preDrain** lifecycle hook is removed.

4. The control plane machine status condition **Drainable** is set to **True**.
5. The machine controller attempts to drain the node that is backed by the control plane machine.
  - If draining fails, **Drained** is set to **False** and the machine controller attempts to drain the node again.
  - If draining succeeds, **Drained** is set to **True**.
6. The control plane machine status condition **Drained** is set to **True**.
7. If no other Operators have added a **preTerminate** lifecycle hook, the control plane machine status condition **Terminable** is set to **True**.
8. The machine controller removes the instance from the infrastructure provider.
9. The machine controller deletes the **Node** object.

#### YAML snippet demonstrating the etcd quorum protection preDrain lifecycle hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain:
      - name: EtcdQuorumOperator 1
        owner: clusteroperator/etcd 2
  ...

```

- 1 The name of the **preDrain** lifecycle hook.
- 2 The hook-implementing controller that manages the **preDrain** lifecycle hook.

## 6.3. ADDITIONAL RESOURCES

- [Machine phases and lifecycle](#)
- [Replacing an unhealthy etcd member](#)
- [Managing control plane machines with control plane machine sets](#)

# CHAPTER 7. APPLYING AUTOSCALING TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

Applying autoscaling to an OpenShift Container Platform cluster involves deploying a cluster autoscaler and then deploying machine autoscalers for each machine type in your cluster.



## IMPORTANT

You can configure the cluster autoscaler only in clusters where the Machine API Operator is operational.

## 7.1. ABOUT THE CLUSTER AUTOSCALER

The cluster autoscaler adjusts the size of an OpenShift Container Platform cluster to meet its current deployment needs. It uses declarative, Kubernetes-style arguments to provide infrastructure management that does not rely on objects of a specific cloud provider. The cluster autoscaler has a cluster scope, and is not associated with a particular namespace.

The cluster autoscaler increases the size of the cluster when there are pods that fail to schedule on any of the current worker nodes due to insufficient resources or when another node is necessary to meet deployment needs. The cluster autoscaler does not increase the cluster resources beyond the limits that you specify.

The cluster autoscaler computes the total memory, CPU, and GPU on all nodes the cluster, even though it does not manage the control plane nodes. These values are not single-machine oriented. They are an aggregation of all the resources in the entire cluster. For example, if you set the maximum memory resource limit, the cluster autoscaler includes all the nodes in the cluster when calculating the current memory usage. That calculation is then used to determine if the cluster autoscaler has the capacity to add more worker resources.



## IMPORTANT

Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** resource definition that you create is large enough to account for the total possible number of machines in your cluster. This value must encompass the number of control plane machines and the possible number of compute machines that you might scale to.

Every 10 seconds, the cluster autoscaler checks which nodes are unnecessary in the cluster and removes them. The cluster autoscaler considers a node for removal if the following conditions apply:

- The node utilization is less than the *node utilization level* threshold for the cluster. The node utilization level is the sum of the requested resources divided by the allocated resources for the node. If you do not specify a value in the **ClusterAutoscaler** custom resource, the cluster autoscaler uses a default value of **0.5**, which corresponds to 50% utilization.
- The cluster autoscaler can move all pods running on the node to the other nodes. The Kubernetes scheduler is responsible for scheduling pods on the nodes.
- The cluster autoscaler does not have scale down disabled annotation.

If the following types of pods are present on a node, the cluster autoscaler will not remove the node:

- Pods with restrictive pod disruption budgets (PDBs).

- Kube-system pods that do not run on the node by default.
- Kube-system pods that do not have a PDB or have a PDB that is too restrictive.
- Pods that are not backed by a controller object such as a deployment, replica set, or stateful set.
- Pods with local storage.
- Pods that cannot be moved elsewhere because of a lack of resources, incompatible node selectors or affinity, matching anti-affinity, and so on.
- Unless they also have a "**cluster-autoscaler.kubernetes.io/safe-to-evict": "true"**" annotation, pods that have a "**cluster-autoscaler.kubernetes.io/safe-to-evict": "false"**" annotation.

For example, you set the maximum CPU limit to 64 cores and configure the cluster autoscaler to only create machines that have 8 cores each. If your cluster starts with 30 cores, the cluster autoscaler can add up to 4 more nodes with 32 cores, for a total of 62.

If you configure the cluster autoscaler, additional usage restrictions apply:

- Do not modify the nodes that are in autoscaled node groups directly. All nodes within the same node group have the same capacity and labels and run the same system pods.
- Specify requests for your pods.
- If you have to prevent pods from being deleted too quickly, configure appropriate PDBs.
- Confirm that your cloud provider quota is large enough to support the maximum node pools that you configure.
- Do not run additional node group autoscalers, especially the ones offered by your cloud provider.

The horizontal pod autoscaler (HPA) and the cluster autoscaler modify cluster resources in different ways. The HPA changes the deployment's or replica set's number of replicas based on the current CPU load. If the load increases, the HPA creates new replicas, regardless of the amount of resources available to the cluster. If there are not enough resources, the cluster autoscaler adds resources so that the HPA-created pods can run. If the load decreases, the HPA stops some replicas. If this action causes some nodes to be underutilized or completely empty, the cluster autoscaler deletes the unnecessary nodes.

The cluster autoscaler takes pod priorities into account. The Pod Priority and Preemption feature enables scheduling pods based on priorities if the cluster does not have enough resources, but the cluster autoscaler ensures that the cluster has resources to run all pods. To honor the intention of both features, the cluster autoscaler includes a priority cutoff function. You can use this cutoff to schedule "best-effort" pods, which do not cause the cluster autoscaler to increase resources but instead run only when spare resources are available.

Pods with priority lower than the cutoff value do not cause the cluster to scale up or prevent the cluster from scaling down. No new nodes are added to run the pods, and nodes running these pods might be deleted to free resources.

Cluster autoscaling is supported for the platforms that have machine API available on it.

### 7.1.1. Configuring the cluster autoscaler

First, deploy the cluster autoscaler to manage automatic resource scaling in your OpenShift Container Platform cluster.

**NOTE**

Because the cluster autoscaler is scoped to the entire cluster, you can make only one cluster autoscaler for the cluster.

### 7.1.1.1. Cluster autoscaler resource definition

This **ClusterAutoscaler** resource definition shows the parameters and sample values for the cluster autoscaler.

```
apiVersion: "autoscaling.openshift.io/v1"
kind: "ClusterAutoscaler"
metadata:
  name: "default"
spec:
  podPriorityThreshold: -10 1
  resourceLimits:
    maxNodesTotal: 24 2
    cores:
      min: 8 3
      max: 128 4
    memory:
      min: 4 5
      max: 256 6
    gpus:
      - type: nvidia.com/gpu 7
        min: 0 8
        max: 16 9
      - type: amd.com/gpu
        min: 0
        max: 4
    logVerbosity: 4 10
  scaleDown:
    enabled: true 12
    delayAfterAdd: 10m 13
    delayAfterDelete: 5m 14
    delayAfterFailure: 30s 15
    unneededTime: 5m 16
    utilizationThreshold: "0.4" 17
  expanders: ["Random"] 18
```

- 1** Specify the priority that a pod must exceed to cause the cluster autoscaler to deploy additional nodes. Enter a 32-bit integer value. The **podPriorityThreshold** value is compared to the value of the **PriorityClass** that you assign to each pod.
- 2** Specify the maximum number of nodes to deploy. This value is the total number of machines that are deployed in your cluster, not just the ones that the autoscaler controls. Ensure that this value is large enough to account for all of your control plane and compute machines and the total number of replicas that you specify in your **MachineAutoscaler** resources.
- 3** Specify the minimum number of cores to deploy in the cluster.
- 4** Specify the maximum number of cores to deploy in the cluster.

- 5 Specify the minimum amount of memory, in GiB, in the cluster.
- 6 Specify the maximum amount of memory, in GiB, in the cluster.
- 7 Optional: Specify the type of GPU node to deploy. Only **nvidia.com/gpu** and **amd.com/gpu** are valid types.
- 8 Specify the minimum number of GPUs to deploy in the cluster.
- 9 Specify the maximum number of GPUs to deploy in the cluster.
- 10 Specify the logging verbosity level between **0** and **10**. The following log level thresholds are provided for guidance:
  - **1:** (Default) Basic information about changes.
  - **4:** Debug-level verbosity for troubleshooting typical issues.
  - **9:** Extensive, protocol-level debugging information.If you do not specify a value, the default value of **1** is used.
- 11 In this section, you can specify the period to wait for each action by using any valid [ParseDuration](#) interval, including **ns**, **us**, **ms**, **s**, **m**, and **h**.
- 12 Specify whether the cluster autoscaler can remove unnecessary nodes.
- 13 Optional: Specify the period to wait before deleting a node after a node has recently been *added*. If you do not specify a value, the default value of **10m** is used.
- 14 Optional: Specify the period to wait before deleting a node after a node has recently been *deleted*. If you do not specify a value, the default value of **0s** is used.
- 15 Optional: Specify the period to wait before deleting a node after a scale down failure occurred. If you do not specify a value, the default value of **3m** is used.
- 16 Optional: Specify a period of time before an unnecessary node is eligible for deletion. If you do not specify a value, the default value of **10m** is used.
- 17 Optional: Specify the *node utilization level*. Nodes below this utilization level are eligible for deletion.

The node utilization level is the sum of the requested resources divided by the allocated resources for the node, and must be a value greater than "**0**" but less than "**1**". If you do not specify a value, the cluster autoscaler uses a default value of "**0.5**", which corresponds to 50% utilization. You must express this value as a string.
- 18 Optional: Specify any expanders that you want the cluster autoscaler to use. The following values are valid:
  - **LeastWaste:** Selects the machine set that minimizes the idle CPU after scaling. If multiple machine sets would yield the same amount of idle CPU, the selection minimizes unused memory.
  - **Priority:** Selects the machine set with the highest user-assigned priority. To use this expander, you must create a config map that defines the priority of your machine sets. For more information, see "Configuring a priority expander for the cluster autoscaler."

- **Random:** (Default) Selects the machine set randomly.

If you do not specify a value, the default value of **Random** is used.

You can specify multiple expanders by using the **[LeastWaste, Priority]** format. The cluster autoscaler applies each expander according to the specified order.

In the **[LeastWaste, Priority]** example, the cluster autoscaler first evaluates according to the **LeastWaste** criteria. If more than one machine set satisfies the **LeastWaste** criteria equally well, the cluster autoscaler then evaluates according to the **Priority** criteria. If more than one machine set satisfies all of the specified expanders equally well, the cluster autoscaler selects one to use at random.



### NOTE

When performing a scaling operation, the cluster autoscaler remains within the ranges set in the **ClusterAutoscaler** resource definition, such as the minimum and maximum number of cores to deploy or the amount of memory in the cluster. However, the cluster autoscaler does not correct the current values in your cluster to be within those ranges.

The minimum and maximum CPUs, memory, and GPU values are determined by calculating those resources on all nodes in the cluster, even if the cluster autoscaler does not manage the nodes. For example, the control plane nodes are considered in the total memory in the cluster, even though the cluster autoscaler does not manage the control plane nodes.

#### 7.1.1.2. Configuring a priority expander for the cluster autoscaler

When the cluster autoscaler uses the priority expander, it scales up by using the machine set with the highest user-assigned priority. To use this expander, you must create a config map that defines the priority of your machine sets.

For each specified priority level, you must create regular expressions to identify machine sets that you want to use when prioritizing a machine set for selection. The regular expressions must match the name of any compute machine set that you want the cluster autoscaler to consider for selection.

#### Prerequisites

- You have deployed an OpenShift Container Platform cluster that uses the Machine API.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

#### Procedure

1. List the compute machine sets on your cluster by running the following command:

```
$ oc get machinesets.machine.openshift.io
```

#### Example output

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
archive-agl030519-vplxk-worker-us-east-1c	1	1	1	1	25m

fast-01-agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
fast-02-agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
fast-03-agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
fast-04-agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
prod-01-agl030519-vplxk-worker-us-east-1a	1	1	1	1	33m
prod-02-agl030519-vplxk-worker-us-east-1c	1	1	1	1	33m

- Using regular expressions, construct one or more patterns that match the name of any compute machine set that you want to set a priority level for.  
For example, use the regular expression pattern **\*fast\*** to match any compute machine set that includes the string **fast** in its name.
- Create a **cluster-autoscaler-priority-expander.yml** YAML file that defines a config map similar to the following:

### Example priority expander config map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander 1
  namespace: openshift-machine-api 2
data:
  priorities: |- 3
    10:
      - .*fast.*
      - .*archive.*
    40:
      - .*prod.*
```

- 1** You must name config map **cluster-autoscaler-priority-expander**.
- 2** You must create the config map in the same namespace as cluster autoscaler pod, which is the **openshift-machine-api** namespace.
- 3** Define the priority of your machine sets.

The **priorities** values must be positive integers. The cluster autoscaler uses higher-value priorities before lower-value priorities.

For each priority level, specify the regular expressions that correspond to the machine sets you want to use.

- Create the config map by running the following command:

```
$ oc create configmap cluster-autoscaler-priority-expander \
--from-file=<location_of_config_map_file>/cluster-autoscaler-priority-expander.yml
```

### Verification

- Review the config map by running the following command:

```
$ oc get configmaps cluster-autoscaler-priority-expander -o yaml
```

## Next steps

- To use the priority expander, ensure that the **ClusterAutoscaler** resource definition is configured to use the **expanders: ["Priority"]** parameter.

### 7.1.2. Deploying a cluster autoscaler

To deploy a cluster autoscaler, you create an instance of the **ClusterAutoscaler** resource.

#### Procedure

1. Create a YAML file for a **ClusterAutoscaler** resource that contains the custom resource definition.
2. Create the custom resource in the cluster by running the following command:

```
$ oc create -f <filename>.yaml ①
```

① **<filename>** is the name of the custom resource file.

#### Next steps

- After you configure the cluster autoscaler, you must [configure at least one machine autoscaler](#).

## 7.2. ABOUT THE MACHINE AUTOSCALER

The machine autoscaler adjusts the number of Machines in the compute machine sets that you deploy in an OpenShift Container Platform cluster. You can scale both the default **worker** compute machine set and any other compute machine sets that you create. The machine autoscaler makes more Machines when the cluster runs out of resources to support more deployments. Any changes to the values in **MachineAutoscaler** resources, such as the minimum or maximum number of instances, are immediately applied to the compute machine set they target.



#### IMPORTANT

You must deploy a machine autoscaler for the cluster autoscaler to scale your machines. The cluster autoscaler uses the annotations on compute machine sets that the machine autoscaler sets to determine the resources that it can scale. If you define a cluster autoscaler without also defining machine autoscalers, the cluster autoscaler will never scale your cluster.

### 7.2.1. Configuring machine autoscalers

After you deploy the cluster autoscaler, deploy **MachineAutoscaler** resources that reference the compute machine sets that are used to scale the cluster.



#### IMPORTANT

You must deploy at least one **MachineAutoscaler** resource after you deploy the **ClusterAutoscaler** resource.



## NOTE

You must configure separate resources for each compute machine set. Remember that compute machine sets are different in each region, so consider whether you want to enable machine scaling in multiple regions. The compute machine set that you scale must have at least one machine in it.

### 7.2.1.1. Machine autoscaler resource definition

This **MachineAutoscaler** resource definition shows the parameters and sample values for the machine autoscaler.

```
apiVersion: "autoscaling.openshift.io/v1beta1"
kind: "MachineAutoscaler"
metadata:
  name: "worker-us-east-1a" ①
  namespace: "openshift-machine-api"
spec:
  minReplicas: 1 ②
  maxReplicas: 12 ③
  scaleTargetRef: ④
    apiVersion: machine.openshift.io/v1beta1
    kind: MachineSet ⑤
    name: worker-us-east-1a ⑥
```

- ① Specify the machine autoscaler name. To make it easier to identify which compute machine set this machine autoscaler scales, specify or include the name of the compute machine set to scale. The compute machine set name takes the following form: <clusterid>-<machineset>-<region>.
- ② Specify the minimum number machines of the specified type that must remain in the specified zone after the cluster autoscaler initiates cluster scaling. If running in AWS, GCP, Azure, RHOSP, or vSphere, this value can be set to **0**. For other providers, do not set this value to **0**.

You can save on costs by setting this value to **0** for use cases such as running expensive or limited-usage hardware that is used for specialized workloads, or by scaling a compute machine set with extra large machines. The cluster autoscaler scales the compute machine set down to zero if the machines are not in use.



## IMPORTANT

Do not set the **spec.minReplicas** value to **0** for the three compute machine sets that are created during the OpenShift Container Platform installation process for an installer provisioned infrastructure.

- ③ Specify the maximum number machines of the specified type that the cluster autoscaler can deploy in the specified zone after it initiates cluster scaling. Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** resource definition is large enough to allow the machine autoscaler to deploy this number of machines.
- ④ In this section, provide values that describe the existing compute machine set to scale.
- ⑤ The **kind** parameter value is always **MachineSet**.
- ⑥ The **name** value must match the name of an existing compute machine set, as shown in the **metadata.name** parameter value.

## 7.2.2. Deploying a machine autoscaler

To deploy a machine autoscaler, you create an instance of the **MachineAutoscaler** resource.

### Procedure

- 1 Create a YAML file for a **MachineAutoscaler** resource that contains the custom resource definition.
- 2 Create the custom resource in the cluster by running the following command:

```
$ oc create -f <filename>.yaml ①
```

① **<filename>** is the name of the custom resource file.

## 7.3. DISABLING AUTOSCALING

You can disable an individual machine autoscaler in your cluster or disable autoscaling on the cluster entirely.

### 7.3.1. Disabling a machine autoscaler

To disable a machine autoscaler, you delete the corresponding **MachineAutoscaler** custom resource (CR).



#### NOTE

Disabling a machine autoscaler does not disable the cluster autoscaler. To disable the cluster autoscaler, follow the instructions in "Disabling the cluster autoscaler".

### Procedure

- 1 List the **MachineAutoscaler** CRs for the cluster by running the following command:

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

#### Example output

NAME	REF	KIND	REF	NAME	MIN	MAX	AGE
compute-us-east-1a	MachineSet		compute-us-east-1a	1	12	39m	
compute-us-west-1a	MachineSet		compute-us-west-1a	2	4	37m	

- 2 Optional: Create a YAML file backup of the **MachineAutoscaler** CR by running the following command:

```
$ oc get MachineAutoscaler/<machine_autoscaler_name> \①
-n openshift-machine-api \
-o yaml> <machine_autoscaler_name_backup>.yaml ②
```

- 1 <**machine\_autoscaler\_name**> is the name of the CR that you want to delete.
- 2 <**machine\_autoscaler\_name\_backup**> is the name for the backup of the CR.

3. Delete the **MachineAutoscaler** CR by running the following command:

```
$ oc delete MachineAutoscaler/<machine_autoscaler_name> -n openshift-machine-api
```

#### Example output

```
machineautoscaler.autoscaling.openshift.io "compute-us-east-1a" deleted
```

#### Verification

- To verify that the machine autoscaler is disabled, run the following command:

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

The disabled machine autoscaler does not appear in the list of machine autoscalers.

#### Next steps

- If you need to re-enable the machine autoscaler, use the <**machine\_autoscaler\_name\_backup.yaml**> backup file and follow the instructions in "Deploying a machine autoscaler".

#### Additional resources

- [Disabling the cluster autoscaler](#)
- [Deploying a machine autoscaler](#)

### 7.3.2. Disabling the cluster autoscaler

To disable the cluster autoscaler, you delete the corresponding **ClusterAutoscaler** resource.



#### NOTE

Disabling the cluster autoscaler disables autoscaling on the cluster, even if the cluster has existing machine autoscalers.

#### Procedure

1. List the **ClusterAutoscaler** resource for the cluster by running the following command:

```
$ oc get ClusterAutoscaler
```

#### Example output

NAME	AGE
default	42m

2. Optional: Create a YAML file backup of the **ClusterAutoscaler** CR by running the following command:

```
$ oc get ClusterAutoscaler/default \ ①  
-o yaml> <cluster_autoscaler_backup_name>.yaml ②
```

① **default** is the name of the **ClusterAutoscaler** CR.

② **<cluster\_autoscaler\_backup\_name>** is the name for the backup of the CR.

3. Delete the **ClusterAutoscaler** CR by running the following command:

```
$ oc delete ClusterAutoscaler/default
```

#### Example output

```
clusterautoscaler.autoscaling.openshift.io "default" deleted
```

#### Verification

- To verify that the cluster autoscaler is disabled, run the following command:

```
$ oc get ClusterAutoscaler
```

#### Expected output

```
No resources found
```

#### Next steps

- Disabling the cluster autoscaler by deleting the **ClusterAutoscaler** CR prevents the cluster from autoscaling but does not delete any existing machine autoscalers on the cluster. To clean up unneeded machine autoscalers, see "Disabling a machine autoscaler".
- If you need to re-enable the cluster autoscaler, use the **<cluster\_autoscaler\_name\_backup>.yaml** backup file and follow the instructions in "Deploying a cluster autoscaler".

#### Additional resources

- [Disabling the machine autoscaler](#)
- [Deploying a cluster autoscaler](#)

## 7.4. ADDITIONAL RESOURCES

- [Including pod priority in pod scheduling decisions in OpenShift Container Platform](#)

# CHAPTER 8. CREATING INFRASTRUCTURE MACHINE SETS



## IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

You can use infrastructure machine sets to create machines that host only infrastructure components, such as the default router, the integrated container image registry, and the components for cluster metrics and monitoring. These infrastructure machines are not counted toward the total number of subscriptions that are required to run the environment.

In a production deployment, it is recommended that you deploy at least three machine sets to hold infrastructure components. Red Hat OpenShift Service Mesh deploys Elasticsearch, which requires three instances to be installed on different nodes. Each of these nodes can be deployed to different availability zones for high availability. This configuration requires three different machine sets, one for each availability zone. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability.

## 8.1. OPENSHIFT CONTAINER PLATFORM INFRASTRUCTURE COMPONENTS

Each self-managed Red Hat OpenShift subscription includes entitlements for OpenShift Container Platform and other OpenShift-related components. These entitlements are included for running OpenShift Container Platform control plane and infrastructure workloads and do not need to be accounted for during sizing.

To qualify as an infrastructure node and use the included entitlement, only components that are supporting the cluster, and not part of an end-user application, can run on those instances. Examples include the following components:

- Kubernetes and OpenShift Container Platform control plane services
- The default router
- The integrated container image registry
- The HAProxy-based Ingress Controller
- The cluster metrics collection, or monitoring service, including components for monitoring user-defined projects
- Cluster aggregated logging

- Red Hat Quay
- Red Hat OpenShift Data Foundation
- Red Hat Advanced Cluster Management for Kubernetes
- Red Hat Advanced Cluster Security for Kubernetes
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines
- Red Hat OpenShift Service Mesh

Any node that runs any other container, pod, or component is a worker node that your subscription must cover.

For information about infrastructure nodes and which components can run on infrastructure nodes, see the "Red Hat OpenShift control plane and infrastructure nodes" section in the [OpenShift sizing and subscription guide for enterprise Kubernetes](#) document.

To create an infrastructure node, you can [use a machine set](#), [label the node](#), or [use a machine config pool](#).

## 8.2. CREATING INFRASTRUCTURE MACHINE SETS FOR PRODUCTION ENVIRONMENTS

In a production deployment, it is recommended that you deploy at least three compute machine sets to hold infrastructure components. Red Hat OpenShift Service Mesh deploys Elasticsearch, which requires three instances to be installed on different nodes. Each of these nodes can be deployed to different availability zones for high availability. A configuration like this requires three different compute machine sets, one for each availability zone. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability.

### 8.2.1. Creating infrastructure machine sets for different clouds

Use the sample compute machine set for your cloud.

#### 8.2.1.1. Sample YAML for a compute machine set custom resource on AWS

The sample YAML defines a compute machine set that runs in the **us-east-1a** Amazon Web Services (AWS) Local Zone and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<infra>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-infra-<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>-infra-<zone> 4
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machine-role: infra 6
      machine.openshift.io/cluster-api-machine-type: infra 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 8
spec:
  metadata:
    labels:
      node-role.kubernetes.io/infra: "" 9
  providerSpec:
    value:
      ami:
        id: ami-046fe691f52a953f9 10
      apiVersion: machine.openshift.io/v1beta1
      blockDevices:
        - ebs:
            iops: 0
            volumeSize: 120
            volumeType: gp2
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile 11
      instanceType: m6i.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: <zone> 12
        region: <region> 13
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg 14
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-<zone> 15
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> 16
          value: owned
        - name: <custom_tag_name> 17
          value: <custom_tag_value> 18
      userDataSecret:
        name: worker-user-data

```

taints: 19

- key: node-role.kubernetes.io/infra
- effect: NoSchedule

1 3 5 11 14 16 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

2 4 8 Specify the infrastructure ID, **infra** role node label, and zone.

6 7 9 Specify the **infra** role node label.

10 Specify a valid Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) for your AWS zone for your OpenShift Container Platform nodes. If you want to use an AWS Marketplace image, you must complete the OpenShift Container Platform subscription from the [AWS Marketplace](#) to obtain an AMI ID for your region.

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}' \
get machineset/<infrastructure_id>-<role>-<zone>
```

17 18 Optional: Specify custom tag data for your cluster. For example, you might add an admin contact email address by specifying a **name:value** pair of **Email:admin-email@example.com**.



#### NOTE

Custom tags can also be specified during installation in the **install-config.yml** file. If the **install-config.yml** file and the machine set include a tag with the same **name** data, the value for the tag from the machine set takes priority over the value for the tag in the **install-config.yml** file.

12 Specify the zone, for example, **us-east-1a**.

13 Specify the region, for example, **us-east-1**.

15 Specify the infrastructure ID and zone.

19 Specify a taint to prevent user workloads from being scheduled on **infra** nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

Machine sets running on AWS support non-guaranteed [Spot Instances](#). You can save on costs by using Spot Instances at a lower price compared to On-Demand Instances on AWS. [Configure Spot Instances](#) by adding **spotMarketOptions** to the **MachineSet** YAML file.

#### 8.2.1.2. Sample YAML for a compute machine set custom resource on Azure

This sample YAML defines a compute machine set that runs in the **1** Microsoft Azure zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **infra** is the node label to add.

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: infra 2
    machine.openshift.io/cluster-api-machine-type: infra
  name: <infrastructure_id>-infra-<region> 3
  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>-infra-<region>
  template:
    metadata:
      creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: infra
      machine.openshift.io/cluster-api-machine-type: infra
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region>
    spec:
      metadata:
        creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-machineset: <machineset_name>
        node-role.kubernetes.io/infra: ""
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image: 4
          offer: ""
          publisher: ""
          resourceId: /resourceGroups/<infrastructure_id>-
            rg/providers/Microsoft.Compute/galleries/gallery_<infrastructure_id>/images/<infrastructure_id>-
              gen2/versions/latest 5
            sku: ""
            version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: <region> 6
        managedIdentity: <infrastructure_id>-identity
        metadata:
          creationTimestamp: null

```

```

natRule: null
networkResourceGroup: ""
osDisk:
  diskSizeGB: 128
  managedDisk:
    storageAccountType: Premium_LRS
  osType: Linux
publicIP: false
publicLoadBalancer: ""
resourceGroup: <infrastructure_id>-rg
sshPrivateKey: ""
sshPublicKey: ""
tags:
  - name: <custom_tag_name> 7
    value: <custom_tag_value>
subnet: <infrastructure_id>-<role>-subnet
userDataSecret:
  name: worker-user-data
vmSize: Standard_D4s_v3
vnet: <infrastructure_id>-vnet
zone: "1" 8
taints: 9
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule

```

- 1 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

You can obtain the subnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.subnet}' \
get machineset/<infrastructure_id>-worker-centralus1
```

You can obtain the vnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.vnet}' \
get machineset/<infrastructure_id>-worker-centralus1
```

- 2 Specify the **infra** node label.
- 3 Specify the infrastructure ID, **infra** node label, and region.
- 4 Specify the image details for your compute machine set. If you want to use an Azure Marketplace image, see "Selecting an Azure Marketplace image".
- 5 Specify an image that is compatible with your instance type. The Hyper-V generation V2 images created by the installation program have a **-gen2** suffix, while V1 images have the same name without the suffix.

- 6 Specify the region to place machines on.
- 7 Optional: Specify custom tags in your machine set. Provide the tag name in **<custom\_tag\_name>** field and the corresponding tag value in **<custom\_tag\_value>** field.
- 8 Specify the zone within your region to place machines on. Be sure that your region supports the zone that you specify.
- 9 Specify a taint to prevent user workloads from being scheduled on infra nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

Machine sets running on Azure support non-guaranteed [Spot VMs](#). You can save on costs by using Spot VMs at a lower price compared to standard VMs on Azure. You can [configure Spot VMs](#) by adding **spotVMOptions** to the **MachineSet** YAML file.

#### Additional resources

- [Selecting an Azure Marketplace image](#)

#### 8.2.1.3. Sample YAML for a compute machine set custom resource on Azure Stack Hub

This sample YAML defines a compute machine set that runs in the **1** Microsoft Azure zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<infra>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 6
  template:
    metadata:
      creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machine-role: <infra> 8
```

```

machine.openshift.io/cluster-api-machine-type: <infra> 9
machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 10
spec:
metadata:
creationTimestamp: null
labels:
node-role.kubernetes.io/infra: "" 11
taints: 12
- key: node-role.kubernetes.io/infra
effect: NoSchedule
providerSpec:
value:
apiVersion: machine.openshift.io/v1beta1
availabilitySet: <availability_set> 13
credentialsSecret:
name: azure-cloud-credentials
namespace: openshift-machine-api
image:
offer: ""
publisher: ""
resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 14
sku: ""
version: ""
internalLoadBalancer: ""
kind: AzureMachineProviderSpec
location: <region> 15
managedIdentity: <infrastructure_id>-identity 16
metadata:
creationTimestamp: null
natRule: null
networkResourceGroup: ""
osDisk:
diskSizeGB: 128
managedDisk:
storageAccountType: Premium_LRS
osType: Linux
publicIP: false
publicLoadBalancer: ""
resourceGroup: <infrastructure_id>-rg 17
sshPrivateKey: ""
sshPublicKey: ""
subnet: <infrastructure_id>-<role>-subnet 18 19
userDataSecret:
name: worker-user-data 20
vmSize: Standard_DS4_v2
vnet: <infrastructure_id>-vnet 21
zone: "1" 22

```

**1** **5** **7** **14** **16** **17** **18** **21** Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

You can obtain the subnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.subnet}{"\n"}' \
get machineset/<infrastructure_id>-worker-centralus1
```

You can obtain the vnet by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.vnet} {"\n"}' \
get machineset/<infrastructure_id>-worker-centralus1
```

**2 3 8 9 11 19 20** Specify the <**infra**> node label.

**4 6 10** Specify the infrastructure ID, <**infra**> node label, and region.

**12** Specify a taint to prevent user workloads from being scheduled on infra nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

**15** Specify the region to place machines on.

**13** Specify the availability set for the cluster.

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



#### NOTE

Machine sets running on Azure Stack Hub do not support non-guaranteed Spot VMs.

