OpenShift Container Platform 4.5

Machine management

Adding and maintaining cluster machines

Last Updated: 2021-07-26

OpenShift Container Platform 4.5 Machine management

Adding and maintaining cluster machines

Legal Notice

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

CHAPTER 1. CREATING MACHINE SETS

1.1. CREATING A MACHINE SET ON AWS

You can create a different 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.

1.1.1. 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 4.5 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.5 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

1.1.2. Sample YAML for a machine set custom resource on AWS

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

In this sample, **<infrastructureID>** 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: <infrastructureID> ❶
  name: <infrastructureID>-<role>-<zone> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> ❸
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<zone> ❹
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> ❺
        machine.openshift.io/cluster-api-machine-role: <role> ❻
        machine.openshift.io/cluster-api-machine-type: <role> ❼
        machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<zone> ❽
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" ❾
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 ❿
          apiVersion: awsproviderconfig.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
          credentialsSecret:
            name: aws-cloud-credentials
          deviceIndex: 0
          iamInstanceProfile:
            id: <infrastructureID>-worker-profile ⓫
          instanceType: m4.large
          kind: AWSMachineProviderConfig
          placement:
```

```
  availabilityZone: us-east-1a
  region: us-east-1
securityGroups:
  - filters:
      - name: tag:Name
        values:
          - <infrastructureID>-worker-sg 12
subnet:
  filters:
    - name: tag:Name
      values:
        - <infrastructureID>-private-us-east-1a 13
tags:
  - name: kubernetes.io/cluster/<infrastructureID> 14
    value: owned
userDataSecret:
  name: worker-user-data
```

1 3 5 11 12 13 14 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 -n openshift-machine-api \
    -o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}{"\n"}' \
    get machineset/<infrastructureID>-worker-us-east-1a
```

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

6 7 9 Specify the node label to add.

10 Specify a valid Red Hat Enterprise Linux CoreOS (RHCOS) AMI for your AWS zone for your OpenShift Container Platform nodes.

1.1.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

Example output

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

b. Check values of a specific machine set:

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

Example output

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

1 The cluster ID.

2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

Next steps

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

1.1.4. Machine sets that deploy machines as Spot Instances

You can save on costs by creating a machine set running on AWS that deploys machines as non-guaranteed Spot Instances. Spot Instances use available 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.

> IMPORTANT
>
> It is strongly recommended that control plane machines are not created on Spot Instances due to the increased likelihood of the instance being terminated. Manual intervention is required to replace a terminated control plane node.

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 machine set **replicas** quantity, the machine set creates a machine that requests a Spot Instance.

1.1.5. Creating Spot Instances by using machine sets

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

Procedure

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

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

Optional: You can 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.

1.2. CREATING A MACHINE SET ON AZURE

You can create a different 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.

1.2.1. Machine API overview

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

For OpenShift Container Platform 4.5 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.5 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

1.2.2. Sample YAML for a machine set custom resource on Azure

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

In this sample, **<infrastructureID>** 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: <infrastructureID> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructureID>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-<role>-<region> 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" 11
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructureID>-
rg/providers/Microsoft.Compute/images/<infrastructureID>
            sku: ""
  version: ""
internalLoadBalancer: ""
kind: AzureMachineProviderSpec
location: centralus
managedIdentity: <infrastructureID>-identity 12
metadata:
  creationTimestamp: null
natRule: null
networkResourceGroup: ""
osDisk:
  diskSizeGB: 128
  managedDisk:
    storageAccountType: Premium_LRS
  osType: Linux
publicIP: false
publicLoadBalancer: ""
resourceGroup: <infrastructureID>-rg 13
sshPrivateKey: ""
sshPublicKey: ""
subnet: <infrastructureID>-<role>-subnet 14 15
userDataSecret:
  name: worker-user-data 16
vmSize: qeci-22538-vnet
vnet: <infrastructureID>-vnet 17
zone: "1" 18
```

1 5 7 12 13 14 17 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/<infrastructureID>-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/<infrastructureID>-worker-centralus1
```

2 3 8 9 11 15 16 Specify the node label to add.

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

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

1.2.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

   #### Example output

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

   b. Check values of a specific machine set:

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

   #### Example output

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

   1 The cluster ID.

   2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

1.3. CREATING A MACHINE SET ON GCP

You can create a different 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.

1.3.1. Machine API overview

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

For OpenShift Container Platform 4.5 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.5 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

1.3.2. Sample YAML for a machine set custom resource on GCP

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

In this sample, **<infrastructureID>** 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: <infrastructureID> 1
  name: <infrastructureID>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 9
      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> 10
            labels: null
            sizeGb: 128
            type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructureID>-network 11
            subnetwork: <infrastructureID>-worker-subnet 12
          projectID: <project_name> 13
          region: us-central1
          serviceAccounts:
          - email: <infrastructureID>-w@<project_name>.iam.gserviceaccount.com 14 15
            scopes:
            - https://www.googleapis.com/auth/cloud-platform
          tags:
          - <infrastructureID>-worker 16
          userDataSecret:
            name: worker-user-data
          zone: us-central1-a
```

1 2 3 4 5 8 11 12 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
```

6 7 9 Specify the node label to add.

10 Specify the path to the image that is used in current machine sets. 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/<infrastructureID>-worker-a
```

13 15 Specify the name of the GCP project that you use for your cluster.

1.3.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

   #### Example output

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

   b. Check values of a specific machine set:

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

   #### Example output

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

   1 The cluster ID.

   2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                               DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

1.4. CREATING A MACHINE SET ON OPENSTACK

You can create a different 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.

1.4.1. 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 4.5 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.5 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

1.4.2. Sample YAML for a machine set custom resource on RHOSP

This sample YAML defines a 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 **node_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: <node_role> 2
    machine.openshift.io/cluster-api-machine-type: <node_role> 3
  name: <infrastructure_ID>-<node_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>-<node_role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 7
        machine.openshift.io/cluster-api-machine-role: <node_role> 8
        machine.openshift.io/cluster-api-machine-type: <node_role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role> 10
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.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>
          primarySubnet: <rhosp_subnet_UUID> 13
          securityGroups:
          - filter: {}
            name: <infrastructure_ID>-worker
          serverMetadata:
            Name: <infrastructure_ID>-worker
            openshiftClusterID: <infrastructure_ID>
          tags:
          - openshiftClusterID=<infrastructure_ID>
          trunk: true
          userDataSecret:
            name: worker-user-data 14
          availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 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 14 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. If deploying to multiple networks, this list 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.

1.4.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

   #### Example output

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

   b. Check values of a specific machine set:

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

   #### Example output

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

   1 The cluster ID.

   2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

1.5. CREATING A MACHINE SET ON RHV

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

1.5.1. 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 4.5 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.5 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales machines in a cloud. You can set the minimum and maximum scaling boundaries for nodes in a specified machine set, and the machine autoscaler maintains that range of nodes. The **MachineAutoscaler** object takes effect after a

ClusterAutoscaler object exists. Both **ClusterAutoscaler** and **MachineAutoscaler** resources are made available by the **ClusterAutoscalerOperator** object.

Cluster autoscaler

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

Machine health check

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

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

1.5.2. Sample YAML for a machine set custom resource on RHV

This sample YAML defines a machine set that runs on RHV and creates nodes that are labeled with **node-role.kubernetes.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 **<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> 5
  Selector: 6
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
        machine.openshift.io/cluster-api-machine-role: <role> 10
        machine.openshift.io/cluster-api-machine-type: <role> 11
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
    spec:
      metadata:
```

```
      labels:
        node-role.kubernetes.io/<role>: "" 13
    providerSpec:
      value:
        apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
        cluster_id: <ovirt_cluster_id> 14
        template_name: <ovirt_template_name> 15
        instance_type_id: <instance_type_id> 16
        cpu: 17
          sockets: <number_of_sockets> 18
          cores: <number_of_cores> 19
          threads: <number_of_threads> 20
        memory_mb: <memory_size> 21
        os_disk: 22
          size_gb: <disk_size> 23
        network_interfaces: 24
          vnic_profile_id:  <vnic_profile_id> 25
        credentialsSecret:
          name: ovirt-credentials 26
        kind: OvirtMachineProviderSpec
        type: <workload_type> 27
        userDataSecret:
          name: worker-user-data
```

1 7 9 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 3 10 11 13 Specify the node label to add.