#### 8.2.1.4. Sample YAML for a compute machine set custom resource on IBM Cloud

This sample YAML defines a compute machine set that runs in a specified IBM Cloud® zone in a region and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, <**infrastructure\_id**> is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and <**infra**> is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-<infra>-<region> 4
  namespace: openshift-machine-api
```

```

spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<region> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<region> 10
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/infra: ""
    providerSpec:
      value:
        apiVersion: ibmcloudproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: ibmcloud-credentials
        image: <infrastructure_id>-rhcos 11
        kind: IBMCloudMachineProviderSpec
        primaryNetworkInterface:
          securityGroups:
            - <infrastructure_id>-sg-cluster-wide
            - <infrastructure_id>-sg-openshift-net
          subnet: <infrastructure_id>-subnet-compute-<zone> 12
        profile: <instance_profile> 13
        region: <region> 14
        resourceGroup: <resource_group> 15
        userDataSecret:
          name: <role>-user-data 16
        vpc: <vpc_name> 17
        zone: <zone> 18
      taints:
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule

```

**1 5 7** The infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

**2 3 8 9 16** The **<infra>** node label.

**4 6 10** The infrastructure ID, **<infra>** node label, and region.

**11** The custom Red Hat Enterprise Linux CoreOS (RHCOS) image that was used for cluster installation.

- 12 The infrastructure ID and zone within your region to place machines on. Be sure that your region supports the zone that you specify.
- 13 Specify the [IBM Cloud® instance profile](#).
- 14 Specify the region to place machines on.
- 15 The resource group that machine resources are placed in. This is either an existing resource group specified at installation time, or an installer-created resource group named based on the infrastructure ID.
- 17 The VPC name.
- 18 Specify the zone within your region to place machines on. Be sure that your region supports the zone that you specify.
- 19 The taint to prevent user workloads from being scheduled on infra nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

### 8.2.1.5. Sample YAML for a compute machine set custom resource on GCP

This sample YAML defines a compute machine set that runs in Google Cloud Platform (GCP) and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**, where **infra** is the node label to add.

#### Values obtained by using the OpenShift CLI

In the following example, you can obtain some of the values for your cluster by using the OpenShift CLI.

#### Infrastructure ID

The **<infrastructure\_id>** string is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

#### Image path

The **<path\_to\_image>** string is the path to the image that was used to create the disk. If you have the OpenShift CLI installed, you can obtain the path to the image by running the following command:

```
$ oc -n openshift-machine-api \
-o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}{"\n"}' \
get machineset/<infrastructure_id>-worker-a
```

#### Sample GCP MachineSet values

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

```

metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ①
  name: <infrastructure_id>-w-a
  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>-w-a
template:
  metadata:
    creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <infra> ②
    machine.openshift.io/cluster-api-machine-type: <infra>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
spec:
  metadata:
    labels:
      node-role.kubernetes.io/infra: ""
  providerSpec:
    value:
      apiVersion: gcpprovider.openshift.io/v1beta1
      canIPForward: false
      credentialsSecret:
        name: gcp-cloud-credentials
      deletionProtection: false
      disks:
        - autoDelete: true
          boot: true
          image: <path_to_image> ③
          labels: null
          sizeGb: 128
          type: pd-ssd
      gcpMetadata: ④
        - key: <custom_metadata_key>
          value: <custom_metadata_value>
      kind: GCPMachineProviderSpec
      machineType: n1-standard-4
      metadata:
        creationTimestamp: null
      networkInterfaces:
        - network: <infrastructure_id>-network
          subnetwork: <infrastructure_id>-worker-subnet
      projectID: <project_name> ⑤
      region: us-central1
      serviceAccounts: ⑥
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com
          scopes:
            - https://www.googleapis.com/auth/cloud-platform
      tags:
        - <infrastructure_id>-worker

```

```

userDataSecret:
  name: worker-user-data
  zone: us-central1-a
  taints: 7
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule

```

- 1 For <infrastructure\_id>, specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.
- 2 For <infra>, specify the <infra> node label.
- 3 Specify the path to the image that is used in current compute machine sets.

To use a GCP Marketplace image, specify the offer to use:

- OpenShift Container Platform:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736>
- OpenShift Kubernetes Engine:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736>

- 4 Optional: Specify custom metadata in the form of a **key:value** pair. For example use cases, see the GCP documentation for [setting custom metadata](#).
- 5 For <project\_name>, specify the name of the GCP project that you use for your cluster.
- 6 Specifies a single service account. Multiple service accounts are not supported.
- 7 Specify a taint to prevent user workloads from being scheduled on infra nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

Machine sets running on GCP support non-guaranteed [preemptible VM instances](#). You can save on costs by using preemptible VM instances at a lower price compared to normal instances on GCP. You can [configure preemptible VM instances](#) by adding **preemptible** to the **MachineSet** YAML file.

#### 8.2.1.6. Sample YAML for a compute machine set custom resource on Nutanix

This sample YAML defines a Nutanix compute machine set that creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, <infrastructure\_id> is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and <infra> is the node label to add.

**Values obtained by using the OpenShift CLI**

In the following example, you can obtain some of the values for your cluster by using the OpenShift CLI (`oc`).

## Infrastructure ID

The `<infrastructure_id>` string is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra>
  name: <infrastructure_id>-<infra>-<zone> 3
  namespace: openshift-machine-api
  annotations: 4
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone>
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <infra>
      machine.openshift.io/cluster-api-machine-type: <infra>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone>
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/infra: ""
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1
        bootType: "" 5
        categories: 6
          - key: <category_name>
            value: <category_value>
        cluster: 7
          type: uuid
          uuid: <cluster_uuid>
        credentialsSecret:
          name: nutanix-credentials
        image:
          name: <infrastructure_id>-rhcos 8
```

```

type: name
kind: NutanixMachineProviderConfig
memorySize: 16Gi ⑨
project: ⑩
  type: name
  name: <project_name>
subnets:
- type: uuid
  uuid: <subnet_uuid>
systemDiskSize: 120Gi ⑪
userDataSecret:
  name: <user_data_secret> ⑫
vcpuSockets: 4 ⑬
  vcpusPerSocket: 1 ⑭
taints: ⑮
- key: node-role.kubernetes.io/infra
  effect: NoSchedule

```

- ① For **<infrastructure\_id>**, specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.
- ② Specify the **<infra>** node label.
- ③ Specify the infrastructure ID, **<infra>** node label, and zone.
- ④ Annotations for the cluster autoscaler.
- ⑤ Specifies the boot type that the compute machines use. For more information about boot types, see [Understanding UEFI, Secure Boot, and TPM in the Virtualized Environment](#). Valid values are **Legacy**, **SecureBoot**, or **UEFI**. The default is **Legacy**.



#### NOTE

You must use the **Legacy** boot type in OpenShift Container Platform 4.17.

- ⑥ Specify one or more Nutanix Prism categories to apply to compute machines. This stanza requires **key** and **value** parameters for a category key-value pair that exists in Prism Central. For more information about categories, see [Category management](#).
- ⑦ Specify a Nutanix Prism Element cluster configuration. In this example, the cluster type is **uuid**, so there is a **uuid** stanza.
- ⑧ Specify the image to use. Use an image from an existing default compute machine set for the cluster.
- ⑨ Specify the amount of memory for the cluster in Gi.
- ⑩ Specify the Nutanix project that you use for your cluster. In this example, the project type is **name**, so there is a **name** stanza.
- ⑪ Specify the size of the system disk in Gi.
- ⑫ Specify the name of the secret in the user data YAML file that is in the **openshift-machine-api** namespace. Use the value that installation program populates in the default compute machine set.

- 13 Specify the number of vCPU sockets.
- 14 Specify the number of vCPUs per socket.
- 15 Specify a taint to prevent user workloads from being scheduled on infra nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

#### 8.2.1.7. Sample YAML for a compute machine set custom resource on RHOSP

This sample YAML defines a compute machine set that runs on Red Hat OpenStack Platform (RHOSP) and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<infra>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 10
    spec:
      metadata:
        creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: ""
      taints: 11
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
      providerSpec:
        value:
```

```

apiVersion: machine.openshift.io/v1alpha1
cloudName: openstack
cloudsSecret:
  name: openstack-cloud-credentials
  namespace: openshift-machine-api
flavor: <nova_flavor>
image: <glance_image_name_or_location>
serverGroupID: <optional_UUID_of_server_group> 12
kind: OpenstackProviderSpec
networks: 13
- filter: {}
  subnets:
    - filter:
        name: <subnet_name>
        tags: openshiftClusterID=<infrastructure_id> 14
primarySubnet: <rhosp_subnet_UUID> 15
securityGroups:
- filter: {}
  name: <infrastructure_id>-worker 16
serverMetadata:
  Name: <infrastructure_id>-worker 17
  openshiftClusterID: <infrastructure_id> 18
tags:
- openshiftClusterID=<infrastructure_id> 19
trunk: true
userDataSecret:
  name: worker-user-data 20
availabilityZone: <optional_openstack_availability_zone>

```

1 5 7 14 16 17 18 19 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

2 3 8 9 20 Specify the <infra> node label.

4 6 10 Specify the infrastructure ID and <infra> node label.

11 Specify a taint to prevent user workloads from being scheduled on infra nodes.



### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

- 12 To set a server group policy for the MachineSet, enter the value that is returned from [creating a server group](#). For most deployments, **anti-affinity** or **soft-anti-affinity** policies are recommended.
- 13 Required for deployments to multiple networks. If deploying to multiple networks, this list must include the network that is used as the **primarySubnet** value.
- 15 Specify the RHOSP subnet that you want the endpoints of nodes to be published on. Usually, this is

the same subnet that is used as the value of **machinesSubnet** in the **install-config.yaml** file.

### 8.2.1.8. Sample YAML for a compute machine set custom resource on vSphere

This sample YAML defines a compute machine set that runs on VMware vSphere and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructure\_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<infra>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-infra 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>-infra 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <infra> 6
        machine.openshift.io/cluster-api-machine-type: <infra> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 8
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
              - networkName: "<vm_network_name>" 11
          numCPUs: 4
```

```

numCoresPerSocket: 1
snapshot: ""
template: <vm_template_name> ⑫
userDataSecret:
  name: worker-user-data
workspace:
  datacenter: <vcenter_data_center_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. If you have the OpenShift CLI (**oc**) installed, you can obtain the infrastructure ID by running the following command:

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

- ② ④ ⑧ Specify the infrastructure ID and **<infra>** node label.

- ⑥ ⑦ ⑨ Specify the **<infra>** node label.

- ⑩ Specify a taint to prevent user workloads from being scheduled on infra nodes.



#### NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or [add toleration on misscheduled DNS pods](#).

- ⑪ Specify the vSphere VM network to deploy the compute machine set to. This VM network must be where other compute machines reside in the cluster.
- ⑫ Specify the vSphere VM template to use, such as **user-5ddjd-rhcos**.
- ⑬ Specify the vCenter data center to deploy the compute machine set on.
- ⑭ Specify the vCenter datastore to deploy the compute machine set on.
- ⑮ Specify the path to the vSphere VM folder in vCenter, such as **/dc1/vm/user-inst-5ddjd**.
- ⑯ Specify the vSphere resource pool for your VMs.
- ⑰ Specify the vCenter server IP or fully qualified domain name.

### 8.2.2. Creating a compute 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 compute 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-infra-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

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

#### 8.2.3. Creating an infrastructure node



#### IMPORTANT

See [Creating infrastructure machine sets for installer-provisioned infrastructure environments](#) or for any cluster where the control plane nodes are managed by the machine API.

Requirements of the cluster dictate that infrastructure, also called **infra** nodes, be provisioned. The installer only provides provisions for control plane and worker nodes. Worker nodes can be designated as infrastructure nodes or application, also called **app**, nodes through labeling.

## Procedure

1. Add a label to the worker node that you want to act as application node:

```
$ oc label node <node-name> node-role.kubernetes.io/app=""
```

2. Add a label to the worker nodes that you want to act as infrastructure nodes:

```
$ oc label node <node-name> node-role.kubernetes.io/infra=""
```

3. Check to see if applicable nodes now have the **infra** role and **app** roles:

```
$ oc get nodes
```

4. Create a default cluster-wide node selector. The default node selector is applied to pods created in all namespaces. This creates an intersection with any existing node selectors on a pod, which additionally constrains the pod's selector.



### IMPORTANT

If the default node selector key conflicts with the key of a pod's label, then the default node selector is not applied.

However, do not set a default node selector that might cause a pod to become unschedulable. For example, setting the default node selector to a specific node role, such as `node-role.kubernetes.io/infra=""`, when a pod's label is set to a different node role, such as `node-role.kubernetes.io/master=""`, can cause the pod to become unschedulable. For this reason, use caution when setting the default node selector to specific node roles.

You can alternatively use a project node selector to avoid cluster-wide node selector key conflicts.

- a. Edit the **Scheduler** object:

```
$ oc edit scheduler cluster
```

- b. Add the **defaultNodeSelector** field with the appropriate node selector:

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
spec:
  defaultNodeSelector: node-role.kubernetes.io/infra="" 1
# ...
```

1 This example node selector deploys pods on infrastructure nodes by default.

- c. Save the file to apply the changes.

You can now move infrastructure resources to the newly labeled **infra** nodes.

## Additional resources

- [Moving resources to infrastructure machine sets](#)

### 8.2.4. Creating a machine config pool for infrastructure machines

If you need infrastructure machines to have dedicated configurations, you must create an infra pool.



#### IMPORTANT

Creating a custom machine configuration pool overrides default worker pool configurations if they refer to the same file or unit.

#### Procedure

1. Add a label to the node you want to assign as the infra node with a specific label:

```
$ oc label node <node_name> <label>
$ oc label node ci-In-n8mqwr2-f76d1-xscn2-worker-c-6fmtx node-role.kubernetes.io/infra=
```

2. Create a machine config pool that contains both the worker role and your custom role as machine config selector:

```
$ cat infra.mcp.yaml
```

#### Example output

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: infra
spec:
  machineConfigSelector:
    matchExpressions:
      - {key: machineconfiguration.openshift.io/role, operator: In, values: [worker,infra]} 1
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: "" 2
```

- 1** Add the worker role and your custom role.
- 2** Add the label you added to the node as a **nodeSelector**.

**NOTE**

Custom machine config pools inherit machine configs from the worker pool. Custom pools use any machine config targeted for the worker pool, but add the ability to also deploy changes that are targeted at only the custom pool. Because a custom pool inherits resources from the worker pool, any change to the worker pool also affects the custom pool.

3. After you have the YAML file, you can create the machine config pool:

```
$ oc create -f infra.mcp.yaml
```

4. Check the machine configs to ensure that the infrastructure configuration rendered successfully:

```
$ oc get machineconfig
```

**Example output**

NAME	GENERATED	BYCONTROLLER
IGNITIONVERSION	CREATED	
00-master	365c1cf14de5b0e3b85e0fc815b0060f36ab955	
3.2.0	31d	
00-worker	365c1cf14de5b0e3b85e0fc815b0060f36ab955	
3.2.0	31d	
01-master-container-runtime	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 31d
01-master-kubelet	365c1cf14de5b0e3b85e0fc815b0060f36ab955	
3.2.0	31d	
01-worker-container-runtime	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 31d
01-worker-kubelet	365c1cf14de5b0e3b85e0fc815b0060f36ab955	
3.2.0	31d	
99-master-1ae2a1e0-a115-11e9-8f14-005056899d54-registries	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 31d
99-master-ssh		3.2.0 31d
99-worker-1ae64748-a115-11e9-8f14-005056899d54-registries	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 31d
99-worker-ssh		3.2.0 31d
rendered-infra-4e48906dca84ee702959c71a53ee80e7	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 23m
rendered-master-072d4b2da7f88162636902b074e9e28e	5b6fb8349a29735e48446d435962dec4547d3090	3.2.0 31d
rendered-master-3e88ec72aed3886dec061df60d16d1af	02c07496ba0417b3e12b78fb32baf6293d314f79	3.2.0 31d
rendered-master-419bee7de96134963a15fdf9dd473b25	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 17d
rendered-master-53f5c91c7661708adce18739cc0f40fb	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 13d
rendered-master-a6a357ec18e5bce7f5ac426fc7c5ffcd	365c1cf14de5b0e3b85e0fc815b0060f36ab955	3.2.0 7d3h
rendered-master-dc7f874ec77fc4b969674204332da037	5b6fb8349a29735e48446d435962dec4547d3090	3.2.0 31d
rendered-worker-1a75960c52ad18ff5dfa6674eb7e533d	5b6fb8349a29735e48446d435962dec4547d3090	3.2.0 31d

```

rendered-worker-2640531be11ba43c61d72e82dc634ce6
5b6fb8349a29735e48446d435962dec4547d3090 3.2.0      31d
rendered-worker-4e48906dca84ee702959c71a53ee80e7
365c1cf14de5b0e3b85e0fc815b0060f36ab955 3.2.0      7d3h
rendered-worker-4f110718fe88e5f349987854a1147755
365c1cf14de5b0e3b85e0fc815b0060f36ab955 3.2.0      17d
rendered-worker-afc758e194d6188677eb837842d3b379
02c07496ba0417b3e12b78fb32baf6293d314f79 3.2.0      31d
rendered-worker-daa08cc1e8f5fcdeba24de60cd955cc3
365c1cf14de5b0e3b85e0fc815b0060f36ab955 3.2.0      13d

```

You should see a new machine config, with the **rendered-infra-\*** prefix.

5. Optional: To deploy changes to a custom pool, create a machine config that uses the custom pool name as the label, such as **infra**. Note that this is not required and only shown for instructional purposes. In this manner, you can apply any custom configurations specific to only your infra nodes.



### NOTE

After you create the new machine config pool, the MCO generates a new rendered config for that pool, and associated nodes of that pool reboot to apply the new configuration.

- a. Create a machine config:

```
$ cat infra.mc.yaml
```

### Example output

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 51-infra
  labels:
    machineconfiguration.openshift.io/role: infra ①
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/infratest
          mode: 0644
          contents:
            source: data:,infra

```

- ① Add the label you added to the node as a **nodeSelector**.

- b. Apply the machine config to the infra-labeled nodes:

```
$ oc create -f infra.mc.yaml
```

6. Confirm that your new machine config pool is available:

```
$ oc get mcp
```

#### Example output

	NAME	CONFIG	UPDATED	UPDATING	DEGRADED
	MACHINECOUNT	READYMACHINECOUNT	UPDATEDMACHINECOUNT		
	DEGRADED MACHINECOUNT	AGE			
infra	rendered-infra-60e35c2e99f42d976e084fa94da4d0fc	True	False	False	1
1	1	0	4m20s		
master	rendered-master-9360fdb895d4c131c7c4bebbae099c90	True	False	False	3
3	3	3	91m		
worker	rendered-worker-60e35c2e99f42d976e084fa94da4d0fc	True	False	False	2
2	2	2	91m		

In this example, a worker node was changed to an infra node.

#### Additional resources

- See [Node configuration management with machine config pools](#) for more information on grouping infra machines in a custom pool.

## 8.3. ASSIGNING MACHINE SET RESOURCES TO INFRASTRUCTURE NODES

After creating an infrastructure machine set, the **worker** and **infra** roles are applied to new infra nodes. Nodes with the **infra** role applied are not counted toward the total number of subscriptions that are required to run the environment, even when the **worker** role is also applied.

However, with an infra node being assigned as a worker, there is a chance user workloads could get inadvertently assigned to an infra node. To avoid this, you can apply a taint to the infra node and tolerations for the pods you want to control.

### 8.3.1. Binding infrastructure node workloads using taints and tolerations

If you have an infra node that has the **infra** and **worker** roles assigned, you must configure the node so that user workloads are not assigned to it.



#### IMPORTANT

It is recommended that you preserve the dual **infra,worker** label that is created for infra nodes and use taints and tolerations to manage nodes that user workloads are scheduled on. If you remove the **worker** label from the node, you must create a custom pool to manage it. A node with a label other than **master** or **worker** is not recognized by the MCO without a custom pool. Maintaining the **worker** label allows the node to be managed by the default worker machine config pool, if no custom pools that select the custom label exists. The **infra** label communicates to the cluster that it does not count toward the total number of subscriptions.

#### Prerequisites

- Configure additional **MachineSet** objects in your OpenShift Container Platform cluster.

## Procedure

1. Add a taint to the infra node to prevent scheduling user workloads on it:

- a. Determine if the node has the taint:

```
$ oc describe nodes <node_name>
```

### Sample output

```
oc describe node ci-1n-iyhx092-f76d1-nvdfm-worker-b-wln2l
Name:      ci-1n-iyhx092-f76d1-nvdfm-worker-b-wln2l
Roles:     worker
...
Taints:    node-role.kubernetes.io/infra:NoSchedule
...
```

This example shows that the node has a taint. You can proceed with adding a toleration to your pod in the next step.

- b. If you have not configured a taint to prevent scheduling user workloads on it:

```
$ oc adm taint nodes <node_name> <key>=<value>:<effect>
```

For example:

```
$ oc adm taint nodes node1 node-role.kubernetes.io/infra=reserved:NoSchedule
```

### TIP

You can alternatively apply the following YAML to add the taint:

```
kind: Node
apiVersion: v1
metadata:
  name: <node_name>
labels:
  ...
spec:
  taints:
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
      value: reserved
  ...
```

This example places a taint on **node1** that has key **node-role.kubernetes.io/infra** and taint effect **NoSchedule**. Nodes with the **NoSchedule** effect schedule only pods that tolerate the taint, but allow existing pods to remain scheduled on the node.



### NOTE

If a descheduler is used, pods violating node taints could be evicted from the cluster.

- c. Add the taint with NoExecute Effect along with the above taint with NoSchedule Effect:

```
$ oc adm taint nodes <node_name> <key>=<value>:<effect>
```

For example:

```
$ oc adm taint nodes node1 node-role.kubernetes.io/infra=reserved:NoExecute
```

### TIP

You can alternatively apply the following YAML to add the taint:

```
kind: Node
apiVersion: v1
metadata:
  name: <node_name>
  labels:
    ...
spec:
  taints:
    - key: node-role.kubernetes.io/infra
      effect: NoExecute
      value: reserved
    ...
```

This example places a taint on **node1** that has the key **node-role.kubernetes.io/infra** and taint effect **NoExecute**. Nodes with the **NoExecute** effect schedule only pods that tolerate the taint. The effect will remove any existing pods from the node that do not have a matching toleration.

2. Add tolerations for the pod configurations you want to schedule on the infra node, like router, registry, and monitoring workloads. Add the following code to the **Pod** object specification:

```
tolerations:
  - effect: NoSchedule ①
    key: node-role.kubernetes.io/infra ②
    value: reserved ③
  - effect: NoExecute ④
    key: node-role.kubernetes.io/infra ⑤
    operator: Exists ⑥
    value: reserved ⑦
```

- ① Specify the effect that you added to the node.
- ② Specify the key that you added to the node.
- ③ Specify the value of the key-value pair taint that you added to the node.
- ④ Specify the effect that you added to the node.
- ⑤ Specify the key that you added to the node.
- ⑥

Specify the **Exists** Operator to require a taint with the key `node-role.kubernetes.io/infra` to be present on the node.

- 7 Specify the value of the key-value pair taint that you added to the node.

This toleration matches the taint created by the `oc adm taint` command. A pod with this toleration can be scheduled onto the infra node.



#### NOTE

Moving pods for an Operator installed via OLM to an infra node is not always possible. The capability to move Operator pods depends on the configuration of each Operator.

3. Schedule the pod to the infra node using a scheduler. See the documentation for *Controlling pod placement onto nodes* for details.

#### Additional resources

- See [Controlling pod placement using the scheduler](#) for general information on scheduling a pod to a node.
- See [Moving resources to infrastructure machine sets](#) for instructions on scheduling pods to infra nodes.
- See [Understanding taints and tolerations](#) for more details about different effects of taints.

## 8.4. MOVING RESOURCES TO INFRASTRUCTURE MACHINE SETS

Some of the infrastructure resources are deployed in your cluster by default. You can move them to the infrastructure machine sets that you created by adding the infrastructure node selector, as shown:

```
spec:
  nodePlacement: ①
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/infra: ""
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
```

- 1 Add a **nodeSelector** parameter with the appropriate value to the component you want to move. You can use a **nodeSelector** in the format shown or use `<key>: <value>` pairs, based on the value specified for the node. If you added a taint to the infrastrucuture node, also add a matching toleration.

Applying a specific node selector to all infrastructure components causes OpenShift Container Platform to [schedule those workloads on nodes with that label](#).

### 8.4.1. Moving the router

You can deploy the router pod to a different compute machine set. By default, the pod is deployed to a worker node.

#### Prerequisites

- Configure additional compute machine sets in your OpenShift Container Platform cluster.

#### Procedure

- View the **IngressController** custom resource for the router Operator:

```
$ oc get ingresscontroller default -n openshift-ingress-operator -o yaml
```

The command output resembles the following text:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: 2019-04-18T12:35:39Z
  finalizers:
  - ingresscontroller.operator.openshift.io/finalizer-ingresscontroller
  generation: 1
  name: default
  namespace: openshift-ingress-operator
  resourceVersion: "11341"
  selfLink: /apis/operator.openshift.io/v1/namespaces/openshift-ingress-
operator/ingresscontrollers/default
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a
spec: {}
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: 2019-04-18T12:36:15Z
    status: "True"
    type: Available
  domain: apps.<cluster>.example.com
  endpointPublishingStrategy:
    type: LoadBalancerService
  selector: ingresscontroller.operator.openshift.io/deployment-ingresscontroller=default
```

- Edit the **ingresscontroller** resource and change the **nodeSelector** to use the **infra** label:

```
$ oc edit ingresscontroller default -n openshift-ingress-operator
```

```
spec:
  nodePlacement:
    nodeSelector: ①
      matchLabels:
        node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
```

```

value: reserved
- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved

```

- Add a **nodeSelector** parameter with the appropriate value to the component you want to move. You can use a **nodeSelector** in the format shown or use **<key>: <value>** pairs, based on the value specified for the node. If you added a taint to the infrastructure node, also add a matching toleration.

3. Confirm that the router pod is running on the **infra** node.

- View the list of router pods and note the node name of the running pod:

```
$ oc get pod -n openshift-ingress -o wide
```

#### Example output

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE	READINESS	GATES				
router-default-86798b4b5d-bdlvd	1/1	Running	0	28s	10.130.2.4	ip-10-0-217-226.ec2.internal
router-default-955d875f4-255g8	0/1	Terminating	0	19h	10.129.2.4	ip-10-0-148-172.ec2.internal

In this example, the running pod is on the **ip-10-0-217-226.ec2.internal** node.

- View the node status of the running pod:

```
$ oc get node <node_name> ①
```

- Specify the **<node\_name>** that you obtained from the pod list.

#### Example output

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-217-226.ec2.internal	Ready	infra,worker	17h	v1.30.3

Because the role list includes **infra**, the pod is running on the correct node.

### 8.4.2. Moving the default registry

You configure the registry Operator to deploy its pods to different nodes.

#### Prerequisites

- Configure additional compute machine sets in your OpenShift Container Platform cluster.

#### Procedure

- View the **config/instance** object:

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

### Example output

```
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  creationTimestamp: 2019-02-05T13:52:05Z
  finalizers:
  - imageregistry.operator.openshift.io/finalizer
  generation: 1
  name: cluster
  resourceVersion: "56174"
  selfLink: /apis/imageregistry.operator.openshift.io/v1/configs/cluster
  uid: 36fd3724-294d-11e9-a524-12ffeee2931b
spec:
  httpSecret: d9a012ccd117b1e6616ceccb2c3bb66a5fed1b5e481623
  logging: 2
  managementState: Managed
  proxy: {}
  replicas: 1
  requests:
    read: {}
    write: {}
  storage:
    s3:
      bucket: image-registry-us-east-1-c92e88cad85b48ec8b312344dff03c82-392c
      region: us-east-1
  status:
...
...
```

2. Edit the **config/instance** object:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - podAffinityTerm:
          namespaces:
          - openshift-image-registry
          topologyKey: kubernetes.io/hostname
          weight: 100
  logLevel: Normal
  managementState: Managed
  nodeSelector: 1
    node-role.kubernetes.io/infra: ""
  tolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
    value: reserved
```

```

- effect: NoExecute
key: node-role.kubernetes.io/infra
value: reserved

```

- Add a **nodeSelector** parameter with the appropriate value to the component you want to move. You can use a **nodeSelector** in the format shown or use **<key>: <value>** pairs, based on the value specified for the node. If you added a taint to the infrastructure node, also add a matching toleration.

- Verify the registry pod has been moved to the infrastructure node.

- Run the following command to identify the node where the registry pod is located:

```
$ oc get pods -o wide -n openshift-image-registry
```

- Confirm the node has the label you specified:

```
$ oc describe node <node_name>
```

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

#### 8.4.3. Moving the monitoring solution

The monitoring stack includes multiple components, including Prometheus, Thanos Querier, and Alertmanager. The Cluster Monitoring Operator manages this stack. To redeploy the monitoring stack to infrastructure nodes, you can create and apply a custom config map.

##### Procedure

- Edit the **cluster-monitoring-config** config map and change the **nodeSelector** to use the **infra** label:

```
$ oc edit configmap cluster-monitoring-config -n openshift-monitoring
```

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector: ①
        node-role.kubernetes.io/infra: ""
      tolerations:
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoSchedule
        - key: node-role.kubernetes.io/infra
          value: reserved
          effect: NoExecute
  prometheusK8s:

```

```
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoSchedule
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoExecute
prometheusOperator:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoSchedule
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoExecute
metricsServer:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoSchedule
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoExecute
kubeStateMetrics:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoSchedule
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoExecute
telemeterClient:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoSchedule
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoExecute
openshiftStateMetrics:
nodeSelector:
  node-role.kubernetes.io/infra: ""
tolerations:
- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoSchedule
```

```

- key: node-role.kubernetes.io/infra
  value: reserved
  effect: NoExecute
thanosQuerier:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
  tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
monitoringPlugin:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
  tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute

```

- 1** Add a **nodeSelector** parameter with the appropriate value to the component you want to move. You can use a **nodeSelector** in the format shown or use **<key>: <value>** pairs, based on the value specified for the node. If you added a taint to the infrastructure node, also add a matching toleration.

2. Watch the monitoring pods move to the new machines:

```
$ watch 'oc get pod -n openshift-monitoring -o wide'
```

3. If a component has not moved to the **infra** node, delete the pod with this component:

```
$ oc delete pod -n openshift-monitoring <pod>
```

The component from the deleted pod is re-created on the **infra** node.

#### 8.4.4. Moving the Vertical Pod Autoscaler Operator components

The Vertical Pod Autoscaler Operator (VPA) consists of three components: the recommender, updater, and admission controller. The Operator and each component has its own pod in the VPA namespace on the control plane nodes. You can move the VPA Operator and component pods to infrastructure nodes by adding a node selector to the VPA subscription and the **VerticalPodAutoscalerController** CR.

The following example shows the default deployment of the VPA pods to the control plane nodes.

#### Example output

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE READINESS GATES						
vertical-pod-autoscaler-operator-6c75fcc9cd-5pb6z	1/1	Running	0	7m59s	10.128.2.24	
c416-tfsbj-master-1 <none>	<none>					

```
vpa-admission-plugin-default-6cb78d6f8b-rpcrj    1/1  Running  0      5m37s  10.129.2.22  
c416-tfsbj-master-1 <none>        <none>  
vpa-recommender-default-66846bd94c-dsmpp       1/1  Running  0      5m37s  10.129.2.20  
c416-tfsbj-master-0 <none>        <none>  
vpa-updater-default-db8b58df-2nkvf      1/1  Running  0      5m37s  10.129.2.21  c416-  
tfsbj-master-1 <none>        <none>
```

## Procedure

- Move the VPA Operator pod by adding a node selector to the **Subscription** custom resource (CR) for the VPA Operator:

- Edit the CR:

```
$ oc edit Subscription vertical-pod-autoscaler -n openshift-vertical-pod-autoscaler
```

- Add a node selector to match the node role label on the infra node:

```
apiVersion: operators.coreos.com/v1alpha1  
kind: Subscription  
metadata:  
  labels:  
    operators.coreos.com/vertical-pod-autoscaler.openshift-vertical-pod-autoscaler: ""  
  name: vertical-pod-autoscaler  
  # ...  
spec:  
  config:  
    nodeSelector:  
      node-role.kubernetes.io/infra: "" 1
```

- 1** Specifies the node role of an infra node.



## NOTE

If the infra node uses taints, you need to add a toleration to the **Subscription** CR.

For example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  labels:
    operators.coreos.com/vertical-pod-autoscaler.openshift-vertical-pod-
    autoscaler: ""
  name: vertical-pod-autoscaler
# ...
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  tolerations: ①
    - key: "node-role.kubernetes.io/infra"
      operator: "Exists"
      effect: "NoSchedule"
```

- ① Specifies a toleration for a taint on the infra node.
- 2. Move each VPA component by adding node selectors to the **VerticalPodAutoscaler** custom resource (CR):

- a. Edit the CR:

```
$ oc edit VerticalPodAutoscalerController default -n openshift-vertical-pod-autoscaler
```

- b. Add node selectors to match the node role label on the infra node:

```
apiVersion: autoscaling.openshift.io/v1
kind: VerticalPodAutoscalerController
metadata:
  name: default
  namespace: openshift-vertical-pod-autoscaler
# ...
spec:
  deploymentOverrides:
    admission:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: "" ①
    recommender:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: "" ②
    updater:
```

```
  container:  
    resources: {}  
  nodeSelector:  
    node-role.kubernetes.io/infra: "" ③
```

- ① Optional: Specifies the node role for the VPA admission pod.
- ② Optional: Specifies the node role for the VPA recommender pod.
- ③ Optional: Specifies the node role for the VPA updater pod.