4 8 12 Specify the infrastructure ID and node label. These two strings together cannot be longer than 35 characters.

5 Specify the number of machines to create.

6 Selector for the machines.

14 Specify the UUID for the RHV cluster to which this VM instance belongs.

15 Specify the RHV VM template to use to create the machine.

16 Optional: Specify the VM instance type. If you include this parameter, you do not need to specify the hardware parameters of the VM including CPU and memory because this parameter overrides all hardware parameters.

17 Optional: The CPU field contains the CPU’s configuration, including sockets, cores, and threads.

18 Optional: Specify the number of sockets for a VM.

19 Optional: Specify the number of cores per socket.

20 Optional: Specify the number of threads per core.

21 Optional: Specify the size of a VM’s memory in MiB.

22 Optional: Root disk of the node.

23 Optional: Specify the size of the bootable disk in GiB.

24 Optional: List of the network interfaces of the VM. If you include this parameter, OpenShift Container Platform discards all network interfaces from the template and creates new ones.

25 Optional: Specify the vNIC profile ID.

26 Specify the name of the secret that holds the RHV credentials.

27 Optional: Specify the workload type for which the instance is optimized. This value affects the **RHV VM** parameter. Supported values: **desktop**, **server**, **high_performance**.

NOTE

Because RHV uses a template when creating a VM, if you do not specify a value for an optional parameter, RHV uses the value for that parameter that is specified in the template.

1.5.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

   Example output

   ```
   NAME                        DESIRED   CURRENT   READY   AVAILABLE   AGE
   agl030519-vplxk-worker-us-east-1a  1       1        1       1          55m
   agl030519-vplxk-worker-us-east-1b  1       1        1       1          55m
   agl030519-vplxk-worker-us-east-1c  1       1        1       1          55m
   ```

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

b. Check values of a specific machine set:

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

Example output

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

1 The cluster ID.

2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

1.6. CREATING A MACHINE SET ON VSPHERE

You can create a different 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.

1.6.1. 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 4.5 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.5 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

1.6.2. Sample YAML for a machine set custom resource on vSphere

This sample YAML defines a 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> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❸
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> ❹
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❺
        machine.openshift.io/cluster-api-machine-role: <role> ❻
        machine.openshift.io/cluster-api-machine-type: <role> ❼
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> ❽
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" ❾
      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>" ❿
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> ⓫
          userDataSecret:
            name: worker-user-data
          workspace:
            datacenter: <vcenter_datacenter_name> ⓬
            datastore: <vcenter_datastore_name> ⓭
        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 machine set to.

11 Specify the vSphere VM clone of the template to use, such as **user-5ddjd-rhcos**.

IMPORTANT

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

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

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

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

15 Specify the vSphere resource pool for your VMs.

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

1.6.3. Creating a machine set

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

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.
- Create a tag inside your vCenter instance based on the cluster API name. This tag is utilized by the machine set to associate the OpenShift Container Platform nodes to the provisioned virtual machines (VM). For directions on creating tags in vCenter, see the VMware documentation for vSphere Tags and Attributes.

- Have the necessary permissions to deploy VMs in your vCenter instance and have the required access to the datastore specified.

Procedure

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

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

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

   #### Example output

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

   b. Check values of a specific machine set:

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

   #### Example output

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

   1 The cluster ID.

   2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

CHAPTER 2. MANUALLY SCALING A MACHINE SET

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

NOTE

If you need to modify aspects of a machine set outside of scaling, see Modifying a machine set.

2.1. PREREQUISITES

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

IMPORTANT

This process is not applicable to clusters where you manually provisioned the machines yourself. You can use the advanced machine management and scaling capabilities only in clusters where the machine API is operational.

2.2. SCALING A MACHINE SET MANUALLY

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

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

Prerequisites

- Install an OpenShift Container Platform cluster and the **oc** command line.

- Log in to **oc** as a user with **cluster-admin** permission.

Procedure

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

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

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

2. Scale the machine set:

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

Or:

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

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

2.3. THE 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 machine sets down. The deletion policy can be set according to the use case by modifying the particular 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/cluster-api-delete-machine** 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 machine set to **0** unless you first relocate the router pods.

NOTE

Custom 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 machine sets are scaling down. This prevents service disruption.

CHAPTER 3. MODIFYING A MACHINE SET

You can make changes to a machine set, such as adding labels, changing the instance type, or changing block storage.

NOTE

If you need to scale a machine set without making other changes, see Manually scaling a machine set.

3.1. MODIFYING A MACHINE SET

To make changes to a machine set, edit the **MachineSet** YAML. Then, remove all machines associated with the machine set by deleting each machine or scaling down the machine set to **0** replicas. Then, scale the replicas back to the desired number. Changes you make to a machine set do not affect existing machines.

If you need to scale a 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 workers. Because the router is required to access some cluster resources, including the web console, do not scale the worker machine set to **0** unless you first relocate the router pods.

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. Edit the machine set:

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

2. Scale down the machine set to **0**:

```
$ oc scale --replicas=0 machineset <machineset> -n openshift-machine-api
```

Or:

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

Wait for the machines to be removed.

3. Scale up the machine set as needed:

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

Or:

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

Wait for the machines to start. The new machines contain changes you made to the machine set.

CHAPTER 4. DELETING A MACHINE

You can delete a specific machine.

4.1. DELETING A SPECIFIC MACHINE

You can delete a specific machine.

Prerequisites

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

Procedure

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

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

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

2. Delete the machine:

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

   IMPORTANT

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

CHAPTER 5. 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 is operational.

5.1. 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 failed to schedule on any of the current 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.

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.

The cluster autoscaler decreases the size of the cluster when some nodes are consistently not needed for a significant period, such as when it has low resource use and all of its important pods can fit on other nodes.

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.

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.

5.2. ABOUT THE MACHINE AUTOSCALER

The machine autoscaler adjusts the number of Machines in the machine sets that you deploy in an OpenShift Container Platform cluster. You can scale both the default **worker** machine set and any other 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 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 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.

5.3. 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.

5.3.1. ClusterAutoscaler 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 ⑩
        min: 0 ⑪
        max: 4 ⑫
  scaleDown: ⑬
    enabled: true ⑭
    delayAfterAdd: 10m ⑮
    delayAfterDelete: 5m ⑯
    delayAfterFailure: 30s ⑰
    unneededTime: 5m 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 10 Optionally, specify the type of GPU node to deploy. Only **nvidia.com/gpu** and **amd.com/gpu** are valid types.

8 11 Specify the minimum number of GPUs to deploy in the cluster.

9 12 Specify the maximum number of GPUs to deploy in the cluster.

13 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**.

14 Specify whether the cluster autoscaler can remove unnecessary nodes.

15 Optionally, 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.

16 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 **10s** is used.

17 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.

18 Specify the period before an unnecessary node is eligible for deletion. If you do not specify a value, the default value of **10m** is used.

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.

5.3.2. Deploying the cluster autoscaler

To deploy the cluster autoscaler, you create an instance of the **ClusterAutoscaler** resource.

Procedure

1. Create a YAML file for the **ClusterAutoscaler** resource that contains the customized resource definition.

2. Create the resource in the cluster:

```
$ oc create -f <filename>.yaml 1
```

1 **<filename>** is the name of the resource file that you customized.