## NOTE

If a target node uses taints, you need to add a toleration to the **VerticalPodAutoscalerController** CR.

For example:

```
apiVersion: autoscaling.openshift.io/v1  
kind: VerticalPodAutoscalerController  
metadata:  
  name: default  
  namespace: openshift-vertical-pod-autoscaler  
# ...  
spec:  
  deploymentOverrides:  
    admission:  
      container:  
        resources: {}  
      nodeSelector:  
        node-role.kubernetes.io/infra: ""  
      tolerations: ①  
        - key: "my-example-node-taint-key"  
          operator: "Exists"  
          effect: "NoSchedule"  
    recommender:  
      container:  
        resources: {}  
      nodeSelector:  
        node-role.kubernetes.io/infra: ""  
      tolerations: ②  
        - key: "my-example-node-taint-key"  
          operator: "Exists"  
          effect: "NoSchedule"  
    updater:  
      container:  
        resources: {}  
      nodeSelector:  
        node-role.kubernetes.io/infra: ""  
      tolerations: ③  
        - key: "my-example-node-taint-key"  
          operator: "Exists"  
          effect: "NoSchedule"
```

- 1 Specifies a toleration for the admission controller pod for a taint on the infra node.
- 2 Specifies a toleration for the recommender pod for a taint on the infra node.
- 3 Specifies a toleration for the updater pod for a taint on the infra node.

## Verification

- You can verify the pods have moved by using the following command:

```
$ oc get pods -n openshift-vertical-pod-autoscaler -o wide
```

The pods are no longer deployed to the control plane nodes.

## Example output

NAME	READY	STATUS	RESTARTS	AGE	IP
NODE	NOMINATED NODE	READINESS	GATES		
vertical-pod-autoscaler-operator-6c75fcc9cd-5pb6z	1/1	Running	0	7m59s	
10.128.2.24 c416-tfsbj-infra-eastus3-2bndt	<none>	<none>			
vpa-admission-plugin-default-6cb78d6f8b-rpcrj	1/1	Running	0	5m37s	
10.129.2.22 c416-tfsbj-infra-eastus1-lrgj8	<none>	<none>			
vpa-recommender-default-66846bd94c-dsmpp	1/1	Running	0	5m37s	
10.129.2.20 c416-tfsbj-infra-eastus1-lrgj8	<none>	<none>			
vpa-updater-default-db8b58df-2nkvf	1/1	Running	0	5m37s	10.129.2.21
c416-tfsbj-infra-eastus1-lrgj8	<none>	<none>			

## 8.4.5. Moving the Cluster Resource Override Operator pods

By default, the Cluster Resource Override Operator installation process creates an Operator pod and two Cluster Resource Override pods on nodes in the **clusterresourceoverride-operator** namespace. You can move these pods to other nodes, such as infrastructure nodes, as needed.

The following examples shows the Cluster Resource Override pods are deployed to control plane nodes and the Cluster Resource Override Operator pod is deployed to a worker node.

## Example Cluster Resource Override pods

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE	READINESS	GATES				
clusterresourceoverride-786b8c898c-9wrdq	1/1	Running	0	23s	10.128.2.32	ip-10-0-14-183.us-west-2.compute.internal
<none>	<none>					
clusterresourceoverride-786b8c898c-vn2lf	1/1	Running	0	26s	10.130.2.10	ip-10-0-20-140.us-west-2.compute.internal
<none>	<none>					
clusterresourceoverride-operator-6b8b8b656b-lvr62	1/1	Running	0	56m	10.131.0.33	ip-10-0-2-39.us-west-2.compute.internal
<none>	<none>					

## Example node list

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-14-183.us-west-2.compute.internal	Ready	control-plane,master	65m	v1.30.4
ip-10-0-2-39.us-west-2.compute.internal	Ready	worker	58m	v1.30.4
ip-10-0-20-140.us-west-2.compute.internal	Ready	control-plane,master	65m	v1.30.4

ip-10-0-23-244.us-west-2.compute.internal	Ready	infra	55m	v1.30.4
ip-10-0-77-153.us-west-2.compute.internal	Ready	control-plane,master	65m	v1.30.4
ip-10-0-99-108.us-west-2.compute.internal	Ready	worker	24m	v1.30.4
ip-10-0-24-233.us-west-2.compute.internal	Ready	infra	55m	v1.30.4
ip-10-0-88-109.us-west-2.compute.internal	Ready	worker	24m	v1.30.4
ip-10-0-67-453.us-west-2.compute.internal	Ready	infra	55m	v1.30.4

## Procedure

- Move the Cluster Resource Override Operator pod by adding a node selector to the **Subscription** custom resource (CR) for the Cluster Resource Override Operator.

- Edit the CR:

```
$ oc edit -n clusterresourceoverride-operator subscriptions.coreos.com
clusterresourceoverride
```

- Add a node selector to match the node role label on the node where you want to install the Cluster Resource Override Operator pod:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: clusterresourceoverride
  namespace: clusterresourceoverride-operator
# ...
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: "" ①
# ...
```

- ① Specify the role of the node where you want to deploy the Cluster Resource Override Operator pod.



## NOTE

If the infra node uses taints, you need to add a toleration to the **Subscription** CR.

For example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: clusterresourceoverride
  namespace: clusterresourceoverride-operator
# ...
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  tolerations: ①
    - key: "node-role.kubernetes.io/infra"
      operator: "Exists"
      effect: "NoSchedule"
```

① Specifies a toleration for a taint on the infra node.

2. Move the Cluster Resource Override pods by adding a node selector to the **ClusterResourceOverride** custom resource (CR):

- a. Edit the CR:

```
$ oc edit ClusterResourceOverride cluster -n clusterresourceoverride-operator
```

- b. Add a node selector to match the node role label on the infra node:

```
apiVersion: operator.autoscaling.openshift.io/v1
kind: ClusterResourceOverride
metadata:
  name: cluster
  resourceVersion: "37952"
spec:
  podResourceOverride:
    spec:
      cpuRequestToLimitPercent: 25
      limitCPUToMemoryPercent: 200
      memoryRequestToLimitPercent: 50
  deploymentOverrides:
    replicas: 1 ①
    nodeSelector:
      node-role.kubernetes.io/infra: "" ②
# ...
```

① Optional: Specify the number of Cluster Resource Override pods to deploy. The default is ②. Only one pod is allowed per node.

②

Optional: Specify the role of the node where you want to deploy the Cluster Resource Override pods.



## NOTE

If the infra node uses taints, you need to add a toleration to the **ClusterResourceOverride** CR.

For example:

```
apiVersion: operator.autoscaling.openshift.io/v1
kind: ClusterResourceOverride
metadata:
  name: cluster
# ...
spec:
  podResourceOverride:
    spec:
      memoryRequestToLimitPercent: 50
      cpuRequestToLimitPercent: 25
      limitCPUToMemoryPercent: 200
  deploymentOverrides:
    replicas: 3
    nodeSelector:
      node-role.kubernetes.io/worker: ""
    tolerations: ①
      - key: "key"
        operator: "Equal"
        value: "value"
        effect: "NoSchedule"
```

① Specifies a toleration for a taint on the infra node.

## Verification

- You can verify that the pods have moved by using the following command:

```
$ oc get pods -n clusterresourceoverride-operator -o wide
```

The Cluster Resource Override pods are now deployed to the infra nodes.

## Example output

NAME	READY	STATUS	RESTARTS	AGE	IP
NODE	NOMINATED NODE	READINESS	GATES		
clusterresourceoverride-786b8c898c-9wrdq	<none>	1/1	Running 0	23s	10.127.2.25
ip-10-0-23-244.us-west-2.compute.internal	<none>	<none>			
clusterresourceoverride-786b8c898c-vn2lf	<none>	1/1	Running 0	26s	10.128.0.80
ip-10-0-24-233.us-west-2.compute.internal	<none>	<none>			
clusterresourceoverride-operator-6b8b8b656b-lvr62	ip-10-0-67-453.us-west-2.compute.internal	1/1	Running 0	56m	
10.129.0.71	<none>				

## Additional resources

- Moving monitoring components to different nodes

# CHAPTER 9. ADDING RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

In OpenShift Container Platform, you can add Red Hat Enterprise Linux (RHEL) compute machines to a user-provisioned infrastructure cluster or an installation-provisioned infrastructure cluster on the **x86\_64** architecture. You can use RHEL as the operating system only on compute machines.

## 9.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.17, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines in your cluster if you use a user-provisioned or installer-provisioned infrastructure installation on the **x86\_64** architecture. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane machines in your cluster.

If you choose to use RHEL compute machines in your cluster, you are responsible for all operating system life cycle management and maintenance. You must perform system updates, apply patches, and complete all other required tasks.

For installer-provisioned infrastructure clusters, you must manually add RHEL compute machines because automatic scaling in installer-provisioned infrastructure clusters adds Red Hat Enterprise Linux CoreOS (RHCOS) compute machines by default.



### IMPORTANT

- Because removing OpenShift Container Platform from a machine in the cluster requires destroying the operating system, you must use dedicated hardware for any RHEL machines that you add to the cluster.
- Swap memory is disabled on all RHEL machines that you add to your OpenShift Container Platform cluster. You cannot enable swap memory on these machines.

## 9.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

The Red Hat Enterprise Linux (RHEL) compute machine hosts in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements:

- You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.
- Production environments must provide compute machines to support your expected workloads. As a cluster administrator, you must calculate the expected workload and add about 10% for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.
- Each system must meet the following hardware requirements:
  - Physical or virtual system, or an instance running on a public or private IaaS.
  - Base OS: [RHEL 8.6 and later](#) with "Minimal" installation option.



## IMPORTANT

Adding RHEL 7 compute machines to an OpenShift Container Platform cluster is not supported.

If you have RHEL 7 compute machines that were previously supported in a past OpenShift Container Platform version, you cannot upgrade them to RHEL 8. You must deploy new RHEL 8 hosts, and the old RHEL 7 hosts should be removed. See the "Deleting nodes" section for more information.

For the most recent list of major functionality that has been deprecated or removed within OpenShift Container Platform, refer to the *Deprecated and removed features* section of the OpenShift Container Platform release notes.

- If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See [Installing a RHEL 8 system with FIPS mode enabled](#) in the RHEL 8 documentation.



## IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see [Switching RHEL to FIPS mode](#).

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86\_64, ppc64le, and s390x architectures.

- NetworkManager 1.0 or later.
- 1 vCPU.
- Minimum 8 GB RAM.
- Minimum 15 GB hard disk space for the file system containing `/var`.
- Minimum 1 GB hard disk space for the file system containing `/usr/local/bin`.
- Minimum 1 GB hard disk space for the file system containing its temporary directory. The temporary system directory is determined according to the rules defined in the `tempfile` module in the Python standard library.
- Each system must meet any additional requirements for your system provider. For example, if you installed your cluster on VMware vSphere, your disks must be configured according to its [storage guidelines](#) and the `disk.enableUUID=true` attribute must be set.
- Each system must be able to access the cluster's API endpoints by using DNS-resolvable hostnames. Any network security access control that is in place must allow system access to the cluster's API service endpoints.
- For clusters installed on Microsoft Azure:

- Ensure the system includes the hardware requirement of a **Standard\_D8s\_v3** virtual machine.
- Enable Accelerated Networking. Accelerated Networking uses single root I/O virtualization (SR-IOV) to provide Microsoft Azure VMs with a more direct path to the switch.

## Additional resources

- [Deleting nodes](#)
- [Accelerated Networking for Microsoft Azure VMs](#)

### 9.2.1. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

## 9.3. PREPARING AN IMAGE FOR YOUR CLOUD

Amazon Machine Images (AMI) are required because various image formats cannot be used directly by AWS. You may use the AMIs that Red Hat has provided, or you can manually import your own images. The AMI must exist before the EC2 instance can be provisioned. You will need a valid AMI ID so that the correct RHEL version needed for the compute machines is selected.

### 9.3.1. Listing latest available RHEL images on AWS

AMI IDs correspond to native boot images for AWS. Because an AMI must exist before the EC2 instance is provisioned, you will need to know the AMI ID before configuration. The [AWS Command Line Interface \(CLI\)](#) is used to list the available Red Hat Enterprise Linux (RHEL) image IDs.

#### Prerequisites

- You have installed the AWS CLI.

#### Procedure

- Use this command to list RHEL 8.4 Amazon Machine Images (AMI):

```
$ aws ec2 describe-images --owners 309956199498 \ ①
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ ②
--filters "Name=name,Values=RHEL-8.4*" \ ③
--region us-east-1 \ ④
--output table ⑤
```

① The **--owners** command option shows Red Hat images based on the account ID **309956199498**.



## IMPORTANT

This account ID is required to display AMI IDs for images that are provided by Red Hat.

- 2 The **--query** command option sets how the images are sorted with the parameters '**'sort\_by(Images, &CreationDate)[\*].[CreationDate,Name,ImageId]**'. In this case, the images are sorted by the creation date, and the table is structured to show the creation date, the name of the image, and the AMI IDs.
- 3 The **--filter** command option sets which version of RHEL is shown. In this example, since the filter is set by "**"Name=name,Values=RHEL-8.4"**", then RHEL 8.4 AMIs are shown.
- 4 The **--region** command option sets where the region where an AMI is stored.
- 5 The **--output** command option sets how the results are displayed.



## NOTE

When creating a RHEL compute machine for AWS, ensure that the AMI is RHEL 8.4 or 8.5.

## Example output

DescribelImages		
+	-----+-----+-----+	
2021-03-18T14:23:11.000Z   RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2   ami-07eeb4db5f7e5a8fb		
2021-03-18T14:38:28.000Z   RHEL-8.4.0_HVM_BETA-20210309-arm64-1-Hourly2-GP2   ami-069d22ec49577d4bf		
2021-05-18T19:06:34.000Z   RHEL-8.4.0_HVM-20210504-arm64-2-Hourly2-GP2   ami-01fc429821bf1f4b4		
2021-05-18T20:09:47.000Z   RHEL-8.4.0_HVM-20210504-x86_64-2-Hourly2-GP2   ami-0b0af3577fe5e3532		
+	-----+-----+-----+	

## Additional resources

- You may also manually [import RHEL images to AWS](#).

## 9.4. PREPARING THE MACHINE TO RUN THE PLAYBOOK

Before you can add compute machines that use Red Hat Enterprise Linux (RHEL) as the operating system to an OpenShift Container Platform 4.17 cluster, you must prepare a RHEL 8 machine to run an Ansible playbook that adds the new node to the cluster. This machine is not part of the cluster but must be able to access it.

## Prerequisites

- Install the OpenShift CLI (**oc**) on the machine that you run the playbook on.
- Log in as a user with **cluster-admin** permission.

## Procedure

1. Ensure that the **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the RHEL 8 machine. One way to accomplish this is to use the same machine that you used to install the cluster.
2. Configure the machine to access all of the RHEL hosts that you plan to use as compute machines. You can use any method that your company allows, including a bastion with an SSH proxy or a VPN.
3. Configure a user on the machine that you run the playbook on that has SSH access to all of the RHEL hosts.



### IMPORTANT

If you use SSH key-based authentication, you must manage the key with an SSH agent.

4. If you have not already done so, register the machine with RHSM and attach a pool with an **OpenShift** subscription to it:

- a. Register the machine with RHSM:

```
# subscription-manager register --username=<user_name> --password=<password>
```

- b. Pull the latest subscription data from RHSM:

```
# subscription-manager refresh
```

- c. List the available subscriptions:

```
# subscription-manager list --available --matches "*OpenShift*"
```

- d. In the output for the previous command, find the pool ID for an OpenShift Container Platform subscription and attach it:

```
# subscription-manager attach --pool=<pool_id>
```

5. Enable the repositories required by OpenShift Container Platform 4.17:

```
# subscription-manager repos \
--enable="rhel-8-for-x86_64-baseos-rpms" \
--enable="rhel-8-for-x86_64-appstream-rpms" \
--enable="rhocp-4.17-for-rhel-8-x86_64-rpms"
```

6. Install the required packages, including **openshift-ansible**:

```
# yum install openshift-ansible openshift-clients jq
```

The **openshift-ansible** package provides installation program utilities and pulls in other packages that you require to add a RHEL compute node to your cluster, such as Ansible, playbooks, and related configuration files. The **openshift-clients** provides the **oc** CLI, and the **jq** package improves the display of JSON output on your command line.

## 9.5. PREPARING A RHEL COMPUTE NODE

Before you add a Red Hat Enterprise Linux (RHEL) machine to your OpenShift Container Platform cluster, you must register each host with Red Hat Subscription Manager (RHSM), attach an active OpenShift Container Platform subscription, and enable the required repositories.

1. On each host, register with RHSM:

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. Pull the latest subscription data from RHSM:

```
# subscription-manager refresh
```

3. List the available subscriptions:

```
# subscription-manager list --available --matches "*OpenShift*"
```

4. In the output for the previous command, find the pool ID for an OpenShift Container Platform subscription and attach it:

```
# subscription-manager attach --pool=<pool_id>
```

5. Disable all yum repositories:

- a. Disable all the enabled RHSM repositories:

```
# subscription-manager repos --disable="*"
```

- b. List the remaining yum repositories and note their names under **repo id**, if any:

```
# yum repolist
```

- c. Use **yum-config-manager** to disable the remaining yum repositories:

```
# yum-config-manager --disable <repo_id>
```

Alternatively, disable all repositories:

```
# yum-config-manager --disable \*
```

Note that this might take a few minutes if you have a large number of available repositories

6. Enable only the repositories required by OpenShift Container Platform 4.17:

```
# subscription-manager repos \
--enable="rhel-8-for-x86_64-baseos-rpms" \
--enable="rhel-8-for-x86_64-appstream-rpms" \
--enable="rhocp-4.17-for-rhel-8-x86_64-rpms" \
--enable="fast-datapath-for-rhel-8-x86_64-rpms"
```

7. Stop and disable firewalld on the host:

```
■
```

```
# systemctl disable --now firewalld.service
```

**NOTE**

You must not enable firewalld later. If you do, you cannot access OpenShift Container Platform logs on the worker.

## 9.6. ATTACHING THE ROLE PERMISSIONS TO RHEL INSTANCE IN AWS

Using the Amazon IAM console in your browser, you may select the needed roles and assign them to a worker node.

**Procedure**

1. From the AWS IAM console, create your [desired IAM role](#).
2. [Attach the IAM role](#) to the desired worker node.

**Additional resources**

- See [Required AWS permissions for IAM roles](#).

## 9.7. TAGGING A RHEL WORKER NODE AS OWNED OR SHARED

A cluster uses the value of the **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** tag to determine the lifetime of the resources related to the AWS cluster.

- The **owned** tag value should be added if the resource should be destroyed as part of destroying the cluster.
- The **shared** tag value should be added if the resource continues to exist after the cluster has been destroyed. This tagging denotes that the cluster uses this resource, but there is a separate owner for the resource.

**Procedure**

- With RHEL compute machines, the RHEL worker instance must be tagged with **kubernetes.io/cluster/<clusterid>=owned** or **kubernetes.io/cluster/<cluster-id>=shared**.

**NOTE**

Do not tag all existing security groups with the **kubernetes.io/cluster/<name>,Value=<clusterid>** tag, or the Elastic Load Balancing (ELB) will not be able to create a load balancer.

## 9.8. ADDING A RHEL COMPUTE MACHINE TO YOUR CLUSTER

You can add compute machines that use Red Hat Enterprise Linux as the operating system to an OpenShift Container Platform 4.17 cluster.

**Prerequisites**

- You installed the required packages and performed the necessary configuration on the machine that you run the playbook on.
- You prepared the RHEL hosts for installation.

## Procedure

Perform the following steps on the machine that you prepared to run the playbook:

- 1 Create an Ansible inventory file that is named `/<path>/inventory/hosts` that defines your compute machine hosts and required variables:

```
[all:vars]
ansible_user=root ①
#ansible_become=True ②

openshift_kubeconfig_path=~/.kube/config" ③

[new_workers] ④
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com
```

- 1 Specify the user name that runs the Ansible tasks on the remote compute machines.
- 2 If you do not specify **root** for the **ansible\_user**, you must set **ansible\_become** to **True** and assign the user sudo permissions.
- 3 Specify the path and file name of the **kubeconfig** file for your cluster.
- 4 List each RHEL machine to add to your cluster. You must provide the fully-qualified domain name for each host. This name is the hostname that the cluster uses to access the machine, so set the correct public or private name to access the machine.

- 2 Navigate to the Ansible playbook directory:

```
$ cd /usr/share/ansible/openshift-ansible
```

- 3 Run the playbook:

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml ①
```

- 1 For **<path>**, specify the path to the Ansible inventory file that you created.

## 9.9. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

### Prerequisites

- You added machines to your cluster.

## Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.30.3
master-1	Ready	master	63m	v1.30.3
master-2	Ready	master	64m	v1.30.3

The output lists all of the machines that you created.



### NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



### NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.



## NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



## NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.30.3
master-1	Ready	master	73m	v1.30.3
master-2	Ready	master	74m	v1.30.3
worker-0	Ready	worker	11m	v1.30.3
worker-1	Ready	worker	11m	v1.30.3



### NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

### Additional information

- [Certificate Signing Requests](#)

## 9.10. REQUIRED PARAMETERS FOR THE ANSIBLE HOSTS FILE

You must define the following parameters in the Ansible hosts file before you add Red Hat Enterprise Linux (RHEL) compute machines to your cluster.

Parameter	Description	Values
<b>ansible_user</b>	The SSH user that allows SSH-based authentication without requiring a password. If you use SSH key-based authentication, then you must manage the key with an SSH agent.	A user name on the system. The default value is <b>root</b> .
<b>ansible_become</b>	If the values of <b>ansible_user</b> is not root, you must set <b>ansible_become</b> to <b>True</b> , and the user that you specify as the <b>ansible_user</b> must be configured for passwordless sudo access.	<b>True</b> . If the value is not <b>True</b> , do not specify and define this parameter.
<b>openshift_kube_config_path</b>	Specifies a path and file name to a local directory that contains the <b>kubeconfig</b> file for your cluster.	The path and name of the configuration file.

### 9.10.1. Optional: Removing RHCOS compute machines from a cluster

After you add the Red Hat Enterprise Linux (RHEL) compute machines to your cluster, you can optionally remove the Red Hat Enterprise Linux CoreOS (RHCOS) compute machines to free up resources.

#### Prerequisites

- You have added RHEL compute machines to your cluster.

#### Procedure

1. View the list of machines and record the node names of the RHCOS compute machines:

```
$ oc get nodes -o wide
```

2. For each RHCOS compute machine, delete the node:

- a. Mark the node as unschedulable by running the **oc adm cordon** command:

```
$ oc adm cordon <node_name> ①
```

① Specify the node name of one of the RHCOS compute machines.

- b. Drain all the pods from the node:

```
$ oc adm drain <node_name> --force --delete-emptydir-data --ignore-daemonsets ①
```

① Specify the node name of the RHCOS compute machine that you isolated.

- c. Delete the node:

```
$ oc delete nodes <node_name> ①
```

① Specify the node name of the RHCOS compute machine that you drained.

3. Review the list of compute machines to ensure that only the RHEL nodes remain:

```
$ oc get nodes -o wide
```

4. Remove the RHCOS machines from the load balancer for your cluster's compute machines. You can delete the virtual machines or reimage the physical hardware for the RHCOS compute machines.

# CHAPTER 10. ADDING MORE RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

If your OpenShift Container Platform cluster already includes Red Hat Enterprise Linux (RHEL) compute machines, which are also known as worker machines, you can add more RHEL compute machines to it.

## 10.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.17, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines in your cluster if you use a user-provisioned or installer-provisioned infrastructure installation on the **x86\_64** architecture. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane machines in your cluster.

If you choose to use RHEL compute machines in your cluster, you are responsible for all operating system life cycle management and maintenance. You must perform system updates, apply patches, and complete all other required tasks.

For installer-provisioned infrastructure clusters, you must manually add RHEL compute machines because automatic scaling in installer-provisioned infrastructure clusters adds Red Hat Enterprise Linux CoreOS (RHCOS) compute machines by default.



### IMPORTANT

- Because removing OpenShift Container Platform from a machine in the cluster requires destroying the operating system, you must use dedicated hardware for any RHEL machines that you add to the cluster.
- Swap memory is disabled on all RHEL machines that you add to your OpenShift Container Platform cluster. You cannot enable swap memory on these machines.

## 10.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

The Red Hat Enterprise Linux (RHEL) compute machine hosts in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements:

- You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.
- Production environments must provide compute machines to support your expected workloads. As a cluster administrator, you must calculate the expected workload and add about 10% for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.
- Each system must meet the following hardware requirements:
  - Physical or virtual system, or an instance running on a public or private IaaS.
  - Base OS: [RHEL 8.6 and later](#) with "Minimal" installation option.



## IMPORTANT

Adding RHEL 7 compute machines to an OpenShift Container Platform cluster is not supported.

If you have RHEL 7 compute machines that were previously supported in a past OpenShift Container Platform version, you cannot upgrade them to RHEL 8. You must deploy new RHEL 8 hosts, and the old RHEL 7 hosts should be removed. See the "Deleting nodes" section for more information.

For the most recent list of major functionality that has been deprecated or removed within OpenShift Container Platform, refer to the *Deprecated and removed features* section of the OpenShift Container Platform release notes.

- If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See [Installing a RHEL 8 system with FIPS mode enabled](#) in the RHEL 8 documentation.



## IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see [Switching RHEL to FIPS mode](#).

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86\_64, ppc64le, and s390x architectures.

- NetworkManager 1.0 or later.
- 1 vCPU.
- Minimum 8 GB RAM.
- Minimum 15 GB hard disk space for the file system containing `/var`.
- Minimum 1 GB hard disk space for the file system containing `/usr/local/bin`.
- Minimum 1 GB hard disk space for the file system containing its temporary directory. The temporary system directory is determined according to the rules defined in the `tempfile` module in the Python standard library.
- Each system must meet any additional requirements for your system provider. For example, if you installed your cluster on VMware vSphere, your disks must be configured according to its [storage guidelines](#) and the `disk.enableUUID=true` attribute must be set.
- Each system must be able to access the cluster's API endpoints by using DNS-resolvable hostnames. Any network security access control that is in place must allow system access to the cluster's API service endpoints.
- For clusters installed on Microsoft Azure:

- Ensure the system includes the hardware requirement of a **Standard\_D8s\_v3** virtual machine.
- Enable Accelerated Networking. Accelerated Networking uses single root I/O virtualization (SR-IOV) to provide Microsoft Azure VMs with a more direct path to the switch.

## Additional resources

- [Deleting nodes](#)
- [Accelerated Networking for Microsoft Azure VMs](#)

### 10.2.1. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

## 10.3. PREPARING AN IMAGE FOR YOUR CLOUD

Amazon Machine Images (AMI) are required since various image formats cannot be used directly by AWS. You may use the AMIs that Red Hat has provided, or you can manually import your own images. The AMI must exist before the EC2 instance can be provisioned. You must list the AMI IDs so that the correct RHEL version needed for the compute machines is selected.

### 10.3.1. Listing latest available RHEL images on AWS

AMI IDs correspond to native boot images for AWS. Because an AMI must exist before the EC2 instance is provisioned, you will need to know the AMI ID before configuration. The [AWS Command Line Interface \(CLI\)](#) is used to list the available Red Hat Enterprise Linux (RHEL) image IDs.

#### Prerequisites

- You have installed the AWS CLI.

#### Procedure

- Use this command to list RHEL 8.4 Amazon Machine Images (AMI):

```
$ aws ec2 describe-images --owners 309956199498 \ ①
--query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ ②
--filters "Name=name,Values=RHEL-8.4*" \ ③
--region us-east-1 \ ④
--output table ⑤
```

**1** The **--owners** command option shows Red Hat images based on the account ID **309956199498**.



## IMPORTANT

This account ID is required to display AMI IDs for images that are provided by Red Hat.

- 2 The **--query** command option sets how the images are sorted with the parameters '**'sort\_by(Images, &CreationDate)[\*].[CreationDate,Name,ImageId]**'. In this case, the images are sorted by the creation date, and the table is structured to show the creation date, the name of the image, and the AMI IDs.
- 3 The **--filter** command option sets which version of RHEL is shown. In this example, since the filter is set by "**"Name=name,Values=RHEL-8.4"**", then RHEL 8.4 AMIs are shown.
- 4 The **--region** command option sets where the region where an AMI is stored.
- 5 The **--output** command option sets how the results are displayed.



## NOTE

When creating a RHEL compute machine for AWS, ensure that the AMI is RHEL 8.4 or 8.5.

## Example output

DescribelImages		
+	-----+-----+-----+	
2021-03-18T14:23:11.000Z   RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2   ami-07eeb4db5f7e5a8fb		
2021-03-18T14:38:28.000Z   RHEL-8.4.0_HVM_BETA-20210309-arm64-1-Hourly2-GP2   ami-069d22ec49577d4bf		
2021-05-18T19:06:34.000Z   RHEL-8.4.0_HVM-20210504-arm64-2-Hourly2-GP2   ami-01fc429821bf1f4b4		
2021-05-18T20:09:47.000Z   RHEL-8.4.0_HVM-20210504-x86_64-2-Hourly2-GP2   ami-0b0af3577fe5e3532		
+	-----+-----+-----+	

## Additional resources

- You may also manually [import RHEL images to AWS](#).

## 10.4. PREPARING A RHEL COMPUTE NODE

Before you add a Red Hat Enterprise Linux (RHEL) machine to your OpenShift Container Platform cluster, you must register each host with Red Hat Subscription Manager (RHSM), attach an active OpenShift Container Platform subscription, and enable the required repositories.

1. On each host, register with RHSM:

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. Pull the latest subscription data from RHSM:

```
# subscription-manager refresh
```

3. List the available subscriptions:

```
# subscription-manager list --available --matches "*OpenShift*"
```

4. In the output for the previous command, find the pool ID for an OpenShift Container Platform subscription and attach it:

```
# subscription-manager attach --pool=<pool_id>
```

5. Disable all yum repositories:

- a. Disable all the enabled RHSMS repositories:

```
# subscription-manager repos --disable="*"
```

- b. List the remaining yum repositories and note their names under **repo id**, if any:

```
# yum repolist
```

- c. Use **yum-config-manager** to disable the remaining yum repositories:

```
# yum-config-manager --disable <repo_id>
```

Alternatively, disable all repositories:

```
# yum-config-manager --disable \*
```

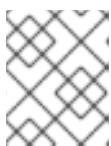
Note that this might take a few minutes if you have a large number of available repositories

6. Enable only the repositories required by OpenShift Container Platform 4.17:

```
# subscription-manager repos \
--enable="rhel-8-for-x86_64-baseos-rpms" \
--enable="rhel-8-for-x86_64-appstream-rpms" \
--enable="rhocp-4.17-for-rhel-8-x86_64-rpms" \
--enable="fast-datapath-for-rhel-8-x86_64-rpms"
```

7. Stop and disable firewalld on the host:

```
# systemctl disable --now firewalld.service
```



#### NOTE

You must not enable firewalld later. If you do, you cannot access OpenShift Container Platform logs on the worker.

## 10.5. ATTACHING THE ROLE PERMISSIONS TO RHEL INSTANCE IN AWS

Using the Amazon IAM console in your browser, you may select the needed roles and assign them to a worker node.

### Procedure

1. From the AWS IAM console, create your [desired IAM role](#).
2. [Attach the IAM role](#) to the desired worker node.

### Additional resources

- See [Required AWS permissions for IAM roles](#).

## 10.6. TAGGING A RHEL WORKER NODE AS OWNED OR SHARED

A cluster uses the value of the **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** tag to determine the lifetime of the resources related to the AWS cluster.

- The **owned** tag value should be added if the resource should be destroyed as part of destroying the cluster.
- The **shared** tag value should be added if the resource continues to exist after the cluster has been destroyed. This tagging denotes that the cluster uses this resource, but there is a separate owner for the resource.

### Procedure

- With RHEL compute machines, the RHEL worker instance must be tagged with **kubernetes.io/cluster/<clusterid>=owned** or **kubernetes.io/cluster/<cluster-id>=shared**.



### NOTE

Do not tag all existing security groups with the **kubernetes.io/cluster/<name>,Value=<clusterid>** tag, or the Elastic Load Balancing (ELB) will not be able to create a load balancer.

## 10.7. ADDING MORE RHEL COMPUTE MACHINES TO YOUR CLUSTER

You can add more compute machines that use Red Hat Enterprise Linux (RHEL) as the operating system to an OpenShift Container Platform 4.17 cluster.

### Prerequisites

- Your OpenShift Container Platform cluster already contains RHEL compute nodes.
- The **hosts** file that you used to add the first RHEL compute machines to your cluster is on the machine that you use the run the playbook.
- The machine that you run the playbook on must be able to access all of the RHEL hosts. You can use any method that your company allows, including a bastion with an SSH proxy or a VPN.
- The **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the machine that you use the run the playbook.

- You must prepare the RHEL hosts for installation.
- Configure a user on the machine that you run the playbook on that has SSH access to all of the RHEL hosts.
- If you use SSH key-based authentication, you must manage the key with an SSH agent.
- Install the OpenShift CLI (**oc**) on the machine that you run the playbook on.

## Procedure

1. Open the Ansible inventory file at /<path>/inventory/hosts that defines your compute machine hosts and required variables.
2. Rename the **[new\_workers]** section of the file to **[workers]**.
3. Add a **[new\_workers]** section to the file and define the fully-qualified domain names for each new host. The file resembles the following example:

```
[all:vars]
ansible_user=root
#ansible_become=True

openshift_kubeconfig_path=~/.kube/config

[workers]
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com

[new_workers]
mycluster-rhel8-2.example.com
mycluster-rhel8-3.example.com
```

In this example, the **mycluster-rhel8-0.example.com** and **mycluster-rhel8-1.example.com** machines are in the cluster and you add the **mycluster-rhel8-2.example.com** and **mycluster-rhel8-3.example.com** machines.

4. Navigate to the Ansible playbook directory:

```
$ cd /usr/share/ansible/openshift-ansible
```

5. Run the scaleup playbook:

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml ①
```

- ① For **<path>**, specify the path to the Ansible inventory file that you created.