5.4. NEXT STEPS

- After you configure the cluster autoscaler, you must configure at least one machine autoscaler.

5.5. CONFIGURING THE MACHINE AUTOSCALERS

After you deploy the cluster autoscaler, deploy **MachineAutoscaler** resources that reference the 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 machine set. Remember that machine sets are different in each region, so consider whether you want to enable machine scaling in multiple regions. The machine set that you scale must have at least one machine in it.

5.5.1. MachineAutoscaler 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 machine set this machine autoscaler scales, specify or include the name of the machine set to scale. The machine set name takes the following form: **<clusterid>-<machineset>-<aws-region-az>**

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, or Azure, this value can be set to **0**. For other providers, do not set this value to **0**.

③ Specify the maximum number machines of the specified type that the cluster autoscaler can deploy in the specified AWS 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 machine set to scale.

⑤ The **kind** parameter value is always **MachineSet**.

⑥ The **name** value must match the name of an existing machine set, as shown in the **metadata.name** parameter value.

5.5.2. Deploying the machine autoscaler

To deploy the machine autoscaler, you create an instance of the **MachineAutoscaler** resource.

Procedure

1. Create a YAML file for the **MachineAutoscaler** resource that contains the customized resource definition.

2. Create the resource in the cluster:

```
$ oc create -f <filename>.yaml 1
```

1 **<filename>** is the name of the resource file that you customized.

5.6. ADDITIONAL RESOURCES

- For more information about pod priority, see Including pod priority in pod scheduling decisions in OpenShift Container Platform.

CHAPTER 6. CREATING INFRASTRUCTURE MACHINE SETS

You can create a machine set to host only infrastructure components. You apply specific Kubernetes labels to these machines and then update the infrastructure components to run on only those machines. These infrastructure nodes are not counted toward the total number of subscriptions that are required to run the environment.

IMPORTANT

Unlike earlier versions of OpenShift Container Platform, you cannot move the infrastructure components to the master machines. To move the components, you must create a new machine set.

6.1. OPENSHIFT CONTAINER PLATFORM INFRASTRUCTURE COMPONENTS

The following infrastructure workloads do not incur OpenShift Container Platform worker subscriptions:

- Kubernetes and OpenShift Container Platform control plane services that run on masters
- The default router
- The integrated container image registry
- The cluster metrics collection, or monitoring service, including components for monitoring user-defined projects
- Cluster aggregated logging
- Service brokers
- Red Hat Quay
- Red Hat OpenShift Container Storage
- Red Hat Advanced Cluster Manager

Any node that runs any other container, pod, or component is a worker node that your subscription must cover.

6.2. CREATING INFRASTRUCTURE MACHINE SETS FOR PRODUCTION ENVIRONMENTS

In a production deployment, deploy at least three machine sets to hold infrastructure components. Both the logging aggregation solution and the service mesh deploy Elasticsearch, and Elasticsearch requires three instances that are installed on different nodes. For high availability, deploy these nodes to different availability zones. Since you need different machine sets for each availability zone, create at least three machine sets.

6.2.1. Creating machine sets for different clouds

Use the sample machine set for your cloud.

6.2.1.1. Sample YAML for a machine set custom resource on AWS

This sample YAML defines a machine set that runs in the **us-east-1a** Amazon Web Services (AWS) zone and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructureID>** 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: <infrastructureID> ①
  name: <infrastructureID>-infra-<zone> ②
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> ③
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-infra-<zone> ④
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-infra-<zone> ⑧
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: "" ⑨
      taints: ⑩
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 ⑪
          apiVersion: awsproviderconfig.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
          credentialsSecret:
            name: aws-cloud-credentials
          deviceIndex: 0
          iamInstanceProfile:
            id: <infrastructureID>-worker-profile ⑫
          instanceType: m4.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: us-east-1a
          region: us-east-1
         securityGroups:
          - filters:
            - name: tag:Name
             values:
              - <infrastructureID>-worker-sg 13
         subnet:
          filters:
           - name: tag:Name
            values:
             - <infrastructureID>-private-us-east-1a 14
         tags:
          - name: kubernetes.io/cluster/<infrastructureID> 15
           value: owned
         userDataSecret:
          name: worker-user-data
```

1 3 5 12 13 14 15 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 -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}{"\n"}' \
  get machineset/<infrastructureID>-worker-us-east-1a
```

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

6 7 9 Specify the **infra** node label.