## 10.8. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

## Prerequisites

- You added machines to your cluster.

## Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.30.3
master-1	Ready	master	63m	v1.30.3
master-2	Ready	master	64m	v1.30.3

The output lists all of the machines that you created.



### NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



## NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approvers** if the Kubelet requests a new certificate with identical parameters.



## NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



## NOTE

Some Operators might not become available until some CSRs are approved.

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
	...		

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

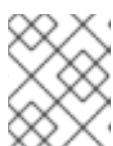
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.30.3
master-1	Ready	master	73m	v1.30.3
master-2	Ready	master	74m	v1.30.3
worker-0	Ready	worker	11m	v1.30.3
worker-1	Ready	worker	11m	v1.30.3



### NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

### Additional information

- [Certificate Signing Requests](#)

## 10.9. REQUIRED PARAMETERS FOR THE ANSIBLE HOSTS FILE

You must define the following parameters in the Ansible hosts file before you add Red Hat Enterprise Linux (RHEL) compute machines to your cluster.

Parameter	Description	Values
<b>ansible_user</b>	The SSH user that allows SSH-based authentication without requiring a password. If you use SSH key-based authentication, then you must manage the key with an SSH agent.	A user name on the system. The default value is <b>root</b> .

Parameter	Description	Values
<b>ansible_become</b>	If the values of <b>ansible_user</b> is not root, you must set <b>ansible_become</b> to <b>True</b> , and the user that you specify as the <b>ansible_user</b> must be configured for passwordless sudo access.	<b>True</b> . If the value is not <b>True</b> , do not specify and define this parameter.
<b>openshift_kubeconfig_path</b>	Specifies a path and file name to a local directory that contains the <b>kubeconfig</b> file for your cluster.	The path and name of the configuration file.

# CHAPTER 11. MANAGING USER-PROVISIONED INFRASTRUCTURE MANUALLY

## 11.1. ADDING COMPUTE MACHINES TO CLUSTERS WITH USER-PROVISIONED INFRASTRUCTURE MANUALLY

You can add compute machines to a cluster on user-provisioned infrastructure either as part of the installation process or after installation. The postinstallation process requires some of the same configuration files and parameters that were used during installation.

### 11.1.1. Adding compute machines to Amazon Web Services

To add more compute machines to your OpenShift Container Platform cluster on Amazon Web Services (AWS), see [Adding compute machines to AWS by using CloudFormation templates](#).

### 11.1.2. Adding compute machines to Microsoft Azure

To add more compute machines to your OpenShift Container Platform cluster on Microsoft Azure, see [Creating additional worker machines in Azure](#).

### 11.1.3. Adding compute machines to Azure Stack Hub

To add more compute machines to your OpenShift Container Platform cluster on Azure Stack Hub, see [Creating additional worker machines in Azure Stack Hub](#).

### 11.1.4. Adding compute machines to Google Cloud Platform

To add more compute machines to your OpenShift Container Platform cluster on Google Cloud Platform (GCP), see [Creating additional worker machines in GCP](#).

### 11.1.5. Adding compute machines to vSphere

You can [use compute machine sets](#) to automate the creation of additional compute machines for your OpenShift Container Platform cluster on vSphere.

To manually add more compute machines to your cluster, see [Adding compute machines to vSphere manually](#).

### 11.1.6. Adding compute machines to bare metal

To add more compute machines to your OpenShift Container Platform cluster on bare metal, see [Adding compute machines to bare metal](#).

## 11.2. ADDING COMPUTE MACHINES TO AWS BY USING CLOUDFORMATION TEMPLATES

You can add more compute machines to your OpenShift Container Platform cluster on Amazon Web Services (AWS) that you created by using the sample CloudFormation templates.

### 11.2.1. Prerequisites

- You installed your cluster on AWS by using the provided [AWS CloudFormation templates](#).
- You have the JSON file and CloudFormation template that you used to create the compute machines during cluster installation. If you do not have these files, you must recreate them by following the instructions in the [installation procedure](#).

### 11.2.2. Adding more compute machines to your AWS cluster by using CloudFormation templates

You can add more compute machines to your OpenShift Container Platform cluster on Amazon Web Services (AWS) that you created by using the sample CloudFormation templates.



#### IMPORTANT

The CloudFormation template creates a stack that represents one compute machine. You must create a stack for each compute machine.



#### NOTE

If you do not use the provided CloudFormation template to create your compute nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

#### Prerequisites

- You installed an OpenShift Container Platform cluster by using CloudFormation templates and have access to the JSON file and CloudFormation template that you used to create the compute machines during cluster installation.
- You installed the AWS CLI.

#### Procedure

1. Create another compute stack.

- a. Launch the template:

```
$ aws cloudformation create-stack --stack-name <name> \ ①
    --template-body file://<template>.yaml \ ②
    --parameters file://<parameters>.json ③
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-workers**. You must provide the name of this stack if you remove the cluster.
    - 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
    - 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- b. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2. Continue to create compute stacks until you have created enough compute machines for your cluster.

### 11.2.3. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

#### Prerequisites

- You added machines to your cluster.

#### Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

#### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.30.3
master-1	Ready	master	63m	v1.30.3
master-2	Ready	master	64m	v1.30.3

The output lists all of the machines that you created.



#### NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

#### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



#### NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approvers** if the Kubelet requests a new certificate with identical parameters.



#### NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



#### NOTE

Some Operators might not become available until some CSRs are approved.

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

#### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	
		Pending	

```
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

#### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.30.3
master-1	Ready	master	73m	v1.30.3
master-2	Ready	master	74m	v1.30.3
worker-0	Ready	worker	11m	v1.30.3
worker-1	Ready	worker	11m	v1.30.3



#### NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

#### Additional information

- [Certificate Signing Requests](#)

### 11.3. ADDING COMPUTE MACHINES TO VSphere MANUALLY

You can add more compute machines to your OpenShift Container Platform cluster on VMware vSphere manually.



#### NOTE

You can also [use compute machine sets](#) to automate the creation of additional VMware vSphere compute machines for your cluster.

### 11.3.1. Prerequisites

- You [installed a cluster on vSphere](#).
- You have installation media and Red Hat Enterprise Linux CoreOS (RHCOS) images that you used to create your cluster. If you do not have these files, you must obtain them by following the instructions in the [installation procedure](#).



#### IMPORTANT

If you do not have access to the Red Hat Enterprise Linux CoreOS (RHCOS) images that were used to create your cluster, you can add more compute machines to your OpenShift Container Platform cluster with newer versions of Red Hat Enterprise Linux CoreOS (RHCOS) images. For instructions, see [Adding new nodes to UPI cluster fails after upgrading to OpenShift 4.6+](#).

### 11.3.2. Adding more compute machines to a cluster in vSphere

You can add more compute machines to a user-provisioned OpenShift Container Platform cluster on VMware vSphere.

After your vSphere template deploys in your OpenShift Container Platform cluster, you can deploy a virtual machine (VM) for a machine in that cluster.

#### Prerequisites

- Obtain the base64-encoded Ignition file for your compute machines.
- You have access to the vSphere template that you created for your cluster.

#### Procedure

1. Right-click the template's name and click **Clone** → **Clone to Virtual Machine**
2. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **compute-1**.



#### NOTE

Ensure that all virtual machine names across a vSphere installation are unique.

3. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.
4. On the **Select a compute resource** tab, select the name of a host in your data center.
5. On the **Select storage** tab, select storage for your configuration and disk files.
6. On the **Select clone options** tab, select **Customize this virtual machine's hardware**.
7. On the **Customize hardware** tab, click **Advanced Parameters**.
  - Add the following configuration parameter names and values by specifying data in the **Attribute** and **Values** fields. Ensure that you select the **Add** button for each parameter that you create.

- **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded compute Ignition config file for this machine type.
  - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
  - **disk.EnableUUID**: Specify **TRUE**.
8. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type. If many networks exist, select **Add New Device > Network Adapter**, and then enter your network information in the fields provided by the **New Network** menu item.
  9. Complete the remaining configuration steps. On clicking the **Finish** button, you have completed the cloning operation.
  10. From the **Virtual Machines** tab, right-click on your VM and then select **Power → Power On**.

### Next steps

- Continue to create more compute machines for your cluster.

### 11.3.3. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

#### Prerequisites

- You added machines to your cluster.

#### Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

#### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.30.3
master-1	Ready	master	63m	v1.30.3
master-2	Ready	master	64m	v1.30.3

The output lists all of the machines that you created.



#### NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



#### NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.



#### NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

- ① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

**NOTE**

Some Operators might not become available until some CSRs are approved.

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

**Example output**

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c571v	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

**Example output**

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.30.3
master-1	Ready	master	73m	v1.30.3
master-2	Ready	master	74m	v1.30.3
worker-0	Ready	worker	11m	v1.30.3
worker-1	Ready	worker	11m	v1.30.3

**NOTE**

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

## Additional information

- [Certificate Signing Requests](#)

## 11.4. ADDING COMPUTE MACHINES TO BARE METAL

You can add more compute machines to your OpenShift Container Platform cluster on bare metal.

### 11.4.1. Prerequisites

- You [installed a cluster on bare metal](#).
- You have installation media and Red Hat Enterprise Linux CoreOS (RHCOS) images that you used to create your cluster. If you do not have these files, you must obtain them by following the instructions in the [installation procedure](#).
- If a DHCP server is available for your user-provisioned infrastructure, you have added the details for the additional compute machines to your DHCP server configuration. This includes a persistent IP address, DNS server information, and a hostname for each machine.
- You have updated your DNS configuration to include the record name and IP address of each compute machine that you are adding. You have validated that DNS lookup and reverse DNS lookup resolve correctly.



#### IMPORTANT

If you do not have access to the Red Hat Enterprise Linux CoreOS (RHCOS) images that were used to create your cluster, you can add more compute machines to your OpenShift Container Platform cluster with newer versions of Red Hat Enterprise Linux CoreOS (RHCOS) images. For instructions, see [Adding new nodes to UPI cluster fails after upgrading to OpenShift 4.6+](#).

### 11.4.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you add more compute machines to a cluster that you installed on bare metal infrastructure, you must create RHCOS machines for it to use. You can either use an ISO image or network PXE booting to create the machines.



#### NOTE

You must use the same ISO image that you used to install a cluster to deploy all new nodes in a cluster. It is recommended to use the same Ignition config file. The nodes automatically upgrade themselves on the first boot before running the workloads. You can add the nodes before or after the upgrade.

#### 11.4.2.1. Creating RHCOS machines using an ISO image

You can create more Red Hat Enterprise Linux CoreOS (RHCOS) compute machines for your bare metal cluster by using an ISO image to create the machines.

#### Prerequisites

- Obtain the URL of the Ignition config file for the compute machines for your cluster. You uploaded this file to your HTTP server during installation.

- You must have the OpenShift CLI (**oc**) installed.

## Procedure

1. Extract the Ignition config file from the cluster by running the following command:
 

```
$ oc extract -n openshift-machine-api secret/worker-user-data-managed --keys=userData --to=- > worker.ign
```
2. Upload the **worker.ign** Ignition config file you exported from your cluster to your HTTP server. Note the URLs of these files.
3. You can validate that the ignition files are available on the URLs. The following example gets the Ignition config files for the compute node:
 

```
$ curl -k http://<HTTP_server>/worker.ign
```
4. You can access the ISO image for booting your new machine by running to following command:
 

```
RHCOS_VHD_ORIGIN_URL=$(oc -n openshift-machine-config-operator get configmap/coreos-bootimages -o jsonpath='{.data.stream}' | jq -r '.architectures.<architecture>.artifacts.metal.formats.iso.disk.location')
```
5. Use the ISO file to install RHCOS on more compute machines. Use the same method that you used when you created machines before you installed the cluster:
  - Burn the ISO image to a disk and boot it directly.
  - Use ISO redirection with a LOM interface.
6. Boot the RHCOS ISO image without specifying any options, or interrupting the live boot sequence. Wait for the installer to boot into a shell prompt in the RHCOS live environment.



### NOTE

You can interrupt the RHCOS installation boot process to add kernel arguments. However, for this ISO procedure you must use the **coreos-installer** command as outlined in the following steps, instead of adding kernel arguments.

7. Run the **coreos-installer** command and specify the options that meet your installation requirements. At a minimum, you must specify the URL that points to the Ignition config file for the node type, and the device that you are installing to:
 

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device> --ignition-hash=sha512-<digest> ①②
```

- 1 You must run the **coreos-installer** command by using **sudo**, because the **core** user does not have the required root privileges to perform the installation.
- 2 The **--ignition-hash** option is required when the Ignition config file is obtained through an HTTP URL to validate the authenticity of the Ignition config file on the cluster node. **<digest>** is the Ignition config file SHA512 digest obtained in a preceding step.



## NOTE

If you want to provide your Ignition config files through an HTTPS server that uses TLS, you can add the internal certificate authority (CA) to the system trust store before running **coreos-installer**.

The following example initializes a bootstrap node installation to the **/dev/sda** device. The Ignition config file for the bootstrap node is obtained from an HTTP web server with the IP address 192.168.1.2:

```
$ sudo coreos-installer install --ignition-url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf0116e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. Monitor the progress of the RHCOS installation on the console of the machine.



## IMPORTANT

Ensure that the installation is successful on each node before commencing with the OpenShift Container Platform installation. Observing the installation process can also help to determine the cause of RHCOS installation issues that might arise.

9. Continue to create more compute machines for your cluster.

### 11.4.2.2. Creating RHCOS machines by PXE or iPXE booting

You can create more Red Hat Enterprise Linux CoreOS (RHCOS) compute machines for your bare metal cluster by using PXE or iPXE booting.

#### Prerequisites

- Obtain the URL of the Ignition config file for the compute machines for your cluster. You uploaded this file to your HTTP server during installation.
- Obtain the URLs of the RHCOS ISO image, compressed metal BIOS, **kernel**, and **initramfs** files that you uploaded to your HTTP server during cluster installation.
- You have access to the PXE booting infrastructure that you used to create the machines for your OpenShift Container Platform cluster during installation. The machines must boot from their local disks after RHCOS is installed on them.
- If you use UEFI, you have access to the **grub.conf** file that you modified during OpenShift Container Platform installation.

#### Procedure

1. Confirm that your PXE or iPXE installation for the RHCOS images is correct.

- For PXE:

```
DEFAULT pxeboot
TIMEOUT 20
```

## PROMPT 0

LABEL pxeboot

```
KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> ①
APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img ②
```

- ① Specify the location of the live **kernel** file that you uploaded to your HTTP server.
- ② Specify locations of the RHCOS files that you uploaded to your HTTP server. The **initrd** parameter value is the location of the live **initramfs** file, the **coreos.inst.ignition\_url** parameter value is the location of the worker Ignition config file, and the **coreos.live.rootfs\_url** parameter value is the location of the live **rootfs** file. The **coreos.inst.ignition\_url** and **coreos.live.rootfs\_url** parameters only support HTTP and HTTPS.



## NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0** **console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

- For iPXE (**x86\_64 + aarch64**):

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign ① ②
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img ③
boot
```

- ① Specify the locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd=main** argument is needed for booting on UEFI systems, the **coreos.live.rootfs\_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition\_url** parameter value is the location of the worker Ignition config file.
- ② If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- ③ Specify the location of the **initramfs** file that you uploaded to your HTTP server.



## NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **kernel** line. For example, add **console=tty0** **console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#) and "Enabling the serial console for PXE and ISO installation" in the "Advanced RHCOS installation configuration" section.



## NOTE

To network boot the CoreOS **kernel** on **aarch64** architecture, you need to use a version of iPXE build with the **IMAGE\_GZIP** option enabled. See [IMAGE\\_GZIP option in iPXE](#).

- For PXE (with UEFI and GRUB as second stage) on **aarch64**:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
    coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
    <architecture>.img coreos.inst.install_dev=/dev/sda
    coreos.inst.ignition_url=http://<HTTP_server>/worker.ign ① ②
    initrd rhcos-<version>-live-initramfs.<architecture>.img ③
}
```

- Specify the locations of the RHCOS files that you uploaded to your HTTP/TFTP server. The **kernel** parameter value is the location of the **kernel** file on your TFTP server. The **coreos.live.rootfs\_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition\_url** parameter value is the location of the worker Ignition config file on your HTTP Server.
- If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- Specify the location of the **initramfs** file that you uploaded to your TFTP server.

- Use the PXE or iPXE infrastructure to create the required compute machines for your cluster.

### 11.4.3. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

#### Prerequisites

- You added machines to your cluster.

#### Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.30.3
master-1	Ready	master	63m	v1.30.3
master-2	Ready	master	64m	v1.30.3

The output lists all of the machines that you created.



### NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



### NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.



## NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



## NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

### Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	
Pending			
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	
Pending			
...			

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ①
```

① **<csr\_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

### Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.30.3
master-1	Ready	master	73m	v1.30.3
master-2	Ready	master	74m	v1.30.3
worker-0	Ready	worker	11m	v1.30.3
worker-1	Ready	worker	11m	v1.30.3



### NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

### Additional information

- [Certificate Signing Requests](#)

# CHAPTER 12. MANAGING CONTROL PLANE MACHINES

## 12.1. ABOUT CONTROL PLANE MACHINE SETS

With control plane machine sets, you can automate management of the control plane machine resources within your OpenShift Container Platform cluster.



### IMPORTANT

Control plane machine sets cannot manage compute machines, and compute machine sets cannot manage control plane machines.

Control plane machine sets provide for control plane machines similar management capabilities as compute machine sets provide for compute machines. However, these two types of machine sets are separate custom resources defined within the Machine API and have several fundamental differences in their architecture and functionality.

### 12.1.1. Control Plane Machine Set Operator overview

The Control Plane Machine Set Operator uses the **ControlPlaneMachineSet** custom resource (CR) to automate management of the control plane machine resources within your OpenShift Container Platform cluster.

When the state of the cluster control plane machine set is set to **Active**, the Operator ensures that the cluster has the correct number of control plane machines with the specified configuration. This allows the automated replacement of degraded control plane machines and rollout of changes to the control plane.

A cluster has only one control plane machine set, and the Operator only manages objects in the **openshift-machine-api** namespace.

#### 12.1.1.1. Control Plane Machine Set Operator limitations

The Control Plane Machine Set Operator has the following limitations:

- Only Amazon Web Services (AWS), Google Cloud Platform (GCP), IBM Power® Virtual Server, Microsoft Azure, Nutanix, VMware vSphere, and Red Hat OpenStack Platform (RHOSP) clusters are supported.
- Clusters that do not have preexisting machines that represent the control plane nodes cannot use a control plane machine set or enable the use of a control plane machine set after installation. Generally, preexisting control plane machines are only present if a cluster was installed using infrastructure provisioned by the installation program.

To determine if a cluster has the required preexisting control plane machines, run the following command as a user with administrator privileges:

```
$ oc get machine \
-n openshift-machine-api \
-l machine.openshift.io/cluster-api-machine-role=master
```

#### Example output showing preexisting control plane machines

NAME	PHASE	TYPE	REGION	ZONE	AGE
------	-------	------	--------	------	-----

```
<infrastructure_id>-master-0  Running  m6i.xlarge  us-west-1  us-west-1a  5h19m
<infrastructure_id>-master-1  Running  m6i.xlarge  us-west-1  us-west-1b  5h19m
<infrastructure_id>-master-2  Running  m6i.xlarge  us-west-1  us-west-1a  5h19m
```

### Example output missing preexisting control plane machines

No resources found in openshift-machine-api namespace.

- The Operator requires the Machine API Operator to be operational and is therefore not supported on clusters with manually provisioned machines. When installing a OpenShift Container Platform cluster with manually provisioned machines for a platform that creates an active generated **ControlPlaneMachineSet** custom resource (CR), you must remove the Kubernetes manifest files that define the control plane machine set as instructed in the installation process.
- Only clusters with three control plane machines are supported.
- Horizontal scaling of the control plane is not supported.
- Deploying Azure control plane machines on Ephemeral OS disks increases risk for data loss and is not supported.
- Deploying control plane machines as AWS Spot Instances, GCP preemptible VMs, or Azure Spot VMs is not supported.



#### IMPORTANT

Attempting to deploy control plane machines as AWS Spot Instances, GCP preemptible VMs, or Azure Spot VMs might cause the cluster to lose etcd quorum. A cluster that loses all control plane machines simultaneously is unrecoverable.

- Making changes to the control plane machine set during or prior to installation is not supported. You must make any changes to the control plane machine set only after installation.

#### 12.1.2. Additional resources

- [Control Plane Machine Set Operator reference](#)
- [\*\*ControlPlaneMachineSet\*\* custom resource](#)

## 12.2. GETTING STARTED WITH CONTROL PLANE MACHINE SETS

The process for getting started with control plane machine sets depends on the state of the **ControlPlaneMachineSet** custom resource (CR) in your cluster.

#### Clusters with an active generated CR

Clusters that have a generated CR with an active state use the control plane machine set by default. No administrator action is required.

#### Clusters with an inactive generated CR

For clusters that include an inactive generated CR, you must review the CR configuration and [activate the CR](#).

#### Clusters without a generated CR

For clusters that do not include a generated CR, you must [create and activate a CR](#) with the appropriate configuration for your cluster.

If you are uncertain about the state of the **ControlPlaneMachineSet** CR in your cluster, you can [verify the CR status](#).

### 12.2.1. Supported cloud providers

In OpenShift Container Platform 4.17, the control plane machine set is supported for Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, Nutanix, and VMware vSphere clusters.

The status of the control plane machine set after installation depends on your cloud provider and the version of OpenShift Container Platform that you installed on your cluster.

**Table 12.1. Control plane machine set implementation for OpenShift Container Platform 4.17**

Cloud provider	Active by default	Generated CR	Manual CR required
Amazon Web Services (AWS)	X [1]	X	
Google Cloud Platform (GCP)	X [2]	X	
Microsoft Azure	X [2]	X	
Nutanix	X [3]	X	
Red Hat OpenStack Platform (RHOSP)	X [3]	X	
VMware vSphere	X [4]	X	

1. AWS clusters that are upgraded from version 4.11 or earlier require [CR activation](#).
2. GCP and Azure clusters that are upgraded from version 4.12 or earlier require [CR activation](#).
3. Nutanix and RHOSP clusters that are upgraded from version 4.13 or earlier require [CR activation](#).
4. vSphere clusters that are upgraded from version 4.15 or earlier require [CR activation](#).

### 12.2.2. Checking the control plane machine set custom resource state

You can verify the existence and state of the **ControlPlaneMachineSet** custom resource (CR).

#### Procedure

- Determine the state of the CR by running the following command:

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
--namespace openshift-machine-api
```

- A result of **Active** indicates that the **ControlPlaneMachineSet** CR exists and is activated. No administrator action is required.
- A result of **Inactive** indicates that a **ControlPlaneMachineSet** CR exists but is not activated.
- A result of **NotFound** indicates that there is no existing **ControlPlaneMachineSet** CR.

## Next steps

To use the control plane machine set, you must ensure that a **ControlPlaneMachineSet** CR with the correct settings for your cluster exists.

- If your cluster has an existing CR, you must verify that the configuration in the CR is correct for your cluster.
- If your cluster does not have an existing CR, you must create one with the correct configuration for your cluster.

### 12.2.3. Activating the control plane machine set custom resource

To use the control plane machine set, you must ensure that a **ControlPlaneMachineSet** custom resource (CR) with the correct settings for your cluster exists. On a cluster with a generated CR, you must verify that the configuration in the CR is correct for your cluster and activate it.



#### NOTE

For more information about the parameters in the CR, see "Control plane machine set configuration".

#### Procedure

1. View the configuration of the CR by running the following command:

```
$ oc --namespace openshift-machine-api edit controlplanemachineset.machine.openshift.io
cluster
```

2. Change the values of any fields that are incorrect for your cluster configuration.
3. When the configuration is correct, activate the CR by setting the **.spec.state** field to **Active** and saving your changes.



#### IMPORTANT

To activate the CR, you must change the **.spec.state** field to **Active** in the same **oc edit** session that you use to update the CR configuration. If the CR is saved with the state left as **Inactive**, the control plane machine set generator resets the CR to its original settings.

#### Additional resources

- [Control plane machine set configuration](#)

## 12.2.4. Creating a control plane machine set custom resource

To use the control plane machine set, you must ensure that a **ControlPlaneMachineSet** custom resource (CR) with the correct settings for your cluster exists. On a cluster without a generated CR, you must create the CR manually and activate it.



### NOTE

For more information about the structure and parameters of the CR, see "Control plane machine set configuration".

### Procedure

- 1 Create a YAML file using the following template:

#### Control plane machine set CR YAML file template

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <cluster_id> 1
      machine.openshift.io/cluster-api-machine-role: master
      machine.openshift.io/cluster-api-machine-type: master
  state: Active 2
  strategy:
    type: RollingUpdate 3
  template:
    machineType: machines_v1beta1_machine_openshift_io
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: <platform> 4
        <platform_failure_domains> 5
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <cluster_id> 6
        machine.openshift.io/cluster-api-machine-role: master
        machine.openshift.io/cluster-api-machine-type: master
  spec:
    providerSpec:
      value:
        <platform_provider_spec> 7
```

- 1 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. You must specify this value when you create a **ControlPlaneMachineSet** CR. If you have the OpenShift CLI (**oc**) installed, you can obtain the infrastructure ID by running the following command:

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

- 2 Specify the state of the Operator. When the state is **Inactive**, the Operator is not operational. You can activate the Operator by setting the value to **Active**.



### IMPORTANT

Before you activate the CR, you must ensure that its configuration is correct for your cluster requirements.

- 3 Specify the update strategy for the cluster. Valid values are **OnDelete** and **RollingUpdate**. The default value is **RollingUpdate**. For more information about update strategies, see "Updating the control plane configuration".
- 4 Specify your cloud provider platform name. Valid values are **AWS**, **Azure**, **GCP**, **Nutanix**, **VSphere**, and **OpenStack**.
- 5 Add the **<platform\_failure\_domains>** configuration for the cluster. The format and values of this section are provider-specific. For more information, see the sample failure domain configuration for your cloud provider.
- 6 Specify the infrastructure ID.
- 7 Add the **<platform\_provider\_spec>** configuration for the cluster. The format and values of this section are provider-specific. For more information, see the sample provider specification for your cloud provider.

- 2 Refer to the sample YAML for a control plane machine set CR and populate your file with values that are appropriate for your cluster configuration.
- 3 Refer to the sample failure domain configuration and sample provider specification for your cloud provider and update those sections of your file with the appropriate values.
- 4 When the configuration is correct, activate the CR by setting the **.spec.state** field to **Active** and saving your changes.
- 5 Create the CR from your YAML file by running the following command:

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

where **<control\_plane\_machine\_set>** is the name of the YAML file that contains the CR configuration.

#### Additional resources

- [Updating the control plane configuration](#)
- [Control plane machine set configuration](#)
- [Provider-specific configuration options](#)

### 12.3. MANAGING CONTROL PLANE MACHINES WITH CONTROL PLANE MACHINE SETS

Control plane machine sets automate several essential aspects of control plane management.

### 12.3.1. Updating the control plane configuration

You can make changes to the configuration of the machines in the control plane by updating the specification in the control plane machine set custom resource (CR).

The Control Plane Machine Set Operator monitors the control plane machines and compares their configuration with the specification in the control plane machine set CR. When there is a discrepancy between the specification in the CR and the configuration of a control plane machine, the Operator marks that control plane machine for replacement.



#### NOTE

For more information about the parameters in the CR, see "Control plane machine set configuration".

#### Prerequisites

- Your cluster has an activated and functioning Control Plane Machine Set Operator.

#### Procedure

1. Edit your control plane machine set CR by running the following command:

```
$ oc edit controlplanemachinecluster.machine.openshift.io cluster \
-n openshift-machine-api
```

2. Change the values of any fields that you want to update in your cluster configuration.
3. Save your changes.

#### Next steps

- For clusters that use the default **RollingUpdate** update strategy, the control plane machine set propagates changes to your control plane configuration automatically.
- For clusters that are configured to use the **OnDelete** update strategy, you must replace your control plane machines manually.

#### 12.3.1.1. Automatic updates to the control plane configuration

The **RollingUpdate** update strategy automatically propagates changes to your control plane configuration. This update strategy is the default configuration for the control plane machine set.

For clusters that use the **RollingUpdate** update strategy, the Operator creates a replacement control plane machine with the configuration that is specified in the CR. When the replacement control plane machine is ready, the Operator deletes the control plane machine that is marked for replacement. The replacement machine then joins the control plane.

If multiple control plane machines are marked for replacement, the Operator protects etcd health during replacement by repeating this replacement process one machine at a time until it has replaced each machine.

#### 12.3.1.2. Manual updates to the control plane configuration

You can use the **OnDelete** update strategy to propagate changes to your control plane configuration by replacing machines manually. Manually replacing machines allows you to test changes to your configuration on a single machine before applying the changes more broadly.

For clusters that are configured to use the **OnDelete** update strategy, the Operator creates a replacement control plane machine when you delete an existing machine. When the replacement control plane machine is ready, the etcd Operator allows the existing machine to be deleted. The replacement machine then joins the control plane.

If multiple control plane machines are deleted, the Operator creates all of the required replacement machines simultaneously. The Operator maintains etcd health by preventing more than one machine being removed from the control plane at once.

### 12.3.2. Replacing a control plane machine

To replace a control plane machine in a cluster that has a control plane machine set, you delete the machine manually. The control plane machine set replaces the deleted machine with one using the specification in the control plane machine set custom resource (CR).

#### Prerequisites

- If your cluster runs on Red Hat OpenStack Platform (RHOSP) and you need to evacuate a compute server, such as for an upgrade, you must disable the RHOSP compute node that the machine runs on by running the following command:

```
$ openstack compute service set <target_node_host_name> nova-compute --disable
```

For more information, see [Preparing to migrate](#) in the RHOSP documentation.

#### Procedure

1. List the control plane machines in your cluster by running the following command:

```
$ oc get machines \
-l machine.openshift.io/cluster-api-machine-role==master \
-n openshift-machine-api
```

2. Delete a control plane machine by running the following command:

```
$ oc delete machine \
-n openshift-machine-api \
<control_plane_machine_name> ①
```

- ① Specify the name of the control plane machine to delete.

**NOTE**

If you delete multiple control plane machines, the control plane machine set replaces them according to the configured update strategy:

- For clusters that use the default **RollingUpdate** update strategy, the Operator replaces one machine at a time until each machine is replaced.
- For clusters that are configured to use the **OnDelete** update strategy, the Operator creates all of the required replacement machines simultaneously.

Both strategies maintain etcd health during control plane machine replacement.

### 12.3.3. Additional resources

- [Control plane machine set configuration](#)
- [Provider-specific configuration options](#)

## 12.4. CONTROL PLANE MACHINE SET CONFIGURATION

This example YAML snippet shows the base structure for a control plane machine set custom resource (CR).

### 12.4.1. Sample YAML for a control plane machine set custom resource

The base of the **ControlPlaneMachineSet** CR is structured the same way for all platforms.

#### Sample ControlPlaneMachineSet CR YAML file

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster ①
  namespace: openshift-machine-api
spec:
  replicas: 3 ②
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <cluster_id> ③
      machine.openshift.io/cluster-api-machine-role: master
      machine.openshift.io/cluster-api-machine-type: master
  state: Active ④
  strategy:
    type: RollingUpdate ⑤
  template:
    machineType: machines_v1beta1_machine_openshift_io
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: <platform> ⑥
        <platform_failure_domains> ⑦
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <cluster_id>
```

```

machine.openshift.io/cluster-api-machine-role: master
machine.openshift.io/cluster-api-machine-type: master
spec:
  providerSpec:
    value:
      <platform_provider_spec> ⑧

```

- ① Specifies the name of the **ControlPlaneMachineSet** CR, which is **cluster**. Do not change this value.
- ② Specifies the number of control plane machines. Only clusters with three control plane machines are supported, so the **replicas** value is **3**. Horizontal scaling is not supported. Do not change this value.
- ③ Specifies the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. You must specify this value when you create a **ControlPlaneMachineSet** CR. If you have the OpenShift CLI (**oc**) installed, you can obtain the infrastructure ID by running the following command:

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

- ④ Specifies the state of the Operator. When the state is **Inactive**, the Operator is not operational. You can activate the Operator by setting the value to **Active**.



### IMPORTANT

Before you activate the Operator, you must ensure that the **ControlPlaneMachineSet** CR configuration is correct for your cluster requirements. For more information about activating the Control Plane Machine Set Operator, see "Getting started with control plane machine sets".

- ⑤ Specifies the update strategy for the cluster. The allowed values are **OnDelete** and **RollingUpdate**. The default value is **RollingUpdate**. For more information about update strategies, see "Updating the control plane configuration".
- ⑥ Specifies the cloud provider platform name. Do not change this value.
- ⑦ Specifies the **<platform\_failure\_domains>** configuration for the cluster. The format and values of this section are provider-specific. For more information, see the sample failure domain configuration for your cloud provider.
- ⑧ Specifies the **<platform\_provider\_spec>** configuration for the cluster. The format and values of this section are provider-specific. For more information, see the sample provider specification for your cloud provider.

### Additional resources

- [Getting started with control plane machine sets](#)
- [Updating the control plane configuration](#)

### 12.4.2. Provider-specific configuration options

The `<platform_provider_spec>` and `<platform_failure_domains>` sections of the control plane machine set manifests are provider specific. For provider-specific configuration options for your cluster, see the following resources:

- Control plane configuration options for Amazon Web Services
- Control plane configuration options for Google Cloud Platform
- Control plane configuration options for Microsoft Azure
- Control plane configuration options for Nutanix
- Control plane configuration options for Red Hat OpenStack Platform (RHOSP)
- Control plane configuration options for VMware vSphere

## 12.5. CONFIGURATION OPTIONS FOR CONTROL PLANE MACHINES

### 12.5.1. Control plane configuration options for Amazon Web Services

You can change the configuration of your Amazon Web Services (AWS) control plane machines and enable features by updating values in the control plane machine set. When you save an update to the control plane machine set, the Control Plane Machine Set Operator updates the control plane machines according to your configured [update strategy](#).

#### 12.5.1.1. Sample YAML for configuring Amazon Web Services clusters

The following example YAML snippets show provider specification and failure domain configurations for an AWS cluster.

##### 12.5.1.1.1. Sample AWS provider specification

When you create a control plane machine set for an existing cluster, the provider specification must match the **providerSpec** configuration in the control plane machine custom resource (CR) that is created by the installation program. You can omit any field that is set in the failure domain section of the CR.

In the following example, `<cluster_id>` is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

##### Sample AWS providerSpec values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
  # ...
  template:
    # ...
```

```

spec:
  providerSpec:
    value:
      ami:
        id: ami-<ami_id_string> ①
      apiVersion: machine.openshift.io/v1beta1
      blockDevices:
        - ebs: ②
          encrypted: true
          iops: 0
          kmsKey:
            arn: ""
          volumeSize: 120
          volumeType: gp3
      credentialsSecret:
        name: aws-cloud-credentials ③
      deviceIndex: 0
      iamInstanceProfile:
        id: <cluster_id>-master-profile ④
      instanceType: m6i.xlarge ⑤
      kind: AWSMachineProviderConfig ⑥
      loadBalancers: ⑦
        - name: <cluster_id>-int
          type: network
        - name: <cluster_id>-ext
          type: network
      metadata:
        creationTimestamp: null
      metadataServiceOptions: {}
      placement: ⑧
        region: <region> ⑨
        availabilityZone: "" ⑩
        tenancy: ⑪
      securityGroups:
        - filters:
          - name: tag:Name
            values:
              - <cluster_id>-master-sg ⑫
      subnet: {} ⑬
      userDataSecret:
        name: master-user-data ⑭

```

- ① Specifies the Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Images (AMI) ID for the cluster. The AMI must belong to the same region as the cluster. If you want to use an AWS Marketplace image, you must complete the OpenShift Container Platform subscription from the [AWS Marketplace](#) to obtain an AMI ID for your region.
- ② Specifies the configuration of an encrypted EBS volume.
- ③ Specifies the secret name for the cluster. Do not change this value.
- ④ Specifies the AWS Identity and Access Management (IAM) instance profile. Do not change this value.

- 5 Specifies the AWS instance type for the control plane.
- 6 Specifies the cloud provider platform type. Do not change this value.
- 7 Specifies the internal (**int**) and external (**ext**) load balancers for the cluster.



#### NOTE

You can omit the external (**ext**) load balancer parameters on private OpenShift Container Platform clusters.

- 8 Specifies where to create the control plane instance in AWS.
- 9 Specifies the AWS region for the cluster.
- 10 This parameter is configured in the failure domain and is shown with an empty value here. If a value specified for this parameter differs from the value in the failure domain, the Control Plane Machine Set Operator overwrites it with the value in the failure domain.
- 11 Specifies the AWS Dedicated Instance configuration for the control plane. For more information, see AWS documentation about [Dedicated Instances](#). The following values are valid:
  - **default**: The Dedicated Instance runs on shared hardware.
  - **dedicated**: The Dedicated Instance runs on single-tenant hardware.
  - **host**: The Dedicated Instance runs on a Dedicated Host, which is an isolated server with configurations that you can control.
- 12 Specifies the control plane machines security group.
- 13 This parameter is configured in the failure domain and is shown with an empty value here. If a value specified for this parameter differs from the value in the failure domain, the Control Plane Machine Set Operator overwrites it with the value in the failure domain.



#### NOTE

If the failure domain configuration does not specify a value, the value in the provider specification is used. Configuring a subnet in the failure domain overwrites the subnet value in the provider specification.

- 14 Specifies the control plane user data secret. Do not change this value.

#### 12.5.1.1.2. Sample AWS failure domain configuration

The control plane machine set concept of a failure domain is analogous to existing AWS concept of an [Availability Zone \(AZ\)](#). The **ControlPlaneMachineSet** CR spreads control plane machines across multiple failure domains when possible.

When configuring AWS failure domains in the control plane machine set, you must specify the availability zone name and the subnet to use.

#### Sample AWS failure domain values

```

apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        aws:
          - placement:
              availabilityZone: <aws_zone_a> ①
            subnet: ②
              filters:
                - name: tag:Name
                  values:
                    - <cluster_id>-private-<aws_zone_a> ③
            type: Filters ④
          - placement:
              availabilityZone: <aws_zone_b> ⑤
            subnet:
              filters:
                - name: tag:Name
                  values:
                    - <cluster_id>-private-<aws_zone_b> ⑥
            type: Filters
        platform: AWS ⑦
# ...

```

- ① Specifies an AWS availability zone for the first failure domain.
- ② Specifies a subnet configuration. In this example, the subnet type is **Filters**, so there is a **filters** stanza.
- ③ Specifies the subnet name for the first failure domain, using the infrastructure ID and the AWS availability zone.
- ④ Specifies the subnet type. The allowed values are: **ARN**, **Filters** and **ID**. The default value is **Filters**.
- ⑤ Specifies the subnet name for an additional failure domain, using the infrastructure ID and the AWS availability zone.
- ⑥ Specifies the cluster's infrastructure ID and the AWS availability zone for the additional failure domain.
- ⑦ Specifies the cloud provider platform name. Do not change this value.

### 12.5.1.2. Enabling Amazon Web Services features for control plane machines

You can enable features by updating values in the control plane machine set.

### 12.5.1.2.1. Restricting the API server to private

After you deploy a cluster to Amazon Web Services (AWS), you can reconfigure the API server to use only the private zone.

#### Prerequisites

- Install the OpenShift CLI (**oc**).
- Have access to the web console as a user with **admin** privileges.

#### Procedure

1. In the web portal or console for your cloud provider, take the following actions:
  - a. Locate and delete the appropriate load balancer component:
    - For AWS, delete the external load balancer. The API DNS entry in the private zone already points to the internal load balancer, which uses an identical configuration, so you do not need to modify the internal load balancer.
  - b. Delete the **api.\$clusternode.\$yourdomain** DNS entry in the public zone.
2. Remove the external load balancers by deleting the following indicated lines in the control plane machine set custom resource:

```
# ...
providerSpec:
  value:
# ...
loadBalancers:
  - name: lk4pj-ext ①
    type: network ②
  - name: lk4pj-int
    type: network
# ...
```

- 1 Delete the **name** value for the external load balancer, which ends in **-ext**.
- 2 Delete the **type** value for the external load balancer.

#### Additional resources

- [Configuring the Ingress Controller endpoint publishing scope to Internal](#)

### 12.5.1.2.2. Changing the Amazon Web Services instance type by using a control plane machine set

You can change the Amazon Web Services (AWS) instance type that your control plane machines use by updating the specification in the control plane machine set custom resource (CR).

#### Prerequisites

- Your AWS cluster uses a control plane machine set.

## Procedure

1. Edit the following line under the **providerSpec** field:

```
providerSpec:
  value:
    ...
  instanceType: <compatible_aws_instance_type> ①
```

- 1 Specify a larger AWS instance type with the same base as the previous selection. For example, you can change **m6i.xlarge** to **m6i.2xlarge** or **m6i.4xlarge**.

2. Save your changes.

### 12.5.1.2.3. Assigning machines to placement groups for Elastic Fabric Adapter instances by using machine sets

You can configure a machine set to deploy machines on [Elastic Fabric Adapter](#) (EFA) instances within an existing AWS placement group.

EFA instances do not require placement groups, and you can use placement groups for purposes other than configuring an EFA. This example uses both to demonstrate a configuration that can improve network performance for machines within the specified placement group.

## Prerequisites

- You created a placement group in the AWS console.



### NOTE

Ensure that the [rules and limitations](#) for the type of placement group that you create are compatible with your intended use case. The control plane machine set spreads the control plane machines across multiple failure domains when possible. To use placement groups for the control plane, you must use a placement group type that can span multiple Availability Zones.

## Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following lines under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          instanceType: <supported_instance_type> ①
          networkInterfaceType: EFA ②
          placement:
            availabilityZone: <zone> ③
```

```

region: <region> 4
placementGroupName: <placement_group> 5
placementGroupPartition: <placement_group_partition_number> 6
# ...

```

- 1** Specify an instance type that [supports EFAs](#).
- 2** Specify the **EFA** network interface type.
- 3** Specify the zone, for example, **us-east-1a**.
- 4** Specify the region, for example, **us-east-1**.
- 5** Specify the name of the existing AWS placement group to deploy machines in.
- 6** Optional: Specify the partition number of the existing AWS placement group to deploy machines in.

## Verification

- In the AWS console, find a machine that the machine set created and verify the following in the machine properties:
  - The placement group field has the value that you specified for the **placementGroupName** parameter in the machine set.
  - The partition number field has the value that you specified for the **placementGroupPartition** parameter in the machine set.
  - The interface type field indicates that it uses an EFA.

### 12.5.1.2.4. Machine set options for the Amazon EC2 Instance Metadata Service

You can use machine sets to create machines that use a specific version of the Amazon EC2 Instance Metadata Service (IMDS). Machine sets can create machines that allow the use of both IMDSv1 and [IMDSv2](#) or machines that require the use of IMDSv2.



#### NOTE

Using IMDSv2 is only supported on AWS clusters that were created with OpenShift Container Platform version 4.7 or later.



#### IMPORTANT

Before configuring a machine set to create machines that require IMDSv2, ensure that any workloads that interact with the AWS metadata service support IMDSv2.

### 12.5.1.2.4.1. Configuring IMDS by using machine sets

You can specify whether to require the use of IMDSv2 by adding or editing the value of **metadataServiceOptions.authentication** in the machine set YAML file for your machines.

## Prerequisites

- To use IMDSv2, your AWS cluster must have been created with OpenShift Container Platform version 4.7 or later.

## Procedure

- Add or edit the following lines under the **providerSpec** field:

```
providerSpec:
  value:
    metadataServiceOptions:
      authentication: Required ①
```

- ① To require IMDSv2, set the parameter value to **Required**. To allow the use of both IMDSv1 and IMDSv2, set the parameter value to **Optional**. If no value is specified, both IMDSv1 and IMDSv2 are allowed.

### 12.5.1.2.5. Machine sets that deploy machines as Dedicated Instances

You can create a machine set running on AWS that deploys machines as Dedicated Instances. Dedicated Instances run in a virtual private cloud (VPC) on hardware that is dedicated to a single customer. These Amazon EC2 instances are physically isolated at the host hardware level. The isolation of Dedicated Instances occurs even if the instances belong to different AWS accounts that are linked to a single payer account. However, other instances that are not dedicated can share hardware with Dedicated Instances if they belong to the same AWS account.

Instances with either public or dedicated tenancy are supported by the Machine API. Instances with public tenancy run on shared hardware. Public tenancy is the default tenancy. Instances with dedicated tenancy run on single-tenant hardware.

#### 12.5.1.2.5.1. Creating Dedicated Instances by using machine sets

You can run a machine that is backed by a Dedicated Instance by using Machine API integration. Set the **tenancy** field in your machine set YAML file to launch a Dedicated Instance on AWS.

## Procedure

- Specify a dedicated tenancy under the **providerSpec** field:

```
providerSpec:
  placement:
    tenancy: dedicated
```

### 12.5.2. Control plane configuration options for Microsoft Azure

You can change the configuration of your Microsoft Azure control plane machines and enable features by updating values in the control plane machine set. When you save an update to the control plane machine set, the Control Plane Machine Set Operator updates the control plane machines according to your configured [update strategy](#).

#### 12.5.2.1. Sample YAML for configuring Microsoft Azure clusters

The following example YAML snippets show provider specification and failure domain configurations for an Azure cluster.

### 12.5.2.1.1. Sample Azure provider specification

When you create a control plane machine set for an existing cluster, the provider specification must match the **providerSpec** configuration in the control plane **Machine** CR that is created by the installation program. You can omit any field that is set in the failure domain section of the CR.

In the following example, **<cluster\_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

#### Sample Azure providerSpec values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
  spec:
    providerSpec:
      value:
        acceleratedNetworking: true
        apiVersion: machine.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials ①
          namespace: openshift-machine-api
        diagnostics: {}
        image: ②
          offer: ""
          publisher: ""
          resourceId: /resourceGroups/<cluster_id>-
            rg/providers/Microsoft.Compute/galleries/gallery_<cluster_id>/images/<cluster_id>-
              gen2/versions/412.86.20220930 ③
          sku: ""
          version: ""
        internalLoadBalancer: <cluster_id>-internal ④
        kind: AzureMachineProviderSpec ⑤
        location: <region> ⑥
        managedIdentity: <cluster_id>-identity
        metadata:
          creationTimestamp: null
          name: <cluster_id>
        networkResourceGroup: <cluster_id>-rg
        osDisk: ⑦
          diskSettings: {}
          diskSizeGB: 1024
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
```

```

publicIP: false
publicLoadBalancer: <cluster_id> ⑧
resourceGroup: <cluster_id>-rg
subnet: <cluster_id>-master-subnet ⑨
userDataSecret:
  name: master-user-data ⑩
vmSize: Standard_D8s_v3
vnet: <cluster_id>-vnet
zone: "1" ⑪

```

- ① Specifies the secret name for the cluster. Do not change this value.
- ② Specifies the image details for your control plane machine set.
- ③ Specifies an image that is compatible with your instance type. The Hyper-V generation V2 images created by the installation program have a **-gen2** suffix, while V1 images have the same name without the suffix.
- ④ Specifies the internal load balancer for the control plane. This field might not be preconfigured but is required in both the **ControlPlaneMachineSet** and control plane **Machine** CRs.
- ⑤ Specifies the cloud provider platform type. Do not change this value.
- ⑥ Specifies the region to place control plane machines on.
- ⑦ Specifies the disk configuration for the control plane.
- ⑧ Specifies the public load balancer for the control plane.



#### NOTE

You can omit the **publicLoadBalancer** parameter on private OpenShift Container Platform clusters that have user-defined outbound routing.

- ⑨ Specifies the subnet for the control plane.
- ⑩ Specifies the control plane user data secret. Do not change this value.
- ⑪ Specifies the zone configuration for clusters that use a single zone for all failure domains.



#### NOTE

If the cluster is configured to use a different zone for each failure domain, this parameter is configured in the failure domain. If you specify this value in the provider specification when using different zones for each failure domain, the Control Plane Machine Set Operator ignores it.

#### 12.5.2.1.2. Sample Azure failure domain configuration

The control plane machine set concept of a failure domain is analogous to existing Azure concept of an [Azure availability zone](#). The **ControlPlaneMachineSet** CR spreads control plane machines across multiple failure domains when possible.

When configuring Azure failure domains in the control plane machine set, you must specify the availability zone name. An Azure cluster uses a single subnet that spans multiple zones.

## Sample Azure failure domain values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        azure:
          - zone: "1" ①
          - zone: "2"
          - zone: "3"
        platform: Azure ②
# ...
```

- ① Each instance of **zone** specifies an Azure availability zone for a failure domain.



### NOTE

If the cluster is configured to use a single zone for all failure domains, the **zone** parameter is configured in the provider specification instead of in the failure domain configuration.

- ② Specifies the cloud provider platform name. Do not change this value.

### 12.5.2.2. Enabling Microsoft Azure features for control plane machines

You can enable features by updating values in the control plane machine set.

#### 12.5.2.2.1. Restricting the API server to private

After you deploy a cluster to Amazon Web Services (AWS), you can reconfigure the API server to use only the private zone.

#### Prerequisites

- Install the OpenShift CLI (**oc**).
- Have access to the web console as a user with **admin** privileges.

#### Procedure

1. In the web portal or console for your cloud provider, take the following actions:
  - a. Locate and delete the appropriate load balancer component:

- b. Delete the **api.\$clustername.\$yourdomain** DNS entry in the public zone.
- 2. Remove the external load balancers by deleting the following indicated lines in the control plane machine set custom resource:

```
# ...
providerSpec:
  value:
# ...
loadBalancers:
  - name: lk4pj-ext ①
    type: network ②
  - name: lk4pj-int
    type: network
# ...
```

- ① Delete the **name** value for the external load balancer, which ends in **-ext**.
- ② Delete the **type** value for the external load balancer.

## Additional resources

- [Configuring the Ingress Controller endpoint publishing scope to Internal](#)

### 12.5.2.2.2. Using the Azure Marketplace offering

You can create a machine set running on Azure that deploys machines that use the Azure Marketplace offering. To use this offering, you must first obtain the Azure Marketplace image. When obtaining your image, consider the following:

- While the images are the same, the Azure Marketplace publisher is different depending on your region. If you are located in North America, specify **redhat** as the publisher. If you are located in EMEA, specify **redhat-limited** as the publisher.
- The offer includes a **rh-ocp-worker** SKU and a **rh-ocp-worker-gen1** SKU. The **rh-ocp-worker** SKU represents a Hyper-V generation version 2 VM image. The default instance types used in OpenShift Container Platform are version 2 compatible. If you plan to use an instance type that is only version 1 compatible, use the image associated with the **rh-ocp-worker-gen1** SKU. The **rh-ocp-worker-gen1** SKU represents a Hyper-V version 1 VM image.



#### IMPORTANT

Installing images with the Azure marketplace is not supported on clusters with 64-bit ARM instances.

## Prerequisites

- You have installed the Azure CLI client (**az**).
- Your Azure account is entitled for the offer and you have logged into this account with the Azure CLI client.

## Procedure

- Display all of the available OpenShift Container Platform images by running one of the following commands:

- North America:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
```

#### Example output

Offer	Publisher	Sku	Urn	Version
rh-ocp-worker	RedHat	rh-ocp-worker	RedHat:rh-ocp-worker:rh-ocp-worker:4.15.2024072409	4.15.2024072409
rh-ocp-worker	RedHat	rh-ocp-worker-gen1	RedHat:rh-ocp-worker:rh-ocp-worker-gen1:4.15.2024072409	4.15.2024072409

- EMEA:

```
$ az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
```

#### Example output

Offer	Publisher	Sku	Urn
rh-ocp-worker	redhat-limited	rh-ocp-worker	redhat-limited:rh-ocp-worker:rh-ocp-worker:4.15.2024072409
Version			
rh-ocp-worker	redhat-limited	rh-ocp-worker-gen1	redhat-limited:rh-ocp-worker:rh-ocp-worker-gen1:4.15.2024072409



#### NOTE

Use the latest image that is available for compute and control plane nodes. If required, your VMs are automatically upgraded as part of the installation process.

- Inspect the image for your offer by running one of the following commands:

- North America:

```
$ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

- Review the terms of the offer by running one of the following commands:

- North America:

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

- Accept the terms of the offering by running one of the following commands:

- North America:

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- EMEA:

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

- Record the image details of your offer, specifically the values for **publisher**, **offer**, **sku**, and **version**.

- Add the following parameters to the **providerSpec** section of your machine set YAML file using the image details for your offer:

#### Sample providerSpec image values for Azure Marketplace machines

```
providerSpec:
  value:
    image:
      offer: rh-ocp-worker
      publisher: redhat
      resourceId: ""
      sku: rh-ocp-worker
      type: MarketplaceWithPlan
      version: 413.92.2023101700
```

##### 12.5.2.2.3. Enabling Azure boot diagnostics

You can enable boot diagnostics on Azure machines that your machine set creates.

#### Prerequisites

- Have an existing Microsoft Azure cluster.

#### Procedure

- Add the **diagnostics** configuration that is applicable to your storage type to the **providerSpec** field in your machine set YAML file:
  - For an Azure Managed storage account:

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged ①
```

① Specifies an Azure Managed storage account.

- For an Azure Unmanaged storage account:

```
providerSpec:  
diagnostics:  
boot:  
  storageAccountType: CustomerManaged ①  
  customerManaged:  
    storageAccountURI: https://<storage-account>.blob.core.windows.net ②
```

- ① Specifies an Azure Unmanaged storage account.
- ② Replace **<storage-account>** with the name of your storage account.



#### NOTE

Only the Azure Blob Storage data service is supported.

### Verification

- On the Microsoft Azure portal, review the **Boot diagnostics** page for a machine deployed by the machine set, and verify that you can see the serial logs for the machine.

#### 12.5.2.2.4. Machine sets that deploy machines with ultra disks as data disks

You can create a machine set running on Azure that deploys machines with ultra disks. Ultra disks are high-performance storage that are intended for use with the most demanding data workloads.

### Additional resources

- [Microsoft Azure ultra disks documentation](#)

#### 12.5.2.2.4.1. Creating machines with ultra disks by using machine sets

You can deploy machines with ultra disks on Azure by editing your machine set YAML file.

### Prerequisites

- Have an existing Microsoft Azure cluster.

### Procedure

- Create a custom secret in the **openshift-machine-api** namespace using the **master** data secret by running the following command:

```
$ oc -n openshift-machine-api \  
get secret <role>-user-data \ ①  
--template='{{index .data.userData | base64decode}}' | jq > userData.txt ②
```

- ① Replace **<role>** with **master**.
- ② Specify **userData.txt** as the name of the new custom secret.

2. In a text editor, open the **userData.txt** file and locate the final } character in the file.

- On the immediately preceding line, add a ,.
- Create a new line after the , and add the following configuration details:

```

"storage": {
  "disks": [ 1
    {
      "device": "/dev/disk/azure/scsi1/lun0", 2
      "partitions": [ 3
        {
          "label": "lun0p1", 4
          "sizeMiB": 1024, 5
          "startMiB": 0
        }
      ]
    }
  ],
  "filesystems": [ 6
    {
      "device": "/dev/disk/by-partlabel/lun0p1",
      "format": "xfs",
      "path": "/var/lib/lun0p1"
    }
  ],
  "systemd": {
    "units": [ 7
      {
        "contents": "[Unit]\nBefore=local-
fs.target\n[Mount]\nWhere=/var/lib/lun0p1\nWhat=/dev/disk/by-
partlabel/lun0p1\nOptions=defaults,pquota\n[Install]\nWantedBy=local-fs.target\n", 8
        "enabled": true,
        "name": "var-lib-lun0p1.mount"
      }
    ]
  }
}

```

- The configuration details for the disk that you want to attach to a node as an ultra disk.
- Specify the **lun** value that is defined in the **dataDisks** stanza of the machine set you are using. For example, if the machine set contains **lun: 0**, specify **lun0**. You can initialize multiple data disks by specifying multiple "**disks**" entries in this configuration file. If you specify multiple "**disks**" entries, ensure that the **lun** value for each matches the value in the machine set.
- The configuration details for a new partition on the disk.
- Specify a label for the partition. You might find it helpful to use hierarchical names, such as **lun0p1** for the first partition of **lun0**.
- Specify the total size in MiB of the partition.
-

Specify the filesystem to use when formatting a partition. Use the partition label to specify the partition.

- 7 Specify a **systemd** unit to mount the partition at boot. Use the partition label to specify the partition. You can create multiple partitions by specifying multiple "**partitions**" entries in this configuration file. If you specify multiple "**partitions**" entries, you must specify a **systemd** unit for each.
- 8 For **Where**, specify the value of **storage.filesystems.path**. For **What**, specify the value of **storage.filesystems.device**.

3. Extract the disabling template value to a file called **disableTemplating.txt** by running the following command:

```
$ oc -n openshift-machine-api get secret <role>-user-data \ ①
--template='{index .data.disableTemplating | base64decode}' | jq > disableTemplating.txt
```

- 1 Replace **<role>** with **master**.

4. Combine the **userData.txt** file and **disableTemplating.txt** file to create a data secret file by running the following command:

```
$ oc -n openshift-machine-api create secret generic <role>-user-data-x5 \ ①
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt
```

- 1 For **<role>-user-data-x5**, specify the name of the secret. Replace **<role>** with **master**.

5. Edit your control plane machine set CR by running the following command:

```
$ oc --namespace openshift-machine-api edit controlplanemachineset.machine.openshift.io
cluster
```

6. Add the following lines in the positions indicated:

```
apiVersion: machine.openshift.io/v1beta1
kind: ControlPlaneMachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd ①
      providerSpec:
        value:
          ultraSSDCapability: Enabled ②
        dataDisks: ③
          - nameSuffix: ultrassd
            lun: ④
            diskSizeGB: ④
            deletionPolicy: Delete
            cachingType: None
```

```

managedDisk:
  storageAccountType: UltraSSD_LRS
userDataSecret:
  name: <role>-user-data-x5 ④

```

- ① Specify a label to use to select a node that is created by this machine set. This procedure uses **disk.ultrassd** for this value.
- ② ③ These lines enable the use of ultra disks. For **dataDisks**, include the entire stanza.
- ④ Specify the user data secret created earlier. Replace **<role>** with **master**.

## 7. Save your changes.

- For clusters that use the default **RollingUpdate** update strategy, the Operator automatically propagates the changes to your control plane configuration.
- For clusters that are configured to use the **OnDelete** update strategy, you must replace your control plane machines manually.

## Verification

1. Validate that the machines are created by running the following command:

```
$ oc get machines
```

The machines should be in the **Running** state.

2. For a machine that is running and has a node attached, validate the partition by running the following command:

```
$ oc debug node/<node-name> -- chroot /host lsblk
```

In this command, **oc debug node/<node-name>** starts a debugging shell on the node **<node-name>** and passes a command with **--**. The passed command **chroot /host** provides access to the underlying host OS binaries, and **lsblk** shows the block devices that are attached to the host OS machine.

## Next steps

- To use an ultra disk on the control plane, reconfigure your workload to use the control plane's ultra disk mount point.

### 12.5.2.2.4.2. Troubleshooting resources for machine sets that enable ultra disks

Use the information in this section to understand and recover from issues you might encounter.

#### 12.5.2.2.4.2.1. Incorrect ultra disk configuration

If an incorrect configuration of the **ultraSSDCapability** parameter is specified in the machine set, the machine provisioning fails.

For example, if the **ultraSSDCapability** parameter is set to **Disabled**, but an ultra disk is specified in the **dataDisks** parameter, the following error message appears:

StorageAccountType UltraSSD\_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.

- To resolve this issue, verify that your machine set configuration is correct.

#### 12.5.2.2.4.2.2. Unsupported disk parameters

If a region, availability zone, or instance size that is not compatible with ultra disks is specified in the machine set, the machine provisioning fails. Check the logs for the following error message:

failed to create vm <machine\_name>: failure sending request for machine <machine\_name>: cannot create vm: compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request: StatusCode=400 -- Original Error: Code="BadRequest" Message="Storage Account type 'UltraSSD\_LRS' is not supported <more\_information\_about\_why>."

- To resolve this issue, verify that you are using this feature in a supported environment and that your machine set configuration is correct.

#### 12.5.2.2.4.2.3. Unable to delete disks

If the deletion of ultra disks as data disks is not working as expected, the machines are deleted and the data disks are orphaned. You must delete the orphaned disks manually if desired.

#### 12.5.2.2.5. Enabling customer-managed encryption keys for a machine set

You can supply an encryption key to Azure to encrypt data on managed disks at rest. You can enable server-side encryption with customer-managed keys by using the Machine API.

An Azure Key Vault, a disk encryption set, and an encryption key are required to use a customer-managed key. The disk encryption set must be in a resource group where the Cloud Credential Operator (CCO) has granted permissions. If not, an additional reader role is required to be granted on the disk encryption set.

#### Prerequisites

- [Create an Azure Key Vault instance](#) .
- [Create an instance of a disk encryption set](#) .
- [Grant the disk encryption set access to key vault](#) .

#### Procedure

- Configure the disk encryption set under the **providerSpec** field in your machine set YAML file. For example:

```
providerSpec:
  value:
    osDisk:
      diskSizeGB: 128
    managedDisk:
      diskEncryptionSet:
        id:
```

```
/subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.Compute/diskEncryptionSets/<disk_encryption_set_name>
storageAccountType: Premium_LRS
```

## Additional resources

- [Azure documentation about customer-managed keys](#)

### 12.5.2.2.6. Configuring trusted launch for Azure virtual machines by using machine sets



#### IMPORTANT

Using trusted launch for Azure virtual machines is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

OpenShift Container Platform 4.17 supports trusted launch for Azure virtual machines (VMs). By editing the machine set YAML file, you can configure the trusted launch options that a machine set uses for machines that it deploys. For example, you can configure these machines to use UEFI security features such as Secure Boot or a dedicated virtual Trusted Platform Module (vTPM) instance.



#### NOTE

Some feature combinations result in an invalid configuration.

**Table 12.2. UEFI feature combination compatibility**

Secure Boot[1]	vTPM[2]	Valid configuration
Enabled	Enabled	Yes
Enabled	Disabled	Yes
Enabled	Omitted	Yes
Disabled	Enabled	Yes
Omitted	Enabled	Yes
Disabled	Disabled	No
Omitted	Disabled	No
Omitted	Omitted	No

1. Using the **secureBoot** field.
2. Using the **virtualizedTrustedPlatformModule** field.

For more information about related features and functionality, see the Microsoft Azure documentation about [Trusted launch for Azure virtual machines](#).

## Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following section under the **providerSpec** field to provide a valid configuration:

### Sample valid configuration with UEFI Secure Boot and vTPM enabled

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    machines_v1beta1_machine.openshift.io:
      spec:
        providerSpec:
          value:
            securityProfile:
              settings:
                securityType: TrustedLaunch ①
                trustedLaunch:
                  uefiSettings: ②
                    secureBoot: Enabled ③
                    virtualizedTrustedPlatformModule: Enabled ④
# ...
```

- ① Enables the use of trusted launch for Azure virtual machines. This value is required for all valid configurations.
- ② Specifies which UEFI security features to use. This section is required for all valid configurations.
- ③ Enables UEFI Secure Boot.
- ④ Enables the use of a vTPM.