10 Specify a taint to prevent user workloads from being scheduled on infra nodes.

11 Specify a valid Red Hat Enterprise Linux CoreOS (RHCOS) AMI for your AWS zone for your OpenShift Container Platform nodes.

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.

6.2.1.2. Sample YAML for a machine set custom resource on Azure

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

In this sample, **<infrastructureID>** 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: <infrastructureID> 1
  machine.openshift.io/cluster-api-machine-role: infra 2
  machine.openshift.io/cluster-api-machine-type: infra 3
 name: <infrastructureID>-infra-<region> 4
```

```
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-infra-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-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: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructureID>-
rg/providers/Microsoft.Compute/images/<infrastructureID>
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: <infrastructureID>-identity 13
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: ""
          resourceGroup: <infrastructureID>-rg 14
          sshPrivateKey: ""
          sshPublicKey: ""
          subnet: <infrastructureID>-<role>-subnet 15 16
          userDataSecret:
            name: worker-user-data 17
          vmSize: qeci-22538-vnet
          vnet: <infrastructureID>-vnet 18
          zone: "1" 19
```

1 5 7 13 14 15 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
```

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/<infrastructureID>-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/<infrastructureID>-worker-centralus1
```

2 3 8 9 11 16 17 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.

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

6.2.1.3. Sample YAML for a machine set custom resource on GCP

This sample YAML defines a machine set that runs in Google Cloud Platform (GCP) and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

In this sample, **<infrastructureID>** 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: <infrastructureID> 1
  name: <infrastructureID>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 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: <infrastructureID>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      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> 11
            labels: null
            sizeGb: 128
            type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructureID>-network 12
            subnetwork: <infrastructureID>-worker-subnet 13
          projectID: <project_name> 14
          region: us-central1
          serviceAccounts:
          - email: <infrastructureID>-w@<project_name>.iam.gserviceaccount.com 15 16
            scopes:
            - https://www.googleapis.com/auth/cloud-platform
          tags:
          - <infrastructureID>-worker 17
          userDataSecret:
            name: worker-user-data
          zone: us-central1-a
```

1 2 3 4 5 8 12 13 15 17 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
```

6 7 9 Specify the **infra** node label.

10 Specify a taint to prevent user workloads from being scheduled on infra nodes.

11 Specify the path to the image that is used in current machine sets. 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/<infrastructureID>-worker-a
```

14 16 Specify the name of the GCP project that you use for your cluster.

6.2.1.4. Sample YAML for a machine set custom resource on RHOSP

This sample YAML defines a 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: openstackproviderconfig.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>
          primarySubnet: <rhosp_subnet_UUID> 14
          securityGroups:
          - filter: {}
            name: <infrastructure_ID>-worker
          serverMetadata:
            Name: <infrastructure_ID>-worker
            openshiftClusterID: <infrastructure_ID>
          tags:
          - openshiftClusterID=<infrastructure_ID>
          trunk: true
          userDataSecret:
            name: worker-user-data 15
          availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 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 15 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.

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.

14 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.