## Verification

- On the Azure portal, review the details for a machine deployed by the machine set and verify that the trusted launch options match the values that you configured.

### 12.5.2.2.7. Configuring Azure confidential virtual machines by using machine sets



## IMPORTANT

Using Azure confidential virtual machines is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

OpenShift Container Platform 4.17 supports Azure confidential virtual machines (VMs).



## NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

By editing the machine set YAML file, you can configure the confidential VM options that a machine set uses for machines that it deploys. For example, you can configure these machines to use UEFI security features such as Secure Boot or a dedicated virtual Trusted Platform Module (vTPM) instance.



## WARNING

Not all instance types support confidential VMs. Do not change the instance type for a control plane machine set that is configured to use confidential VMs to a type that is incompatible. Using an incompatible instance type can cause your cluster to become unstable.

For more information about related features and functionality, see the Microsoft Azure documentation about [Confidential virtual machines](#).

## Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following section under the **providerSpec** field:

### Sample configuration

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          osDisk:
```

```

# ...
managedDisk:
  securityProfile: ①
    securityEncryptionType: VMGuestStateOnly ②
  # ...
  securityProfile: ③
  settings:
    securityType: ConfidentialVM ④
    confidentialVM:
      uefiSettings: ⑤
        secureBoot: Disabled ⑥
        virtualizedTrustedPlatformModule: Enabled ⑦
    vmSize: Standard_DC16ads_v5 ⑧
# ...

```

- ① Specifies security profile settings for the managed disk when using a confidential VM.
- ② Enables encryption of the Azure VM Guest State (VMGS) blob. This setting requires the use of vTPM.
- ③ Specifies security profile settings for the confidential VM.
- ④ Enables the use of confidential VMs. This value is required for all valid configurations.
- ⑤ Specifies which UEFI security features to use. This section is required for all valid configurations.
- ⑥ Disables UEFI Secure Boot.
- ⑦ Enables the use of a vTPM.
- ⑧ Specifies an instance type that supports confidential VMs.

## Verification

- On the Azure portal, review the details for a machine deployed by the machine set and verify that the confidential VM options match the values that you configured.

### 12.5.2.2.8. Accelerated Networking for Microsoft Azure VMs

Accelerated Networking uses single root I/O virtualization (SR-IOV) to provide Microsoft Azure VMs with a more direct path to the switch. This enhances network performance. This feature can be enabled after installation.

#### 12.5.2.2.8.1. Limitations

Consider the following limitations when deciding whether to use Accelerated Networking:

- Accelerated Networking is only supported on clusters where the Machine API is operational.
- Accelerated Networking requires an Azure VM size that includes at least four vCPUs. To satisfy this requirement, you can change the value of **vmSize** in your machine set. For information about Azure VM sizes, see [Microsoft Azure documentation](#).

### 12.5.2.2.9. Configuring Capacity Reservation by using machine sets

OpenShift Container Platform version 4.17 and later supports on-demand Capacity Reservation with Capacity Reservation groups on Microsoft Azure clusters.

You can configure a machine set to deploy machines on any available resources that match the parameters of a capacity request that you define. These parameters specify the VM size, region, and number of instances that you want to reserve. If your Azure subscription quota can accommodate the capacity request, the deployment succeeds.

For more information, including limitations and suggested use cases for this Azure instance type, see the Microsoft Azure documentation about [On-demand Capacity Reservation](#).



#### NOTE

You cannot change an existing Capacity Reservation configuration for a machine set. To use a different Capacity Reservation group, you must replace the machine set and the machines that the previous machine set deployed.

#### Prerequisites

- You have access to the cluster with **cluster-admin** privileges.
- You installed the OpenShift CLI (**oc**).
- You created a Capacity Reservation group.

For more information, see the Microsoft Azure documentation [Create a Capacity Reservation](#).

#### Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following section under the **providerSpec** field:

#### Sample configuration

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    machines_v1beta1_machine_openshift_io:
      spec:
        providerSpec:
          value:
            capacityReservationGroupID: <capacity_reservation_group> ①
# ...
```

- 1 Specify the ID of the Capacity Reservation group that you want the machine set to deploy machines on.

#### Verification

▲ This section is part of the documentation for the OpenShift Container Platform 4.17 release, which is no longer in active development. It is provided for informational purposes only.

- To verify machine deployment, list the machines that the machine set created by running the following command:

```
$ oc get machine \
-n openshift-machine-api \
-l machine.openshift.io/cluster-api-machine-role=master
```

In the output, verify that the characteristics of the listed machines match the parameters of your Capacity Reservation.

### 12.5.2.2.9.1. Enabling Accelerated Networking on an existing Microsoft Azure cluster

You can enable Accelerated Networking on Azure by adding **acceleratedNetworking** to your machine set YAML file.

#### Prerequisites

- Have an existing Microsoft Azure cluster where the Machine API is operational.

#### Procedure

- Add the following to the **providerSpec** field:

```
providerSpec:
  value:
    acceleratedNetworking: true ①
    vmSize: <azure-vm-size> ②
```

- ① This line enables Accelerated Networking.
- ② Specify an Azure VM size that includes at least four vCPUs. For information about VM sizes, see [Microsoft Azure documentation](#).

#### Verification

- On the Microsoft Azure portal, review the **Networking** settings page for a machine provisioned by the machine set, and verify that the **Accelerated networking** field is set to **Enabled**.

### 12.5.3. Control plane configuration options for Google Cloud Platform

You can change the configuration of your Google Cloud Platform (GCP) control plane machines and enable features by updating values in the control plane machine set. When you save an update to the control plane machine set, the Control Plane Machine Set Operator updates the control plane machines according to your configured [update strategy](#).

#### 12.5.3.1. Sample YAML for configuring Google Cloud Platform clusters

The following example YAML snippets show provider specification and failure domain configurations for a GCP cluster.

##### 12.5.3.1.1. Sample GCP provider specification

When you create a control plane machine set for an existing cluster, the provider specification must

match the **providerSpec** configuration in the control plane machine custom resource (CR) that is created by the installation program. You can omit any field that is set in the failure domain section of the CR.

### Values obtained by using the OpenShift CLI

In the following example, you can obtain some of the values for your cluster by using the OpenShift CLI.

#### Infrastructure ID

The **<cluster\_id>** string is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

#### Image path

The **<path\_to\_image>** string is the path to the image that was used to create the disk. If you have the OpenShift CLI installed, you can obtain the path to the image by running the following command:

```
$ oc -n openshift-machine-api \
-o
jsonpath='{.spec.template.machines_v1beta1_machine.openshift_io.spec.providerSpec.value.disks[0].image}{"\n"}' \
get ControlPlaneMachineSet/cluster
```

### Sample GCP providerSpec values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
  spec:
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1beta1
        canIPForward: false
        credentialsSecret:
          name: gcp-cloud-credentials ①
        deletionProtection: false
        disks:
          - autoDelete: true
            boot: true
            image: <path_to_image> ②
            labels: null
            sizeGb: 200
            type: pd-ssd
        kind: GCPMachineProviderSpec ③
        machineType: e2-standard-4
        metadata:
```

```

creationTimestamp: null
metadataServiceOptions: {}
networkInterfaces:
- network: <cluster_id>-network
  subnetwork: <cluster_id>-master-subnet
projectID: <project_name> ④
region: <region> ⑤
serviceAccounts: ⑥
- email: <cluster_id>-m@<project_name>.iam.gserviceaccount.com
  scopes:
    - https://www.googleapis.com/auth/cloud-platform
shieldedInstanceConfig: {}
tags:
- <cluster_id>-master
targetPools:
- <cluster_id>-api
userDataSecret:
  name: master-user-data ⑦
zone: "" ⑧

```

- ① Specifies the secret name for the cluster. Do not change this value.
- ② Specifies the path to the image that was used to create the disk.

To use a GCP Marketplace image, specify the offer to use:

- OpenShift Container Platform:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736>
- OpenShift Kubernetes Engine:  
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736>

- ③ Specifies the cloud provider platform type. Do not change this value.
- ④ Specifies the name of the GCP project that you use for your cluster.
- ⑤ Specifies the GCP region for the cluster.
- ⑥ Specifies a single service account. Multiple service accounts are not supported.
- ⑦ Specifies the control plane user data secret. Do not change this value.
- ⑧ This parameter is configured in the failure domain, and is shown with an empty value here. If a value specified for this parameter differs from the value in the failure domain, the Operator overwrites it with the value in the failure domain.

#### 12.5.3.1.2. Sample GCP failure domain configuration

The control plane machine set concept of a failure domain is analogous to the existing GCP concept of a [zone](#). The **ControlPlaneMachineSet** CR spreads control plane machines across multiple failure domains when possible.

When configuring GCP failure domains in the control plane machine set, you must specify the zone name to use.

### Sample GCP failure domain values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        gcp:
          - zone: <gcp_zone_a> ①
          - zone: <gcp_zone_b> ②
          - zone: <gcp_zone_c>
          - zone: <gcp_zone_d>
        platform: GCP ③
# ...
```

- ① Specifies a GCP zone for the first failure domain.
- ② Specifies an additional failure domain. Further failure domains are added the same way.
- ③ Specifies the cloud provider platform name. Do not change this value.

#### 12.5.3.2. Enabling Google Cloud Platform features for control plane machines

You can enable features by updating values in the control plane machine set.

##### 12.5.3.2.1. Configuring persistent disk types by using machine sets

You can configure the type of persistent disk that a machine set deploys machines on by editing the machine set YAML file.

For more information about persistent disk types, compatibility, regional availability, and limitations, see the GCP Compute Engine documentation about [persistent disks](#).

#### Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.
2. Edit the following line under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
```

```

...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            type: pd-ssd ①

```

- ① Control plane nodes must use the **pd-ssd** disk type.

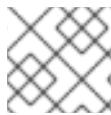
## Verification

- Using the Google Cloud console, review the details for a machine deployed by the machine set and verify that the **Type** field matches the configured disk type.

### 12.5.3.2.2. Configuring Confidential VM by using machine sets

By editing the machine set YAML file, you can configure the Confidential VM options that a machine set uses for machines that it deploys.

For more information about Confidential VM features, functions, and compatibility, see the GCP Compute Engine documentation about [Confidential VM](#).



#### NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.



#### IMPORTANT

OpenShift Container Platform 4.17 does not support some Confidential Compute features, such as Confidential VMs with AMD Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP).

## Procedure

- In a text editor, open the YAML file for an existing machine set or create a new one.
- Edit the following section under the **providerSpec** field:

```

apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          confidentialCompute: Enabled ①
          onHostMaintenance: Terminate ②
          machineType: n2d-standard-8 ③
...

```

- 1 Specify whether Confidential VM is enabled. Valid values are **Disabled** or **Enabled**.
- 2 Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VM does not support live VM migration.
- 3 Specify a machine type that supports Confidential VM. Confidential VM supports the N2D and C2D series of machine types.

## Verification

- On the Google Cloud console, review the details for a machine deployed by the machine set and verify that the Confidential VM options match the values that you configured.

### 12.5.3.2.3. Configuring Shielded VM options by using machine sets

By editing the machine set YAML file, you can configure the Shielded VM options that a machine set uses for machines that it deploys.

For more information about Shielded VM features and functionality, see the GCP Compute Engine documentation about [Shielded VM](#).

## Procedure

- 1 In a text editor, open the YAML file for an existing machine set or create a new one.
- 2 Edit the following section under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          shieldedInstanceStateConfig: 1
          integrityMonitoring: Enabled 2
          secureBoot: Disabled 3
          virtualizedTrustedPlatformModule: Enabled 4
# ...
```

- 1 In this section, specify any Shielded VM options that you want.
- 2 Specify whether integrity monitoring is enabled. Valid values are **Disabled** or **Enabled**.



#### NOTE

When integrity monitoring is enabled, you must not disable virtual trusted platform module (vTPM).

- 3 Specify whether UEFI Secure Boot is enabled. Valid values are **Disabled** or **Enabled**.

- 4** Specify whether vTPM is enabled. Valid values are **Disabled** or **Enabled**.

## Verification

- Using the Google Cloud console, review the details for a machine deployed by the machine set and verify that the Shielded VM options match the values that you configured.

## Additional resources

- [What is Shielded VM?](#)
  - [Secure Boot](#)
  - [Virtual Trusted Platform Module \(vTPM\)](#)
  - [Integrity monitoring](#)

### 12.5.3.2.4. Enabling customer-managed encryption keys for a machine set

Google Cloud Platform (GCP) Compute Engine allows users to supply an encryption key to encrypt data on disks at rest. The key is used to encrypt the data encryption key, not to encrypt the customer's data. By default, Compute Engine encrypts this data by using Compute Engine keys.

You can enable encryption with a customer-managed key in clusters that use the Machine API. You must first [create a KMS key](#) and assign the correct permissions to a service account. The KMS key name, key ring name, and location are required to allow a service account to use your key.



#### NOTE

If you do not want to use a dedicated service account for the KMS encryption, the Compute Engine default service account is used instead. You must grant the default service account permission to access the keys if you do not use a dedicated service account. The Compute Engine default service account name follows the **service-@compute-system.iam.gserviceaccount.com** pattern.

## Procedure

- To allow a specific service account to use your KMS key and to grant the service account the correct IAM role, run the following command with your KMS key name, key ring name, and location:

```
$ gcloud kms keys add-iam-policy-binding <key_name> \
--keyring <key_ring_name> \
--location <key_ring_location> \
--member "serviceAccount:service-<project_number>@compute-
system.iam.gserviceaccount.com" \
--role roles/cloudkms.cryptoKeyEncrypterDecrypter
```

- Configure the encryption key under the **providerSpec** field in your machine set YAML file. For example:

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
...
```

```

spec:
template:
spec:
providerSpec:
value:
disks:
- type:
  encryptionKey:
    kmsKey:
      name: machine-encryption-key 1
      keyRing: openshift-encrpytion-ring 2
      location: global 3
      projectID: openshift-gcp-project 4
      kmsKeyServiceAccount: openshift-service-account@openshift-gcp-
project.iam.gserviceaccount.com 5

```

- 1** The name of the customer-managed encryption key that is used for the disk encryption.
- 2** The name of the KMS key ring that the KMS key belongs to.
- 3** The GCP location in which the KMS key ring exists.
- 4** Optional: The ID of the project in which the KMS key ring exists. If a project ID is not set, the machine set **projectId** in which the machine set was created is used.
- 5** Optional: The service account that is used for the encryption request for the given KMS key. If a service account is not set, the Compute Engine default service account is used.

When a new machine is created by using the updated **providerSpec** object configuration, the disk encryption key is encrypted with the KMS key.

#### 12.5.4. Control plane configuration options for Nutanix

You can change the configuration of your Nutanix control plane machines by updating values in the control plane machine set. When you save an update to the control plane machine set, the Control Plane Machine Set Operator updates the control plane machines according to your configured [update strategy](#).

##### 12.5.4.1. Sample YAML for configuring Nutanix clusters

The following example YAML snippet shows a provider specification configuration for a Nutanix cluster.

###### 12.5.4.1.1. Sample Nutanix provider specification

When you create a control plane machine set for an existing cluster, the provider specification must match the **providerSpec** configuration in the control plane machine custom resource (CR) that is created by the installation program.

###### Values obtained by using the OpenShift CLI

In the following example, you can obtain some of the values for your cluster by using the OpenShift CLI.

###### Infrastructure ID

The **<cluster\_id>** string is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

## Sample Nutanix providerSpec values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
  spec:
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1
        bootType: "" 1
        categories: 2
          - key: <category_name>
            value: <category_value>
        cluster: 3
          type: uuid
          uuid: <cluster_uuid>
        credentialsSecret:
          name: nutanix-credentials 4
        image: 5
          name: <cluster_id>-rhcos
          type: name
        kind: NutanixMachineProviderConfig 6
        memorySize: 16Gi 7
        metadata:
          creationTimestamp: null
        project: 8
          type: name
          name: <project_name>
        subnets: 9
          - type: uuid
            uuid: <subnet_uuid>
        systemDiskSize: 120Gi 10
        userDataSecret:
          name: master-user-data 11
        vcpuSockets: 8 12
        vcpusPerSocket: 1 13
```

**1** Specifies the boot type that the control plane machines use. For more information about boot types, see [Understanding UEFI, Secure Boot, and TPM in the Virtualized Environment](#). Valid values are **Legacy**, **SecureBoot**, or **UEFI**. The default is **Legacy**.

**NOTE**

You must use the **Legacy** boot type in OpenShift Container Platform 4.17.

- 2** Specifies one or more Nutanix Prism categories to apply to control plane machines. This stanza requires **key** and **value** parameters for a category key-value pair that exists in Prism Central. For more information about categories, see [Category management](#).
- 3** Specifies a Nutanix Prism Element cluster configuration. In this example, the cluster type is **uuid**, so there is a **uuid** stanza.

**NOTE**

Clusters that use OpenShift Container Platform version 4.15 or later can use failure domain configurations.

If the cluster is configured to use a failure domain, this parameter is configured in the failure domain. If you specify this value in the provider specification when using failure domains, the Control Plane Machine Set Operator ignores it.

- 4** Specifies the secret name for the cluster. Do not change this value.
- 5** Specifies the image that was used to create the disk.
- 6** Specifies the cloud provider platform type. Do not change this value.
- 7** Specifies the memory allocated for the control plane machines.
- 8** Specifies the Nutanix project that you use for your cluster. In this example, the project type is **name**, so there is a **name** stanza.
- 9** Specifies a subnet configuration. In this example, the subnet type is **uuid**, so there is a **uuid** stanza.

**NOTE**

Clusters that use OpenShift Container Platform version 4.15 or later can use failure domain configurations.

If the cluster is configured to use a failure domain, this parameter is configured in the failure domain. If you specify this value in the provider specification when using failure domains, the Control Plane Machine Set Operator ignores it.

- 10** Specifies the VM disk size for the control plane machines.
- 11** Specifies the control plane user data secret. Do not change this value.
- 12** Specifies the number of vCPU sockets allocated for the control plane machines.
- 13** Specifies the number of vCPUs for each control plane vCPU socket.

#### 12.5.4.1.2. Failure domains for Nutanix clusters

To add or update the failure domain configuration on a Nutanix cluster, you must make coordinated changes to several resources. The following actions are required:

1. Modify the cluster infrastructure custom resource (CR).
2. Modify the cluster control plane machine set CR.
3. Modify or replace the compute machine set CRs.

For more information, see "Adding failure domains to an existing Nutanix cluster" in the *Post-installation configuration* content.

## Additional resources

- [Adding failure domains to an existing Nutanix cluster](#)

### 12.5.5. Control plane configuration options for Red Hat OpenStack Platform

You can change the configuration of your Red Hat OpenStack Platform (RHOSP) control plane machines and enable features by updating values in the control plane machine set. When you save an update to the control plane machine set, the Control Plane Machine Set Operator updates the control plane machines according to your configured [update strategy](#).

#### 12.5.5.1. Sample YAML for configuring Red Hat OpenStack Platform (RHOSP) clusters

The following example YAML snippets show provider specification and failure domain configurations for an RHOSP cluster.

##### 12.5.5.1.1. Sample RHOSP provider specification

When you create a control plane machine set for an existing cluster, the provider specification must match the **providerSpec** configuration in the control plane machine custom resource (CR) that is created by the installation program.

#### Sample OpenStack providerSpec values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
  spec:
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1alpha1
        cloudName: openstack
        cloudsSecret:
          name: openstack-cloud-credentials 1
          namespace: openshift-machine-api
        flavor: m1.xlarge 2
        image: ocp1-2g2xs-rhcos
        kind: OpenstackProviderSpec 3
      metadata:
```

```

creationTimestamp: null
networks:
- filter: {}
  subnets:
  - filter:
    name: ocp1-2g2xs-nodes
    tags: openshiftClusterID=ocp1-2g2xs
  securityGroups:
  - filter: {}
    name: ocp1-2g2xs-master 4
serverGroupName: ocp1-2g2xs-master
serverMetadata:
  Name: ocp1-2g2xs-master
  openshiftClusterID: ocp1-2g2xs
tags:
- openshiftClusterID=ocp1-2g2xs
trunk: true
userDataSecret:
  name: master-user-data

```

- 1** The secret name for the cluster. Do not change this value.
- 2** The RHOSP flavor type for the control plane.
- 3** The RHOSP cloud provider platform type. Do not change this value.
- 4** The control plane machines security group.

#### 12.5.5.1.2. Sample RHOSP failure domain configuration

The control plane machine set concept of a failure domain is analogous to the existing Red Hat OpenStack Platform (RHOSP) concept of an [availability zone](#). The **ControlPlaneMachineSet** CR spreads control plane machines across multiple failure domains when possible.

The following example demonstrates the use of multiple Nova availability zones as well as Cinder availability zones.

#### Sample OpenStack failure domain values

```

apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: OpenStack
        openstack:
        - availabilityZone: nova-az0
          rootVolume:
            availabilityZone: cinder-az0

```

```

- availabilityZone: nova-az1
  rootVolume:
    availabilityZone: cinder-az1
- availabilityZone: nova-az2
  rootVolume:
    availabilityZone: cinder-az2
# ...

```

### 12.5.5.2. Enabling Red Hat OpenStack Platform (RHOSP) features for control plane machines

You can enable features by updating values in the control plane machine set.

#### 12.5.5.2.1. Changing the RHOSP compute flavor by using a control plane machine set

You can change the Red Hat OpenStack Platform (RHOSP) compute service (Nova) flavor that your control plane machines use by updating the specification in the control plane machine set custom resource.

In RHOSP, flavors define the compute, memory, and storage capacity of computing instances. By increasing or decreasing the flavor size, you can scale your control plane vertically.

#### Prerequisites

- Your RHOSP cluster uses a control plane machine set.

#### Procedure

1. Edit the following line under the **providerSpec** field:

```

providerSpec:
  value:
# ...
  flavor: m1.xlarge ①

```

- 1 Specify a RHOSP flavor type that has the same base as the existing selection. For example, you can change **m6i.xlarge** to **m6i.2xlarge** or **m6i.4xlarge**. You can choose larger or smaller flavors depending on your vertical scaling needs.

2. Save your changes.

After you save your changes, machines are replaced with ones that use the flavor you chose.

### 12.5.6. Control plane configuration options for VMware vSphere

You can change the configuration of your VMware vSphere control plane machines by updating values in the control plane machine set. When you save an update to the control plane machine set, the Control Plane Machine Set Operator updates the control plane machines according to your configured [update strategy](#).

#### 12.5.6.1. Sample YAML for configuring VMware vSphere clusters

The following example YAML snippets show provider specification and failure domain configurations for a vSphere cluster.

#### 12.5.6.1.1. Sample VMware vSphere provider specification

When you create a control plane machine set for an existing cluster, the provider specification must match the **providerSpec** configuration in the control plane machine custom resource (CR) that is created by the installation program.

#### Sample vSphere providerSpec values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
  spec:
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1beta1
        credentialsSecret:
          name: vsphere-cloud-credentials 1
        diskGiB: 120 2
        kind: VSphereMachineProviderSpec 3
        memoryMiB: 16384 4
        metadata:
          creationTimestamp: null
        network: 5
        devices:
          - networkName: <vm_network_name>
        numCPUs: 4 6
        numCoresPerSocket: 4 7
        snapshot: ""
        template: <vm_template_name> 8
        userDataSecret:
          name: master-user-data 9
        workspace: 10
          datacenter: <vcenter_data_center_name> 11
          datastore: <vcenter_datastore_name> 12
          folder: <path_to_vcenter_vm_folder> 13
          resourcePool: <vsphere_resource_pool> 14
          server: <vcenter_server_ip> 15
```

- 1** Specifies the secret name for the cluster. Do not change this value.
- 2** Specifies the VM disk size for the control plane machines.
- 3** Specifies the cloud provider platform type. Do not change this value.

- 4 Specifies the memory allocated for the control plane machines.
- 5 Specifies the network on which the control plane is deployed.

**NOTE**

If the cluster is configured to use a failure domain, this parameter is configured in the failure domain. If you specify this value in the provider specification when using failure domains, the Control Plane Machine Set Operator ignores it.

- 6 Specifies the number of CPUs allocated for the control plane machines.
- 7 Specifies the number of cores for each control plane CPU.
- 8 Specifies the vSphere VM template to use, such as **user-5ddjd-rhcos**.

**NOTE**

If the cluster is configured to use a failure domain, this parameter is configured in the failure domain. If you specify this value in the provider specification when using failure domains, the Control Plane Machine Set Operator ignores it.

- 9 Specifies the control plane user data secret. Do not change this value.
- 10 Specifies the workspace details for the control plane.

**NOTE**

If the cluster is configured to use a failure domain, these parameters are configured in the failure domain. If you specify these values in the provider specification when using failure domains, the Control Plane Machine Set Operator ignores them.

- 11 Specifies the vCenter data center for the control plane.
- 12 Specifies the vCenter datastore for the control plane.
- 13 Specifies the path to the vSphere VM folder in vCenter, such as **/dc1/vm/user-inst-5ddjd**.
- 14 Specifies the vSphere resource pool for your VMs.
- 15 Specifies the vCenter server IP or fully qualified domain name.

#### 12.5.6.1.2. Sample VMware vSphere failure domain configuration

On VMware vSphere infrastructure, the cluster-wide infrastructure Custom Resource Definition (CRD), **infrastructures.config.openshift.io**, defines failure domains for your cluster. The **providerSpec** in the **ControlPlaneMachineSet** custom resource (CR) specifies names for failure domains that the control plane machine set uses to ensure control plane nodes are deployed to the appropriate failure domain. A failure domain is an infrastructure resource made up of a control plane machine set, a vCenter data center, vCenter datastore, and a network.

By using a failure domain resource, you can use a control plane machine set to deploy control plane machines on separate clusters or data centers. A control plane machine set also balances control plane machines across defined failure domains to provide fault tolerance capabilities to your infrastructure.



### NOTE

If you modify the **ProviderSpec** configuration in the **ControlPlaneMachineSet** CR, the control plane machine set updates all control plane machines deployed on the primary infrastructure and each failure domain infrastructure.

### Sample VMware vSphere failure domain values

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains: ①
        platform: VSphere
        vsphere: ②
          - name: <failure_domain_name1>
          - name: <failure_domain_name2>
# ...
```

- ① Specifies the vCenter location for OpenShift Container Platform cluster nodes.
- ② Specifies failure domains by name for the control plane machine set.



### IMPORTANT

Each **name** field value in this section must match the corresponding value in the **failureDomains.name** field of the cluster-wide infrastructure CRD. You can find the value of the **failureDomains.name** field by running the following command:

```
$ oc get infrastructure cluster -o=jsonpath=
{.spec.platformSpec.vsphere.failureDomains[0].name}
```

The **name** field is the only supported failure domain field that you can specify in the **ControlPlaneMachineSet** CR.

For an example of a cluster-wide infrastructure CRD that defines resources for each failure domain, see "Specifying multiple regions and zones for your cluster on vSphere."

### Additional resources

- [Specifying multiple regions and zones for your cluster on vSphere](#)

## 12.5.6.2. Enabling VMware vSphere features for control plane machines

You can enable features by updating values in the control plane machine set.

### 12.5.6.2.1. Adding tags to machines by using machine sets

OpenShift Container Platform adds a cluster-specific tag to each virtual machine (VM) that it creates. The installation program uses these tags to select the VMs to delete when uninstalling a cluster.

In addition to the cluster-specific tags assigned to VMs, you can configure a machine set to add up to 10 additional vSphere tags to the VMs it provisions.

#### Prerequisites

- You have access to an OpenShift Container Platform cluster installed on vSphere using an account with **cluster-admin** permissions.
- You have access to the VMware vCenter console associated with your cluster.
- You have created a tag in the vCenter console.
- You have installed the OpenShift CLI (**oc**).

#### Procedure

1. Use the vCenter console to find the tag ID for any tag that you want to add to your machines:
  - a. Log in to the vCenter console.
  - b. From the **Home** menu, click **Tags & Custom Attributes**
  - c. Select a tag that you want to add to your machines.
  - d. Use the browser URL for the tag that you select to identify the tag ID.

#### Example tag URL

```
https://vcenter.example.com/ui/app/tags/tag/urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL/permissions
```

#### Example tag ID

```
urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL
```

2. In a text editor, open the YAML file for an existing machine set or create a new one.
3. Edit the following lines under the **providerSpec** field:

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
```

```

    value:
    tagIDs: ①
    - <tag_id_value> ②
# ...

```

- ① Specify a list of up to 10 tags to add to the machines that this machine set provisions.
- ② Specify the value of the tag that you want to add to your machines. For example, **urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL**.

## 12.6. CONTROL PLANE RESILIENCY AND RECOVERY

You can use the control plane machine set to improve the resiliency of the control plane for your OpenShift Container Platform cluster.

### 12.6.1. High availability and fault tolerance with failure domains

When possible, the control plane machine set spreads the control plane machines across multiple failure domains. This configuration provides high availability and fault tolerance within the control plane. This strategy can help protect the control plane when issues arise within the infrastructure provider.

#### 12.6.1.1. Failure domain platform support and configuration

The control plane machine set concept of a failure domain is analogous to existing concepts on cloud providers. Not all platforms support the use of failure domains.

**Table 12.3. Failure domain support matrix**

Cloud provider	Support for failure domains	Provider nomenclature
Amazon Web Services (AWS)	X	<a href="#">Availability Zone (AZ)</a>
Google Cloud Platform (GCP)	X	<a href="#">zone</a>
Microsoft Azure	X	<a href="#">Azure availability zone</a>
Nutanix	X	<a href="#">failure domain</a>
Red Hat OpenStack Platform (RHOSP)	X	<a href="#">OpenStack Nova availability zones</a> and <a href="#">OpenStack Cinder availability zones</a>
VMware vSphere	X	failure domain mapped to a <a href="#">vSphere Zone</a> [1]

1. For more information, see "Regions and zones for a VMware vCenter".

The failure domain configuration in the control plane machine set custom resource (CR) is platform-specific. For more information about failure domain parameters in the CR, see the sample failure domain configuration for your provider.

## Additional resources

- [Sample Amazon Web Services failure domain configuration](#)
- [Sample Google Cloud Platform failure domain configuration](#)
- [Sample Microsoft Azure failure domain configuration](#)
- [Adding failure domains to an existing Nutanix cluster](#)
- [Sample Red Hat OpenStack Platform \(RHOSP\) failure domain configuration](#)
- [Sample VMware vSphere failure domain configuration](#)
- [Regions and zones for a VMware vCenter](#)

### 12.6.1.2. Balancing control plane machines

The control plane machine set balances control plane machines across the failure domains that are specified in the custom resource (CR).

When possible, the control plane machine set uses each failure domain equally to ensure appropriate fault tolerance. If there are fewer failure domains than control plane machines, failure domains are selected for reuse alphabetically by name. For clusters with no failure domains specified, all control plane machines are placed within a single failure domain.

Some changes to the failure domain configuration cause the control plane machine set to rebalance the control plane machines. For example, if you add failure domains to a cluster with fewer failure domains than control plane machines, the control plane machine set rebalances the machines across all available failure domains.

### 12.6.2. Recovery of failed control plane machines

The Control Plane Machine Set Operator automates the recovery of control plane machines. When a control plane machine is deleted, the Operator creates a replacement with the configuration that is specified in the **ControlPlaneMachineSet** custom resource (CR).

For clusters that use control plane machine sets, you can configure a machine health check. The machine health check deletes unhealthy control plane machines so that they are replaced.



#### IMPORTANT

If you configure a **MachineHealthCheck** resource for the control plane, set the value of **maxUnhealthy** to 1.

This configuration ensures that the machine health check takes no action when multiple control plane machines appear to be unhealthy. Multiple unhealthy control plane machines can indicate that the etcd cluster is degraded or that a scaling operation to replace a failed machine is in progress.

If the etcd cluster is degraded, manual intervention might be required. If a scaling operation is in progress, the machine health check should allow it to finish.

## Additional resources

- [Deploying machine health checks](#)

### 12.6.3. Quorum protection with machine lifecycle hooks

For OpenShift Container Platform clusters that use the Machine API Operator, the etcd Operator uses lifecycle hooks for the machine deletion phase to implement a quorum protection mechanism.

By using a **preDrain** lifecycle hook, the etcd Operator can control when the pods on a control plane machine are drained and removed. To protect etcd quorum, the etcd Operator prevents the removal of an etcd member until it migrates that member onto a new node within the cluster.

This mechanism allows the etcd Operator precise control over the members of the etcd quorum and allows the Machine API Operator to safely create and remove control plane machines without specific operational knowledge of the etcd cluster.

#### 12.6.3.1. Control plane deletion with quorum protection processing order

When a control plane machine is replaced on a cluster that uses a control plane machine set, the cluster temporarily has four control plane machines. When the fourth control plane node joins the cluster, the etcd Operator starts a new etcd member on the replacement node. When the etcd Operator observes that the old control plane machine is marked for deletion, it stops the etcd member on the old node and promotes the replacement etcd member to join the quorum of the cluster.

The control plane machine **Deleting** phase proceeds in the following order:

1. A control plane machine is slated for deletion.
2. The control plane machine enters the **Deleting** phase.
3. To satisfy the **preDrain** lifecycle hook, the etcd Operator takes the following actions:
  - a. The etcd Operator waits until a fourth control plane machine is added to the cluster as an etcd member. This new etcd member has a state of **Running** but not **ready** until it receives the full database update from the etcd leader.
  - b. When the new etcd member receives the full database update, the etcd Operator promotes the new etcd member to a voting member and removes the old etcd member from the cluster.

After this transition is complete, it is safe for the old etcd pod and its data to be removed, so the **preDrain** lifecycle hook is removed.

4. The control plane machine status condition **Drainable** is set to **True**.
5. The machine controller attempts to drain the node that is backed by the control plane machine.
  - If draining fails, **Drained** is set to **False** and the machine controller attempts to drain the node again.
  - If draining succeeds, **Drained** is set to **True**.
6. The control plane machine status condition **Drained** is set to **True**.
7. If no other Operators have added a **preTerminate** lifecycle hook, the control plane machine status condition **Terminable** is set to **True**.
8. The machine controller removes the instance from the infrastructure provider.
9. The machine controller deletes the **Node** object.

## YAML snippet demonstrating the etcd quorum protection preDrain lifecycle hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
...
spec:
lifecycleHooks:
preDrain:
- name: EtcdQuorumOperator 1
  owner: clusteroperator/etcd 2
...

```

- 1** The name of the **preDrain** lifecycle hook.
- 2** The hook-implementing controller that manages the **preDrain** lifecycle hook.

### Additional resources

- [Lifecycle hooks for the machine deletion phase](#)

## 12.7. TROUBLESHOOTING THE CONTROL PLANE MACHINE SET

Use the information in this section to understand and recover from issues you might encounter.

### 12.7.1. Checking the control plane machine set custom resource state

You can verify the existence and state of the **ControlPlaneMachineSet** custom resource (CR).

#### Procedure

- Determine the state of the CR by running the following command:

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
--namespace openshift-machine-api
```

- A result of **Active** indicates that the **ControlPlaneMachineSet** CR exists and is activated. No administrator action is required.
- A result of **Inactive** indicates that a **ControlPlaneMachineSet** CR exists but is not activated.
- A result of **NotFound** indicates that there is no existing **ControlPlaneMachineSet** CR.

#### Next steps

To use the control plane machine set, you must ensure that a **ControlPlaneMachineSet** CR with the correct settings for your cluster exists.

- If your cluster has an existing CR, you must verify that the configuration in the CR is correct for your cluster.

- If your cluster does not have an existing CR, you must create one with the correct configuration for your cluster.

## Additional resources

- [Activating the control plane machine set custom resource](#)
- [Creating a control plane machine set custom resource](#)

### 12.7.2. Adding a missing Azure internal load balancer

The **internalLoadBalancer** parameter is required in both the **ControlPlaneMachineSet** and control plane **Machine** custom resources (CRs) for Azure. If this parameter is not preconfigured on your cluster, you must add it to both CRs.

For more information about where this parameter is located in the Azure provider specification, see the sample Azure provider specification. The placement in the control plane **Machine** CR is similar.

## Procedure

1. List the control plane machines in your cluster by running the following command:

```
$ oc get machines \
-l machine.openshift.io/cluster-api-machine-role==master \
-n openshift-machine-api
```

2. For each control plane machine, edit the CR by running the following command:

```
$ oc edit machine <control_plane_machine_name>
```

3. Add the **internalLoadBalancer** parameter with the correct details for your cluster and save your changes.

4. Edit your control plane machine set CR by running the following command:

```
$ oc edit controlplanemachineset.machine.openshift.io cluster \
-n openshift-machine-api
```

5. Add the **internalLoadBalancer** parameter with the correct details for your cluster and save your changes.

## Next steps

- For clusters that use the default **RollingUpdate** update strategy, the Operator automatically propagates the changes to your control plane configuration.
- For clusters that are configured to use the **OnDelete** update strategy, you must replace your control plane machines manually.

## Additional resources

- [Sample Microsoft Azure provider specification](#)

### 12.7.3. Recovering a degraded etcd Operator

Certain situations can cause the etcd Operator to become degraded.

For example, while performing remediation, the machine health check might delete a control plane machine that is hosting etcd. If the etcd member is not reachable at that time, the etcd Operator becomes degraded.

When the etcd Operator is degraded, manual intervention is required to force the Operator to remove the failed member and restore the cluster state.

#### Procedure

1. List the control plane machines in your cluster by running the following command:

```
$ oc get machines \
-l machine.openshift.io/cluster-api-machine-role==master \
-n openshift-machine-api \
-o wide
```

Any of the following conditions might indicate a failed control plane machine:

- The **STATE** value is **stopped**.
- The **PHASE** value is **Failed**.
- The **PHASE** value is **Deleting** for more than ten minutes.



#### IMPORTANT

Before continuing, ensure that your cluster has two healthy control plane machines. Performing the actions in this procedure on more than one control plane machine risks losing etcd quorum and can cause data loss.

If you have lost the majority of your control plane hosts, leading to etcd quorum loss, then you must follow the disaster recovery procedure "Restoring to a previous cluster state" instead of this procedure.

2. Edit the machine CR for the failed control plane machine by running the following command:

```
$ oc edit machine <control_plane_machine_name>
```

3. Remove the contents of the **lifecycleHooks** parameter from the failed control plane machine and save your changes.

The etcd Operator removes the failed machine from the cluster and can then safely add new etcd members.

#### Additional resources

- [Restoring to a previous cluster state](#)

### 12.7.4. Upgrading clusters that run on RHOSP

For clusters that run on Red Hat OpenStack Platform (RHOSP) that were created with OpenShift Container Platform 4.13 or earlier, you might have to perform post-upgrade tasks before you can use control plane machine sets.

#### 12.7.4.1. Configuring RHOSP clusters that have machines with root volume availability zones after an upgrade

For some clusters that run on Red Hat OpenStack Platform (RHOSP) that you upgrade, you must manually update machine resources before you can use control plane machine sets if the following configurations are true:

- The upgraded cluster was created with OpenShift Container Platform 4.13 or earlier.
- The cluster infrastructure is installer-provisioned.
- Machines were distributed across multiple availability zones.
- Machines were configured to use root volumes for which block storage availability zones were not defined.

To understand why this procedure is necessary, see [Solution #7024383](#).

#### Procedure

1. For all control plane machines, edit the provider spec for all control plane machines that match the environment. For example, to edit the machine **master-0**, enter the following command:

```
$ oc edit machine/<cluster_id>-master-0 -n openshift-machine-api
```

where:

**<cluster\_id>**

Specifies the ID of the upgraded cluster.

2. In the provider spec, set the value of the property **rootVolume.availabilityZone** to the volume of the availability zone you want to use.

#### An example RHOSP provider spec

```
providerSpec:
  value:
    apiVersion: machine.openshift.io/v1alpha1
    availabilityZone: az0
    cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: m1.xlarge
    image: rhcos-4.14
    kind: OpenstackProviderSpec
    metadata:
      creationTimestamp: null
    networks:
      - filter: {}
        subnets:
```

```

- filter:
  name: refarch-lv7q9-nodes
  tags: openshiftClusterID=refarch-lv7q9
rootVolume:
  availabilityZone: nova ①
  diskSize: 30
  sourceUUID: rhcos-4.12
  volumeType: fast-0
securityGroups:
- filter: {}
  name: refarch-lv7q9-master
serverGroupName: refarch-lv7q9-master
serverMetadata:
  Name: refarch-lv7q9-master
  openshiftClusterID: refarch-lv7q9
tags:
- openshiftClusterID=refarch-lv7q9
trunk: true
userDataSecret:
  name: master-user-data

```

- ① Set the zone name as this value.



#### NOTE

If you edited or recreated machine resources after your initial cluster deployment, you might have to adapt these steps for your configuration.

In your RHOSP cluster, find the availability zone of the root volumes for your machines and use that as the value.

- Run the following command to retrieve information about the control plane machine set resource:

```
$ oc describe controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

- Run the following command to edit the resource:

```
$ oc edit controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

- For that resource, set the value of the **spec.state** property to **Active** to activate control plane machine sets for your cluster.

Your control plane is ready to be managed by the Cluster Control Plane Machine Set Operator.

#### 12.7.4.2. Configuring RHOSP clusters that have control plane machines with availability zones after an upgrade

For some clusters that run on Red Hat OpenStack Platform (RHOSP) that you upgrade, you must manually update machine resources before you can use control plane machine sets if the following configurations are true:

- The upgraded cluster was created with OpenShift Container Platform 4.13 or earlier.
- The cluster infrastructure is installer-provisioned.
- Control plane machines were distributed across multiple compute availability zones.

To understand why this procedure is necessary, see [Solution #7013893](#).

## Procedure

1. For the **master-1** and **master-2** control plane machines, open the provider specs for editing. For example, to edit the first machine, enter the following command:

```
$ oc edit machine/<cluster_id>-master-1 -n openshift-machine-api
```

where:

**<cluster\_id>**

Specifies the ID of the upgraded cluster.

2. For the **master-1** and **master-2** control plane machines, edit the value of the **serverGroupName** property in their provider specs to match that of the machine **master-0**.

### An example RHOSP provider spec

```
providerSpec:
  value:
    apiVersion: machine.openshift.io/v1alpha1
    availabilityZone: az0
    cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: m1.xlarge
    image: rhcos-4.17
    kind: OpenstackProviderSpec
    metadata:
      creationTimestamp: null
    networks:
      - filter: {}
        subnets:
          - filter:
              name: refarch-lv7q9-nodes
              tags: openshiftClusterID=refarch-lv7q9
    securityGroups:
      - filter: {}
        name: refarch-lv7q9-master
    serverGroupName: refarch-lv7q9-master-az0 ①
    serverMetadata:
      Name: refarch-lv7q9-master
      openshiftClusterID: refarch-lv7q9
    tags:
      - openshiftClusterID=refarch-lv7q9
```

```

trunk: true
userDataSecret:
  name: master-user-data

```

- 1** This value must match for machines **master-0**, **master-1**, and **master-3**.

**NOTE**

If you edited or recreated machine resources after your initial cluster deployment, you might have to adapt these steps for your configuration.

In your RHOSP cluster, find the server group that your control plane instances are in and use that as the value.

- Run the following command to retrieve information about the control plane machine set resource:

```
$ oc describe controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

- Run the following command to edit the resource:

```
$ oc edit controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

- For that resource, set the value of the **spec.state** property to **Active** to activate control plane machine sets for your cluster.

Your control plane is ready to be managed by the Cluster Control Plane Machine Set Operator.

## 12.8. DISABLING THE CONTROL PLANE MACHINE SET

The **.spec.state** field in an activated **ControlPlaneMachineSet** custom resource (CR) cannot be changed from **Active** to **Inactive**. To disable the control plane machine set, you must delete the CR so that it is removed from the cluster.

When you delete the CR, the Control Plane Machine Set Operator performs cleanup operations and disables the control plane machine set. The Operator then removes the CR from the cluster and creates an inactive control plane machine set with default settings.

### 12.8.1. Deleting the control plane machine set

To stop managing control plane machines with the control plane machine set on your cluster, you must delete the **ControlPlaneMachineSet** custom resource (CR).

#### Procedure

- Delete the control plane machine set CR by running the following command:

```
$ oc delete controlplanemachineset.machine.openshift.io cluster \
-n openshift-machine-api
```

## Verification

- Check the control plane machine set custom resource state. A result of **Inactive** indicates that the removal and replacement process is successful. A **ControlPlaneMachineSet** CR exists but is not activated.

### 12.8.2. Checking the control plane machine set custom resource state

You can verify the existence and state of the **ControlPlaneMachineSet** custom resource (CR).

#### Procedure

- Determine the state of the CR by running the following command:

```
$ oc get controlplanemachineset.machine.openshift.io cluster \  
--namespace openshift-machine-api
```

- A result of **Active** indicates that the **ControlPlaneMachineSet** CR exists and is activated. No administrator action is required.
- A result of **Inactive** indicates that a **ControlPlaneMachineSet** CR exists but is not activated.
- A result of **NotFound** indicates that there is no existing **ControlPlaneMachineSet** CR.

### 12.8.3. Re-enabling the control plane machine set

To re-enable the control plane machine set, you must ensure that the configuration in the CR is correct for your cluster and activate it.

#### Additional resources

- [Activating the control plane machine set custom resource](#)

# CHAPTER 13. MANAGING MACHINES WITH THE CLUSTER API

## 13.1. ABOUT THE CLUSTER API



### IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

The [Cluster API](#) is an upstream project that is integrated into OpenShift Container Platform as a Technology Preview for Amazon Web Services (AWS), Google Cloud Platform (GCP), and VMware vSphere.

### 13.1.1. Cluster API overview

You can use the Cluster API to create and manage compute machine sets and compute machines in your OpenShift Container Platform cluster. This capability is in addition or an alternative to managing machines with the Machine API.

For OpenShift Container Platform 4.17 clusters, you can use the Cluster API to perform node host provisioning management actions after the cluster installation finishes. This system enables an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

With the Cluster API Technology Preview, you can create compute machines and compute machine sets on OpenShift Container Platform clusters for supported providers. You can also explore the features that are enabled by this implementation that might not be available with the Machine API.

#### 13.1.1.1. Cluster API benefits

By using the Cluster API, OpenShift Container Platform users and developers gain the following advantages:

- The option to use upstream community Cluster API infrastructure providers that might not be supported by the Machine API.
- The opportunity to collaborate with third parties who maintain machine controllers for infrastructure providers.
- The ability to use the same set of Kubernetes tools for infrastructure management in OpenShift Container Platform.
- The ability to create compute machine sets by using the Cluster API that support features that are not available with the Machine API.

#### 13.1.1.2. Cluster API limitations

Using the Cluster API to manage machines is a Technology Preview feature and has the following limitations:

- To use this feature, you must enable the **TechPreviewNoUpgrade** feature set.



## IMPORTANT

Enabling this feature set cannot be undone and prevents minor version updates.

- Only Amazon Web Services (AWS), Google Cloud Platform (GCP), and VMware vSphere clusters can use the Cluster API.
- You must manually create the primary resources that the Cluster API requires. For more information, see "Getting started with the Cluster API".
- You cannot use the Cluster API to manage control plane machines.
- Migration of existing compute machine sets created by the Machine API to Cluster API compute machine sets is not supported.
- Full feature parity with the Machine API is not available.
- For clusters that use the Cluster API, OpenShift CLI (**oc**) commands prioritize Cluster API objects over Machine API objects. This behavior impacts any **oc** command that acts upon any object that is represented in both the Cluster API and the Machine API.  
For more information and a workaround for this issue, see "Referencing the intended objects when using the CLI" in the troubleshooting content.

## Additional resources

- [Enabling features using feature gates](#)
- [Getting started with the Cluster API](#)
- [Referencing the intended objects when using the CLI](#)

### 13.1.2. Cluster API architecture

The OpenShift Container Platform integration of the upstream Cluster API is implemented and managed by the Cluster CAPI Operator. The Cluster CAPI Operator and its operands are provisioned in the **openshift-cluster-api** namespace, in contrast to the Machine API, which uses the **openshift-machine-api** namespace.

#### 13.1.2.1. The Cluster CAPI Operator

The Cluster CAPI Operator is an OpenShift Container Platform Operator that maintains the lifecycle of Cluster API resources. This Operator is responsible for all administrative tasks related to deploying the Cluster API project within an OpenShift Container Platform cluster.

If a cluster is configured correctly to allow the use of the Cluster API, the Cluster CAPI Operator installs the Cluster API components on the cluster.

For more information, see the "Cluster CAPI Operator" entry in the *Cluster Operators* reference content.

## Additional resources

- [Cluster CAPI Operator](#)

### 13.1.2.2. Cluster API primary resources

The Cluster API consists of the following primary resources. For the Technology Preview of this feature, you must create these resources manually in the **openshift-cluster-api** namespace.

#### Cluster

A fundamental unit that represents a cluster that is managed by the Cluster API.

#### Infrastructure

A provider-specific resource that defines properties that are shared by all the compute machine sets in the cluster, such as the region and subnets.

#### Machine template

A provider-specific template that defines the properties of the machines that a compute machine set creates.

#### Machine set

A group of machines.

Compute machine sets are to machines as replica sets are to pods. To add machines or scale them down, change the **replicas** field on the compute machine set custom resource to meet your compute needs.

With the Cluster API, a compute machine set references a **Cluster** object and a provider-specific machine template.

#### Machine

A fundamental unit that describes the host for a node.

The Cluster API creates machines based on the configuration in the machine template.

## 13.2. GETTING STARTED WITH THE CLUSTER API



### IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

For the Cluster API Technology Preview, you must create the primary resources that the Cluster API requires manually.

### 13.2.1. Creating the Cluster API primary resources

To create the Cluster API primary resources, you must obtain the cluster ID value, which you use for the **<cluster\_name>** parameter in the cluster resource manifest.

### 13.2.1.1. Obtaining the cluster ID value

You can find the cluster ID value by using the OpenShift CLI (**oc**).

#### Prerequisites

- You have deployed an OpenShift Container Platform cluster.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

#### Procedure

- Obtain the value of the cluster ID by running the following command:

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

You can create the Cluster API primary resources manually by creating YAML manifest files and applying them with the OpenShift CLI (**oc**).

### 13.2.1.2. Creating the Cluster API cluster resource

You can create the cluster resource by creating a YAML manifest file and applying it with the OpenShift CLI (**oc**).

#### Prerequisites

- You have deployed an OpenShift Container Platform cluster.
- You have enabled the use of the Cluster API.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have the cluster ID value.

#### Procedure

- 1 Create a YAML file similar to the following. This procedure uses **<cluster\_resource\_file>.yaml** as an example file name.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: <cluster_name> ①
  namespace: openshift-cluster-api
spec:
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: <infrastructure_kind> ②
    name: <cluster_name>
    namespace: openshift-cluster-api
```

- 1 Specify the cluster ID as the name of the cluster.
- 2 Specify the infrastructure kind for the cluster. The following values are valid:
  - **AWSCluster**: The cluster is running on Amazon Web Services (AWS).
  - **GCPCluster**: The cluster is running on Google Cloud Platform (GCP).
  - **VSphereCluster**: The cluster is running on VMware vSphere.

2. Create the cluster CR by running the following command:

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

## Verification

- Confirm that the cluster CR exists by running the following command:

```
$ oc get cluster
```

### Example output

NAME	PHASE	AGE	VERSION
<cluster_name>	Provisioning	4h6m	

The cluster resource is ready when the value of **PHASE** is **Provisioned**.

## Additional resources

- [Cluster API configuration](#)

### 13.2.1.3. Creating a Cluster API infrastructure resource

You can create a provider-specific infrastructure resource by creating a YAML manifest file and applying it with the OpenShift CLI (**oc**).

#### Prerequisites

- You have deployed an OpenShift Container Platform cluster.
- You have enabled the use of the Cluster API.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have the cluster ID value.
- You have created and applied the cluster resource.

#### Procedure

- Create a YAML file similar to the following. This procedure uses **<infrastructure\_resource\_file>.yaml** as an example file name.

```
apiVersion: infrastructure.cluster.x-k8s.io/<version> 1
kind: <infrastructure_kind> 2
metadata:
  name: <cluster_name> 3
  namespace: openshift-cluster-api
spec: 4
```

- The **apiVersion** varies by platform. For more information, see the sample Cluster API infrastructure resource YAML for your provider. The following values are valid:
  - infrastructure.cluster.x-k8s.io/v1beta2**: The version that Amazon Web Services (AWS) clusters use.
  - infrastructure.cluster.x-k8s.io/v1beta1**: The version that Google Cloud Platform (GCP) and VMware vSphere clusters use.
- Specify the infrastructure kind for the cluster. This value must match the value for your platform. The following values are valid:
  - AWSCluster**: The cluster is running on AWS.
  - GCPCluster**: The cluster is running on GCP.
  - VSphereCluster**: The cluster is running on vSphere.
- Specify the name of the cluster.
- Specify the details for your environment. These parameters are provider specific. For more information, see the sample Cluster API infrastructure resource YAML for your provider.

- Create the infrastructure CR by running the following command:

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

## Verification

- Confirm that the infrastructure CR is created by running the following command:

```
$ oc get <infrastructure_kind>
```

where **<infrastructure\_kind>** is the value that corresponds to your platform.

## Example output

```
NAME      CLUSTER      READY
<cluster_name>  <cluster_name>  true
```

**NOTE**

This output might contain additional columns that are specific to your cloud provider.

**Additional resources**

- [Sample YAML for a Cluster API infrastructure resource on Amazon Web Services](#)
- [Sample YAML for a Cluster API infrastructure resource on Google Cloud Platform](#)
- [Sample YAML for a Cluster API infrastructure resource on VMware vSphere](#)

**13.2.1.4. Creating a Cluster API machine template**

You can create a provider-specific machine template resource by creating a YAML manifest file and applying it with the OpenShift CLI (**oc**).

**Prerequisites**

- You have deployed an OpenShift Container Platform cluster.
- You have enabled the use of the Cluster API.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have created and applied the cluster and infrastructure resources.

**Procedure**

- 1 Create a YAML file similar to the following. This procedure uses **<machine\_template\_resource\_file>.yaml** as an example file name.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: <machine_template_kind> ①
metadata:
  name: <template_name> ②
  namespace: openshift-cluster-api
spec:
  template:
    spec: ③
```

- 1 Specify the machine template kind. This value must match the value for your platform. The following values are valid:
  - **AWSMachineTemplate**: The cluster is running on Amazon Web Services (AWS).
  - **GCPMachineTemplate**: The cluster is running on Google Cloud Platform (GCP).
  - **VSphereMachineTemplate**: The cluster is running on VMware vSphere.
- 2 Specify a name for the machine template.

- 3** Specify the details for your environment. These parameters are provider specific. For more information, see the sample Cluster API machine template YAML for your provider.

2. Create the machine template CR by running the following command:

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

## Verification

- Confirm that the machine template CR is created by running the following command:

```
$ oc get <machine_template_kind>
```

where **<machine\_template\_kind>** is the value that corresponds to your platform.

## Example output

```
NAME      AGE
<template_name>  77m
```

## Additional resources

- [Sample YAML for a Cluster API machine template resource on Amazon Web Services](#)
- [Sample YAML for a Cluster API machine template resource on Google Cloud Platform](#)
- [Sample YAML for a Cluster API machine template resource on VMware vSphere](#)

### 13.2.1.5. Creating a Cluster API compute machine set

You can create compute machine sets that use the Cluster API to dynamically manage the machine compute resources for specific workloads of your choice.

## Prerequisites

- You have deployed an OpenShift Container Platform cluster.
- You have enabled the use of the Cluster API.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have created the cluster, infrastructure, and machine template resources.

## Procedure

1. Create a YAML file similar to the following. This procedure uses **<machine\_set\_resource\_file>.yaml** as an example file name.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
```

```

name: <machine_set_name> ①
namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> ②
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec: ③
# ...

```

- ① Specify a name for the compute machine set.
- ② Specify the name of the cluster.
- ③ Specify the details for your environment. These parameters are provider specific. For more information, see the sample Cluster API compute machine set YAML for your provider.

## 2. Create the compute machine set CR by running the following command:

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

## 3. Confirm that the compute machine set CR is created by running the following command:

```
$ oc get machineset -n openshift-cluster-api ①
```

- ① Specify the **openshift-cluster-api** namespace.

### Example output

NAME	CLUSTER	REPLICAS	READY	AVAILABLE	AGE	VERSION
<machine_set_name>	<cluster_name>	1	1	1	17m	

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

## Verification

- To verify that the compute machine set is creating machines according to your required configuration, review the lists of machines and nodes in the cluster by running the following commands:
  - View the list of Cluster API machines:

```
$ oc get machine -n openshift-cluster-api ①
```

- ① Specify the **openshift-cluster-api** namespace.

### Example output

```

NAME          CLUSTER      NODENAME      PROVIDERID
PHASE AGE    VERSION
<machine_set_name>-<string_id> <cluster_name> <ip_address>.
<region>.compute.internal <provider_id> Running 8m23s

```

- View the list of nodes:

```
$ oc get node
```

### Example output

```

NAME          STATUS ROLES AGE VERSION
<ip_address_1>.<region>.compute.internal Ready worker 5h14m v1.28.5
<ip_address_2>.<region>.compute.internal Ready master 5h19m v1.28.5
<ip_address_3>.<region>.compute.internal Ready worker 7m v1.28.5

```

### Additional resources

- [Sample YAML for a Cluster API compute machine set resource on Amazon Web Services](#)
- [Sample YAML for a Cluster API compute machine set resource on Google Cloud Platform](#)
- [Sample YAML for a Cluster API compute machine set resource on VMware vSphere](#)

## 13.3. MANAGING MACHINES WITH THE CLUSTER API



### IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

### 13.3.1. Modifying a Cluster API machine template

You can update the machine template resource for your cluster by modifying the YAML manifest file and applying it with the OpenShift CLI (**oc**).

#### Prerequisites

- You have deployed an OpenShift Container Platform cluster that uses the Cluster API.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

## Procedure

1. List the machine template resource for your cluster by running the following command:

```
$ oc get <machine_template_kind> ①
```

- 1 Specify the value that corresponds to your platform. The following values are valid:

- **AWSMachineTemplate**: The cluster is running on Amazon Web Services (AWS).
- **GCPMachineTemplate**: The cluster is running on Google Cloud Platform (GCP).
- **VSphereMachineTemplate**: The cluster is running on VMware vSphere.

### Example output

```
NAME      AGE  
<template_name>  77m
```

2. Write the machine template resource for your cluster to a file that you can edit by running the following command:

```
$ oc get <template_name> -o yaml > <template_name>.yaml
```

where **<template\_name>** is the name of the machine template resource for your cluster.

3. Make a copy of the **<template\_name>.yaml** file with a different name. This procedure uses **<modified\_template\_name>.yaml** as an example file name.
4. Use a text editor to make changes to the **<modified\_template\_name>.yaml** file that defines the updated machine template resource for your cluster. When editing the machine template resource, observe the following:
  - The parameters in the **spec** stanza are provider specific. For more information, see the sample Cluster API machine template YAML for your provider.
  - You must use a value for the **metadata.name** parameter that differs from any existing values.



### IMPORTANT

For any Cluster API compute machine sets that reference this template, you must update the **spec.template.spec.infrastructureRef.name** parameter to match the **metadata.name** value in the new machine template resource.

5. Apply the machine template CR by running the following command:

```
$ oc apply -f <modified_template_name>.yaml ①
```

- 1 Use the edited YAML file with a new name.

## Next steps

- For any Cluster API compute machine sets that reference this template, update the **spec.template.spec.infrastructureRef.name** parameter to match the **metadata.name** value in the new machine template resource. For more information, see "Modifying a compute machine set by using the CLI."

## Additional resources

- [Sample YAML for a Cluster API machine template resource on Amazon Web Services](#)
- [Sample YAML for a Cluster API machine template resource on Google Cloud Platform](#)
- [Sample YAML for a Cluster API machine template resource on VMware vSphere](#)
- [Modifying a compute machine set by using the CLI](#)

### 13.3.2. Modifying a compute machine set by using the CLI

You can modify the configuration of a compute machine set, and then propagate the changes to the machines in your cluster by using the CLI.

By updating the compute machine set configuration, you can enable features or change the properties of the machines it creates. When you modify a compute machine set, your changes only apply to compute machines that are created after you save the updated **MachineSet** custom resource (CR). The changes do not affect existing machines.



#### NOTE

Changes made in the underlying cloud provider are not reflected in the **Machine** or **MachineSet** CRs. To adjust instance configuration in cluster-managed infrastructure, use the cluster-side resources.

You can replace the existing machines with new ones that reflect the updated configuration by scaling the compute machine set to create twice the number of replicas and then scaling it down to the original number of replicas.

If you need to scale a compute machine set without making other changes, you do not need to delete the machines.



#### NOTE

By default, the OpenShift Container Platform router pods are deployed on compute machines. Because the router is required to access some cluster resources, including the web console, do not scale the compute machine set to **0** unless you first relocate the router pods.

The output examples in this procedure use the values for an AWS cluster.

## Prerequisites

- Your OpenShift Container Platform cluster uses the Cluster API.
- You are logged in to the cluster as an administrator by using the OpenShift CLI (**oc**).

## Procedure

1. List the compute machine sets in your cluster by running the following command:

```
$ oc get machinesets.cluster.x-k8s.io -n openshift-cluster-api
```

#### Example output

NAME VERSION	CLUSTER	REPLICAS	READY	AVAILABLE	AGE
<compute_machine_set_name_1>	<cluster_name>	1	1	1	26m
<compute_machine_set_name_2>	<cluster_name>	1	1	1	26m

2. Edit a compute machine set by running the following command:

```
$ oc edit machinesets.cluster.x-k8s.io <machine_set_name> \  
-n openshift-cluster-api
```

3. Note the value of the **spec.replicas** field, because you need it when scaling the machine set to apply the changes.

```
apiVersion: cluster.x-k8s.io/v1beta1  
kind: MachineSet  
metadata:  
  name: <machine_set_name>  
  namespace: openshift-cluster-api  
spec:  
  replicas: 2 ①  
# ...
```

① The examples in this procedure show a compute machine set that has a **replicas** value of **2**.

4. Update the compute machine set CR with the configuration options that you want and save your changes.
5. List the machines that are managed by the updated compute machine set by running the following command:

```
$ oc get machines.cluster.x-k8s.io \  
-n openshift-cluster-api \  
-l cluster.x-k8s.io/set-name=<machine_set_name>
```

#### Example output for an AWS cluster

NAME PHASE	CLUSTER	NODENAME	PROVIDERID
AGE	VERSION		
<machine_name_original_1>	<cluster_name>	<original_1_ip>.<region>.compute.internal aws://us-east-2a/i-04e7b2cbd61fd2075	Running 4h
		<original_2_ip>.<region>.compute.internal aws://us-east-2a/i-04e7b2cbd61fd2075	Running 4h

6. For each machine that is managed by the updated compute machine set, set the **delete** annotation by running the following command:

```
$ oc annotate machines.cluster.x-k8s.io/<machine_name_original_1> \
-n openshift-cluster-api \
cluster.x-k8s.io/delete-machine="true"
```

7. To create replacement machines with the new configuration, scale the compute machine set to twice the number of replicas by running the following command:

```
$ oc scale --replicas=4 \① \
machinesets.cluster.x-k8s.io <machine_set_name> \
-n openshift-cluster-api
```

- ① The original example value of 2 is doubled to 4.

8. List the machines that are managed by the updated compute machine set by running the following command:

```
$ oc get machines.cluster.x-k8s.io \
-n openshift-cluster-api \
-l cluster.x-k8s.io/set-name=<machine_set_name>
```

### Example output for an AWS cluster

NAME	CLUSTER	NODENAME	PROVIDERID
PHASE	AGE	VERSION	
<machine_name_original_1>	<cluster_name>	<original_1_ip>.<region>.compute.internal	
aws://us-east-2a/i-04e7b2cbd61fd2075	Running	4h	
<machine_name_original_2>	<cluster_name>	<original_2_ip>.<region>.compute.internal	
aws://us-east-2a/i-04e7b2cbd61fd2075	Running	4h	
<machine_name_updated_1>	<cluster_name>	<updated_1_ip>.<region>.compute.internal	
aws://us-east-2a/i-04e7b2cbd61fd2075	Provisioned	55s	
<machine_name_updated_2>	<cluster_name>	<updated_2_ip>.<region>.compute.internal	
aws://us-east-2a/i-04e7b2cbd61fd2075	Provisioning	55s	

When the new machines are in the **Running** phase, you can scale the compute machine set to the original number of replicas.

9. To remove the machines that were created with the old configuration, scale the compute machine set to the original number of replicas by running the following command:

```
$ oc scale --replicas=2 \① \
machinesets.cluster.x-k8s.io <machine_set_name> \
-n openshift-cluster-api
```

- ① The original example value of 2.

### Verification

- To verify that a machine created by the updated machine set has the correct configuration, examine the relevant fields in the CR for one of the new machines by running the following command:

```
$ oc describe machines.cluster.x-k8s.io <machine_name_updated_1> \
-n openshift-cluster-api
```

- To verify that the compute machines without the updated configuration are deleted, list the machines that are managed by the updated compute machine set by running the following command:

```
$ oc get machines.cluster.x-k8s.io \
-n openshift-cluster-api \
cluster.x-k8s.io/set-name=<machine_set_name>
```

#### Example output while deletion is in progress for an AWS cluster

NAME PHASE	CLUSTER AGE	NODENAME VERSION	PROVIDERID
<machine_name_original_1>	<cluster_name>	<original_1_ip>.<region>.compute.internal	
aws://us-east-2a/i-04e7b2cbd61fd2075	Running	18m	
<machine_name_original_2>	<cluster_name>	<original_2_ip>.<region>.compute.internal	
aws://us-east-2a/i-04e7b2cbd61fd2075	Running	18m	
<machine_name_updated_1>	<cluster_name>	<updated_1_ip>.	
<region>.compute.internal	aws://us-east-2a/i-04e7b2cbd61fd2075	Running	18m
<machine_name_updated_2>	<cluster_name>	<updated_2_ip>.	
<region>.compute.internal	aws://us-east-2a/i-04e7b2cbd61fd2075	Running	18m

#### Example output when deletion is complete for an AWS cluster

NAME PHASE	CLUSTER AGE	NODENAME VERSION	PROVIDERID
<machine_name_updated_1>	<cluster_name>	<updated_1_ip>.	
<region>.compute.internal	aws://us-east-2a/i-04e7b2cbd61fd2075	Running	18m
<machine_name_updated_2>	<cluster_name>	<updated_2_ip>.	
<region>.compute.internal	aws://us-east-2a/i-04e7b2cbd61fd2075	Running	18m

#### Additional resources

- [Sample YAML for a Cluster API compute machine set resource on Amazon Web Services](#)
- [Sample YAML for a Cluster API compute machine set resource on Google Cloud Platform](#)
- [Sample YAML for a Cluster API compute machine set resource on VMware vSphere](#)

## 13.4. CLUSTER API CONFIGURATION



## IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

The following example YAML files show how to make the Cluster API primary resources work together and configure settings for the machines that they create that are appropriate for your environment.

### 13.4.1. Sample YAML for a Cluster API cluster resource

The cluster resource defines the name and infrastructure provider for the cluster and is managed by the Cluster API. This resource has the same structure for all providers.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: <cluster_name> ①
  namespace: openshift-cluster-api
spec:
  controlPlaneEndpoint: ②
    host: <control_plane_endpoint_address>
    port: 6443
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: <infrastructure_kind> ③
    name: <cluster_name>
    namespace: openshift-cluster-api
```

- ① Specify the name of the cluster.
- ② Specify the IP address of the control plane endpoint and the port used to access it.
- ③ Specify the infrastructure kind for the cluster. Valid values are:
  - **AWSCluster**: The cluster is running on Amazon Web Services.
  - **GCPCluster**: The cluster is running on Google Cloud Platform.
  - **VSphereCluster**: The cluster is running on VMware vSphere.

### 13.4.2. Provider-specific configuration options

The remaining Cluster API resources are provider-specific. For provider-specific configuration options for your cluster, see the following resources:

- [Cluster API configuration options for Amazon Web Services](#)

- Cluster API configuration options for Google Cloud Platform
- Cluster API configuration options for VMware vSphere

## 13.5. CONFIGURATION OPTIONS FOR CLUSTER API MACHINES

### 13.5.1. Cluster API configuration options for Amazon Web Services



#### IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

You can change the configuration of your Amazon Web Services (AWS) Cluster API machines by updating values in the Cluster API custom resource manifests.

#### 13.5.1.1. Sample YAML for configuring Amazon Web Services clusters

The following example YAML files show configurations for an Amazon Web Services cluster.

##### 13.5.1.1.1. Sample YAML for a Cluster API infrastructure resource on Amazon Web Services

The infrastructure resource is provider-specific and defines properties that are shared by all the compute machine sets in the cluster, such as the region and subnets. The compute machine set references this resource when creating machines.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: AWSCluster 1
metadata:
  name: <cluster_name> 2
  namespace: openshift-cluster-api
spec:
  controlPlaneEndpoint: 3
    host: <control_plane_endpoint_address>
    port: 6443
  region: <region> 4
```

- 1 Specify the infrastructure kind for the cluster. This value must match the value for your platform.
- 2 Specify the cluster ID as the name of the cluster.
- 3 Specify the address of the control plane endpoint and the port to use to access it.
- 4 Specify the AWS region.

### 13.5.1.2. Sample YAML for a Cluster API machine template resource on Amazon Web Services

The machine template resource is provider-specific and defines the basic properties of the machines that a compute machine set creates. The compute machine set references this template when creating machines.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: AWSMachineTemplate 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      uncompressedUserData: true
      iamInstanceProfile: # ...
      instanceType: m5.large
      ignition:
        storageType: UnencryptedUserData
        version: "3.2"
      ami:
        id: # ...
      subnet:
        filters:
          - name: tag:Name
            values:
              - # ...
      additionalSecurityGroups:
        - filters:
          - name: tag:Name
            values:
              - # ...
```

- 1** Specify the machine template kind. This value must match the value for your platform.
- 2** Specify a name for the machine template.
- 3** Specify the details for your environment. The values here are examples.

### 13.5.1.3. Sample YAML for a Cluster API compute machine set resource on Amazon Web Services

The compute machine set resource defines additional properties of the machines that it creates. The compute machine set also references the infrastructure resource and machine template when creating machines.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
```

```

matchLabels:
  test: example
template:
  metadata:
    labels:
      test: example
spec:
  bootstrap:
    dataSecretName: worker-user-data 3
  clusterName: <cluster_name>
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: AWSMachineTemplate 4
    name: <template_name> 5

```

- 1** Specify a name for the compute machine set.
- 2** Specify the cluster ID as the name of the cluster.
- 3** For the Cluster API Technology Preview, the Operator can use the worker user data secret from the **openshift-machine-api** namespace.
- 4** Specify the machine template kind. This value must match the value for your platform.
- 5** Specify the machine template name.

### 13.5.2. Cluster API configuration options for Google Cloud Platform



#### IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

You can change the configuration of your Google Cloud Platform (GCP) Cluster API machines by updating values in the Cluster API custom resource manifests.

#### 13.5.2.1. Sample YAML for configuring Google Cloud Platform clusters

The following example YAML files show configurations for a Google Cloud Platform cluster.

##### 13.5.2.1.1. Sample YAML for a Cluster API infrastructure resource on Google Cloud Platform

The infrastructure resource is provider-specific and defines properties that are shared by all the compute machine sets in the cluster, such as the region and subnets. The compute machine set references this resource when creating machines.

```

apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPCluster ①
metadata:
  name: <cluster_name> ②
spec:
  controlPlaneEndpoint: ③
    host: <control_plane_endpoint_address>
    port: 6443
  network:
    name: <cluster_name>-network
    project: <project> ④
    region: <region> ⑤

```

- ① Specify the infrastructure kind for the cluster. This value must match the value for your platform.
- ② Specify the cluster ID as the name of the cluster.
- ③ Specify the IP address of the control plane endpoint and the port used to access it.
- ④ Specify the GCP project name.
- ⑤ Specify the GCP region.

### 13.5.2.1.2. Sample YAML for a Cluster API machine template resource on Google Cloud Platform

The machine template resource is provider-specific and defines the basic properties of the machines that a compute machine set creates. The compute machine set references this template when creating machines.

```

apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPMachineTemplate ①
metadata:
  name: <template_name> ②
  namespace: openshift-cluster-api
spec:
  template:
    spec: ③
      rootDeviceType: pd-ssd
      rootDeviceSize: 128
      instanceType: n1-standard-4
      image: projects/rhcos-cloud/global/images/rhcos-411-85-202203181601-0-gcp-x86-64
      subnet: <cluster_name>-worker-subnet
      serviceAccounts:
        email: <service_account_email_address>
        scopes:
          - https://www.googleapis.com/auth/cloud-platform
      additionalLabels:
        kubernetes-io-cluster-<cluster_name>: owned
      additionalNetworkTags:
        - <cluster_name>-worker
      ipForwarding: Disabled

```

- ① Specify the machine template kind. This value must match the value for your platform.

- 2 Specify a name for the machine template.
- 3 Specify the details for your environment. The values here are examples.

### 13.5.2.1.3. Sample YAML for a Cluster API compute machine set resource on Google Cloud Platform

The compute machine set resource defines additional properties of the machines that it creates. The compute machine set also references the infrastructure resource and machine template when creating machines.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: GCPMachineTemplate 4
        name: <template_name> 5
      failureDomain: <failure_domain> 6
```

- 1 Specify a name for the compute machine set.
- 2 Specify the cluster ID as the name of the cluster.
- 3 For the Cluster API Technology Preview, the Operator can use the worker user data secret from the **openshift-machine-api** namespace.
- 4 Specify the machine template kind. This value must match the value for your platform.
- 5 Specify the machine template name.
- 6 Specify the failure domain within the GCP region.

### 13.5.3. Cluster API configuration options for VMware vSphere



## IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

You can change the configuration of your VMware vSphere Cluster API machines by updating values in the Cluster API custom resource manifests.

### 13.5.3.1. Sample YAML for configuring VMware vSphere clusters

The following example YAML files show configurations for a VMware vSphere cluster.

#### 13.5.3.1.1. Sample YAML for a Cluster API infrastructure resource on VMware vSphere

The infrastructure resource is provider-specific and defines properties that are shared by all the compute machine sets in the cluster, such as the region and subnets. The compute machine set references this resource when creating machines.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: VSphereCluster ①
metadata:
  name: <cluster_name> ②
spec:
  controlPlaneEndpoint: ③
    host: <control_plane_endpoint_address>
    port: 6443
  identityRef:
    kind: Secret
    name: <cluster_name>
  server: <vsphere_server> ④
```

- ① Specify the infrastructure kind for the cluster. This value must match the value for your platform.
- ② Specify the cluster ID as the name of the cluster.
- ③ Specify the IP address of the control plane endpoint and the port used to access it.
- ④ Specify the vSphere server for the cluster. You can find this value on an existing vSphere cluster by running the following command:

```
$ oc get infrastructure cluster \
-o jsonpath='{.spec.platformSpec.vsphere.vcenters[0].server}'
```

#### 13.5.3.1.2. Sample YAML for a Cluster API machine template resource on VMware vSphere

The machine template resource is provider-specific and defines the basic properties of the machines that a compute machine set creates. The compute machine set references this template when creating machines.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: VSphereMachineTemplate 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      template: <vm_template_name> 4
      server: <vcenter_server_ip> 5
      diskGiB: 128
      cloneMode: linkedClone 6
      datacenter: <vcenter_data_center_name> 7
      datastore: <vcenter_datastore_name> 8
      folder: <vcenter_vm_folder_path> 9
      resourcePool: <vsphere_resource_pool> 10
      numCPUs: 4
      memoryMiB: 16384
      network:
        devices:
          - dhcp4: true
        networkName: "<vm_network_name>" 11
```

**1** Specify the machine template kind. This value must match the value for your platform.

**2** Specify a name for the machine template.

**3** Specify the details for your environment. The values here are examples.

**4** Specify the vSphere VM template to use, such as **user-5ddjd-rhcos**.

**5** Specify the vCenter server IP or fully qualified domain name.

**6** Specify the type of VM clone to use. The following values are valid:

- **fullClone**
- **linkedClone**

When using the **linkedClone** type, the disk size matches the clone source instead of using the **diskGiB** value. For more information, see the vSphere documentation about VM clone types.

**7** Specify the vCenter data center to deploy the compute machine set on.

**8** Specify the vCenter datastore to deploy the compute machine set on.

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

**10** Specify the vSphere resource pool for your VMs.

**11**

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

#### 13.5.3.1.3. Sample YAML for a Cluster API compute machine set resource on VMware vSphere

The compute machine set resource defines additional properties of the machines that it creates. The compute machine set also references the infrastructure resource and machine template when creating machines.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> ①
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> ②
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data ③
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: VSphereMachineTemplate ④
        name: <template_name> ⑤
      failureDomain: ⑥
        - name: <failure_domain_name>
          region: <region_a>
          zone: <zone_a>
          server: <vcenter_server_name>
          topology:
            datacenter: <region_a_data_center>
            computeCluster: "</region_a_data_center/host/zone_a_cluster>"
            resourcePool: "</region_a_data_center/host/zone_a_cluster/Resources/resource_pool>"
            datastore: "</region_a_data_center/datastore/datastore_a>"
          networks:
            - port-group

```

- ① Specify a name for the compute machine set.
- ② Specify the cluster ID as the name of the cluster.
- ③ For the Cluster API Technology Preview, the Operator can use the worker user data secret from the **openshift-machine-api** namespace.
- ④ Specify the machine template kind. This value must match the value for your platform.

- 5 Specify the machine template name.
- 6 Specify the failure domain configuration details.



#### NOTE

Using multiple regions and zones on a vSphere cluster that uses the Cluster API is not a validated configuration.

## 13.6. TROUBLESHOOTING CLUSTERS THAT USE THE CLUSTER API



#### IMPORTANT

Managing machines with the Cluster API is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

Use the information in this section to understand and recover from issues you might encounter. Generally, troubleshooting steps for problems with the Cluster API are similar to those steps for problems with the Machine API.

The Cluster CAPI Operator and its operands are provisioned in the **openshift-cluster-api** namespace, whereas the Machine API uses the **openshift-machine-api** namespace. When using **oc** commands that reference a namespace, be sure to reference the correct one.

### 13.6.1. Referencing the intended objects when using the CLI

For clusters that use the Cluster API, OpenShift CLI (**oc**) commands prioritize Cluster API objects over Machine API objects.

This behavior impacts any **oc** command that acts upon any object that is represented in both the Cluster API and the Machine API. This explanation uses the **oc delete machine** command, which deletes a machine, as an example.

#### Cause

When you run an **oc** command, **oc** communicates with the Kube API server to determine which objects to act upon. The Kube API server uses the first installed custom resource definition (CRD) it encounters alphabetically when an **oc** command is run.

CRDs for Cluster API objects are in the **cluster.x-k8s.io** group, while CRDs for Machine API objects are in the **machine.openshift.io** group. Because the letter **c** precedes the letter **m** alphabetically, the Kube API server matches on the Cluster API object CRD. As a result, the **oc** command acts upon Cluster API objects.

#### Consequences

Due to this behavior, the following unintended outcomes can occur on a cluster that uses the Cluster API:

- For namespaces that contain both types of objects, commands such as **oc get machine** return only Cluster API objects.
- For namespaces that contain only Machine API objects, commands such as **oc get machine** return no results.

## Workaround

You can ensure that **oc** commands act on the type of objects you intend by using the corresponding fully qualified name.

## Prerequisites

- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

## Procedure

- To delete a Machine API machine, use the fully qualified name **machine.machine.openshift.io** when running the **oc delete machine** command:

```
$ oc delete machine.machine.openshift.io <machine_name>
```

- To delete a Cluster API machine, use the fully qualified name **machine.cluster.x-k8s.io** when running the **oc delete machine** command:

```
$ oc delete machine.cluster.x-k8s.io <machine_name>
```

# CHAPTER 14. DEPLOYING MACHINE HEALTH CHECKS

You can configure and deploy a machine health check to automatically repair damaged machines in a machine pool.



## IMPORTANT

You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational. Clusters with user-provisioned infrastructure require additional validation and configuration to use the Machine API.

Clusters with the infrastructure platform type **none** cannot use the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that supports the feature. This parameter cannot be changed after installation.

To view the platform type for your cluster, run the following command:

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

## 14.1. ABOUT MACHINE HEALTH CHECKS



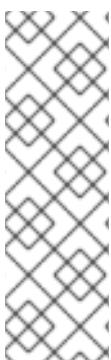
## NOTE

You can only apply a machine health check to machines that are managed by compute machine sets or control plane machine sets.

To monitor machine health, create a resource to define the configuration for a controller. Set a condition to check, such as staying in the **NotReady** status for five minutes or displaying a permanent condition in the node-problem-detector, and a label for the set of machines to monitor.

The controller that observes a **MachineHealthCheck** resource checks for the defined condition. If a machine fails the health check, the machine is automatically deleted and one is created to take its place. When a machine is deleted, you see a **machine deleted** event.

To limit disruptive impact of the machine deletion, the controller drains and deletes only one node at a time. If there are more unhealthy machines than the **maxUnhealthy** threshold allows for in the targeted pool of machines, remediation stops and therefore enables manual intervention.



## NOTE

Consider the timeouts carefully, accounting for workloads and requirements.

- Long timeouts can result in long periods of downtime for the workload on the unhealthy machine.
- Too short timeouts can result in a remediation loop. For example, the timeout for checking the **NotReady** status must be long enough to allow the machine to complete the startup process.

To stop the check, remove the resource.

### 14.1.1. Limitations when deploying machine health checks

There are limitations to consider before deploying a machine health check:

- Only machines owned by a machine set are remediated by a machine health check.
- If the node for a machine is removed from the cluster, a machine health check considers the machine to be unhealthy and remediates it immediately.
- If the corresponding node for a machine does not join the cluster after the **nodeStartupTimeout**, the machine is remediated.
- A machine is remediated immediately if the **Machine** resource phase is **Failed**.

#### Additional resources

- [About listing all the nodes in a cluster](#)
- [Short-circuiting machine health check remediation](#)
- [About the Control Plane Machine Set Operator](#)

## 14.2. SAMPLE MACHINEHEALTHCHECK RESOURCE

The **MachineHealthCheck** resource for all cloud-based installation types, and other than bare metal, resembles the following YAML file:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example 1
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> 2
      machine.openshift.io/cluster-api-machine-type: <role> 3
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 4
  unhealthyConditions:
    - type: "Ready"
      timeout: "300s" 5
      status: "False"
    - type: "Ready"
      timeout: "300s" 6
      status: "Unknown"
  maxUnhealthy: "40%" 7
  nodeStartupTimeout: "10m" 8
```

**1** Specify the name of the machine health check to deploy.

**2** **3** Specify a label for the machine pool that you want to check.

**4** Specify the machine set to track in **<cluster\_name>-<label>-<zone>** format. For example, **prod-node-us-east-1a**.

- 5 Specify the timeout duration for a node condition. If a condition is met for the duration of the timeout, the machine will be remediated. Long timeouts can result in long periods of downtime for machines.
- 6 Specify the amount of machines allowed to be concurrently remediated in the targeted pool. This can be set as a percentage or an integer. If the number of unhealthy machines exceeds the limit set by **maxUnhealthy**, remediation is not performed.
- 7 Specify the timeout duration that a machine health check must wait for a node to join the cluster before a machine is determined to be unhealthy.



#### NOTE

The **matchLabels** are examples only; you must map your machine groups based on your specific needs.

### 14.2.1. Short-circuiting machine health check remediation

Short-circuiting ensures that machine health checks remediate machines only when the cluster is healthy. Short-circuiting is configured through the **maxUnhealthy** field in the **MachineHealthCheck** resource.

If the user defines a value for the **maxUnhealthy** field, before remediating any machines, the **MachineHealthCheck** compares the value of **maxUnhealthy** with the number of machines within its target pool that it has determined to be unhealthy. Remediation is not performed if the number of unhealthy machines exceeds the **maxUnhealthy** limit.



#### IMPORTANT

If **maxUnhealthy** is not set, the value defaults to **100%** and the machines are remediated regardless of the state of the cluster.

The appropriate **maxUnhealthy** value depends on the scale of the cluster you deploy and how many machines the **MachineHealthCheck** covers. For example, you can use the **maxUnhealthy** value to cover multiple compute machine sets across multiple availability zones so that if you lose an entire zone, your **maxUnhealthy** setting prevents further remediation within the cluster. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability.



#### IMPORTANT

If you configure a **MachineHealthCheck** resource for the control plane, set the value of **maxUnhealthy** to **1**.

This configuration ensures that the machine health check takes no action when multiple control plane machines appear to be unhealthy. Multiple unhealthy control plane machines can indicate that the etcd cluster is degraded or that a scaling operation to replace a failed machine is in progress.

If the etcd cluster is degraded, manual intervention might be required. If a scaling operation is in progress, the machine health check should allow it to finish.

The **maxUnhealthy** field can be set as either an integer or percentage. There are different remediation implementations depending on the **maxUnhealthy** value.

### 14.2.1.1. Setting maxUnhealthy by using an absolute value

If **maxUnhealthy** is set to **2**:

- Remediation will be performed if 2 or fewer nodes are unhealthy
- Remediation will not be performed if 3 or more nodes are unhealthy

These values are independent of how many machines are being checked by the machine health check.

### 14.2.1.2. Setting maxUnhealthy by using percentages

If **maxUnhealthy** is set to **40%** and there are 25 machines being checked:

- Remediation will be performed if 10 or fewer nodes are unhealthy
- Remediation will not be performed if 11 or more nodes are unhealthy

If **maxUnhealthy** is set to **40%** and there are 6 machines being checked:

- Remediation will be performed if 2 or fewer nodes are unhealthy
- Remediation will not be performed if 3 or more nodes are unhealthy



#### NOTE

The allowed number of machines is rounded down when the percentage of **maxUnhealthy** machines that are checked is not a whole number.

## 14.3. CREATING A MACHINE HEALTH CHECK RESOURCE

You can create a **MachineHealthCheck** resource for machine sets in your cluster.



#### NOTE

You can only apply a machine health check to machines that are managed by compute machine sets or control plane machine sets.

#### Prerequisites

- Install the **oc** command line interface.

#### Procedure

1. Create a **healthcheck.yml** file that contains the definition of your machine health check.
2. Apply the **healthcheck.yml** file to your cluster:

```
$ oc apply -f healthcheck.yml
```

You can configure and deploy a machine health check to detect and repair unhealthy bare metal nodes.

## 14.4. ABOUT POWER-BASED REMEDIATION OF BARE METAL

In a bare metal cluster, remediation of nodes is critical to ensuring the overall health of the cluster. Physically remediating a cluster can be challenging and any delay in putting the machine into a safe or an operational state increases the time the cluster remains in a degraded state, and the risk that subsequent failures might bring the cluster offline. Power-based remediation helps counter such challenges.

Instead of reprovisioning the nodes, power-based remediation uses a power controller to power off an inoperable node. This type of remediation is also called power fencing.

OpenShift Container Platform uses the **MachineHealthCheck** controller to detect faulty bare metal nodes. Power-based remediation is fast and reboots faulty nodes instead of removing them from the cluster.

Power-based remediation provides the following capabilities:

- Allows the recovery of control plane nodes
- Reduces the risk of data loss in hyperconverged environments
- Reduces the downtime associated with recovering physical machines

#### 14.4.1. MachineHealthChecks on bare metal

Machine deletion on bare metal cluster triggers reprovisioning of a bare metal host. Usually bare metal reprovisioning is a lengthy process, during which the cluster is missing compute resources and applications might be interrupted.

There are two ways to change the default remediation process from machine deletion to host power-cycle:

1. Annotate the **MachineHealthCheck** resource with the **machine.openshift.io/remediation-strategy: external-baremetal** annotation.
2. Create a **Metal3RemediationTemplate** resource, and refer to it in the **spec.remediationTemplate** of the **MachineHealthCheck**.

After using one of these methods, unhealthy machines are power-cycled by using Baseboard Management Controller (BMC) credentials.

#### 14.4.2. Understanding the annotation-based remediation process

The remediation process operates as follows:

1. The MachineHealthCheck (MHC) controller detects that a node is unhealthy.
2. The MHC notifies the bare metal machine controller which requests to power-off the unhealthy node.
3. After the power is off, the node is deleted, which allows the cluster to reschedule the affected workload on other nodes.
4. The bare metal machine controller requests to power on the node.
5. After the node is up, the node re-registers itself with the cluster, resulting in the creation of a new node.

- After the node is recreated, the bare metal machine controller restores the annotations and labels that existed on the unhealthy node before its deletion.



#### NOTE

If the power operations did not complete, the bare metal machine controller triggers the reprovisioning of the unhealthy node unless this is a control plane node or a node that was provisioned externally.

### 14.4.3. Understanding the metal3-based remediation process

The remediation process operates as follows:

- The MachineHealthCheck (MHC) controller detects that a node is unhealthy.
- The MHC creates a metal3 remediation custom resource for the metal3 remediation controller, which requests to power-off the unhealthy node.
- After the power is off, the node is deleted, which allows the cluster to reschedule the affected workload on other nodes.
- The metal3 remediation controller requests to power on the node.
- After the node is up, the node re-registers itself with the cluster, resulting in the creation of a new node.
- After the node is recreated, the metal3 remediation controller restores the annotations and labels that existed on the unhealthy node before its deletion.



#### NOTE

If the power operations did not complete, the metal3 remediation controller triggers the reprovisioning of the unhealthy node unless this is a control plane node or a node that was provisioned externally.

### 14.4.4. Creating a MachineHealthCheck resource for bare metal

#### Prerequisites

- The OpenShift Container Platform is installed using installer-provisioned infrastructure (IPI).
- Access to BMC credentials (or BMC access to each node).
- Network access to the BMC interface of the unhealthy node.

#### Procedure

- Create a **healthcheck.yaml** file that contains the definition of your machine health check.
- Apply the **healthcheck.yaml** file to your cluster using the following command:

```
$ oc apply -f healthcheck.yaml
```

#### Sample MachineHealthCheck resource for bare metal, annotation-based remediation

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example 1
  namespace: openshift-machine-api
  annotations:
    machine.openshift.io/remediation-strategy: external-baremetal 2
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> 3
      machine.openshift.io/cluster-api-machine-type: <role> 4
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 5
  unhealthyConditions:
    - type: "Ready"
      timeout: "300s" 6
      status: "False"
    - type: "Ready"
      timeout: "300s" 7
      status: "Unknown"
  maxUnhealthy: "40%" 8
  nodeStartupTimeout: "10m" 9

```

- 1** Specify the name of the machine health check to deploy.
- 2** For bare metal clusters, you must include the **machine.openshift.io/remediation-strategy: external-baremetal** annotation in the **annotations** section to enable power-cycle remediation. With this remediation strategy, unhealthy hosts are rebooted instead of removed from the cluster.
- 3** **4** Specify a label for the machine pool that you want to check.
- 5** Specify the compute machine set to track in **<cluster\_name>-<label>-<zone>** format. For example, **prod-node-us-east-1a**.
- 6** **7** Specify the timeout duration for the node condition. If the condition is met for the duration of the timeout, the machine will be remediated. Long timeouts can result in long periods of downtime for a workload on an unhealthy machine.
- 8** Specify the amount of machines allowed to be concurrently remediated in the targeted pool. This can be set as a percentage or an integer. If the number of unhealthy machines exceeds the limit set by **maxUnhealthy**, remediation is not performed.
- 9** Specify the timeout duration that a machine health check must wait for a node to join the cluster before a machine is determined to be unhealthy.



#### NOTE

The **matchLabels** are examples only; you must map your machine groups based on your specific needs.

#### Sample MachineHealthCheck resource for bare metal, metal3-based remediation

```
apiVersion: machine.openshift.io/v1beta1
```

```

kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone>
remediationTemplate:
  apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
  kind: Metal3RemediationTemplate
  name: metal3-remediation-template
  namespace: openshift-machine-api
unhealthyConditions:
- type: "Ready"
  timeout: "300s"

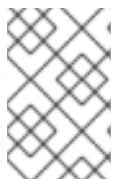
```

### Sample Metal3RemediationTemplate resource for bare metal, metal3-based remediation

```

apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3RemediationTemplate
metadata:
  name: metal3-remediation-template
  namespace: openshift-machine-api
spec:
  template:
    spec:
      strategy:
        type: Reboot
        retryLimit: 1
        timeout: 5m0s

```



#### NOTE

The **matchLabels** are examples only; you must map your machine groups based on your specific needs. The **annotations** section does not apply to metal3-based remediation. Annotation-based remediation and metal3-based remediation are mutually exclusive.

#### 14.4.5. Troubleshooting issues with power-based remediation

To troubleshoot an issue with power-based remediation, verify the following:

- You have access to the BMC.
- BMC is connected to the control plane node that is responsible for running the remediation task.