6.2.1.5. Sample YAML for a machine set custom resource on RHV

This sample YAML defines a machine set that runs on RHV and creates nodes that are labeled with **node-role.kubernetes.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 **<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> 5
  Selector: 6
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
        machine.openshift.io/cluster-api-machine-role: <role> 10
        machine.openshift.io/cluster-api-machine-type: <role> 11
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 13
      providerSpec:
        value:
          apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
          cluster_id: <ovirt_cluster_id> 14
          template_name: <ovirt_template_name> 15
          instance_type_id: <instance_type_id> 16
          cpu: 17
            sockets: <number_of_sockets> 18
            cores: <number_of_cores> 19
            threads: <number_of_threads> 20
          memory_mb: <memory_size> 21
          os_disk: 22
            size_gb: <disk_size> 23
          network_interfaces: 24
            vnic_profile_id:  <vnic_profile_id> 25
          credentialsSecret:
            name: ovirt-credentials 26
          kind: OvirtMachineProviderSpec
          type: <workload_type> 27
          userDataSecret:
            name: worker-user-data
```

1 7 9 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 3 10 11 13 Specify the node label to add.

4 8 12 Specify the infrastructure ID and node label. These two strings together cannot be longer than 35 characters.

5 Specify the number of machines to create.

6 Selector for the machines.

14 Specify the UUID for the RHV cluster to which this VM instance belongs.

15 Specify the RHV VM template to use to create the machine.

16 Optional: Specify the VM instance type. If you include this parameter, you do not need to specify the hardware parameters of the VM including CPU and memory because this parameter overrides all hardware parameters.

17 Optional: The CPU field contains the CPU's configuration, including sockets, cores, and threads.

18 Optional: Specify the number of sockets for a VM.

19 Optional: Specify the number of cores per socket.

20 Optional: Specify the number of threads per core.

21 Optional: Specify the size of a VM's memory in MiB.

22 Optional: Root disk of the node.

23 Optional: Specify the size of the bootable disk in GiB.

24 Optional: List of the network interfaces of the VM. If you include this parameter, OpenShift Container Platform discards all network interfaces from the template and creates new ones.

25 Optional: Specify the vNIC profile ID.

26 Specify the name of the secret that holds the RHV credentials.

27 Optional: Specify the workload type for which the instance is optimized. This value affects the **RHV VM** parameter. Supported values: **desktop**, **server**, **high_performance**.

NOTE

Because RHV uses a template when creating a VM, if you do not specify a value for an optional parameter, RHV uses the value for that parameter that is specified in the template.

6.2.1.6. Sample YAML for a machine set custom resource on vSphere

This sample YAML defines a 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_datacenter_name> ⑬
    datastore: <vcenter_datastore_name> ⑭
    folder: <vcenter_vm_folder_path> ⑮
    resourcepool: <vsphere_resource_pool> ⑯
    server: <vcenter_server_ip> ⑰
```

① ③ ⑤ Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. 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 **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.

⑪ Specify the vSphere VM network to deploy the machine set to.

⑫ Specify the vSphere VM template to use, such as **user-5ddjd-rhcos**.

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

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

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

⑯ Specify the vSphere resource pool for your VMs.

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

6.2.2. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

Example output

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

b. Check values of a specific machine set:

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

Example output

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

1 The cluster ID.

2 A default node label.

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

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

3. View the list of machine sets:

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

Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-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 machine set is available, the **DESIRED** and **CURRENT** values match. If the machine set is not available, wait a few minutes and run the command again.

6.2.3. Creating an infrastructure node

IMPORTANT

See Creating infrastructure machine sets for installer-provisioned infrastructure environments or for any cluster where the master 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 master 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 node selector so that pods without a node selector are assigned a subset of nodes to be deployed on, for example by default deployment in worker nodes. As an example, the **defaultNodeSelector** to deploy pods on worker nodes by default would look like:

```
defaultNodeSelector: node-role.kubernetes.io/app=
```

5. Move infrastructure resources to the newly labeled **infra** nodes.

6.2.4. Creating a machine config pool for infrastructure machines

If you need infrastructure machines to have dedicated configurations, you must create an infra pool.

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-ln-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                                  GENERATEDBYCONTROLLER
IGNITIONVERSION   CREATED
00-master                              365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0             31d
00-worker                              365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0             31d
01-master-container-runtime
```

```
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            31d
01-master-kubelet                                  365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0            31d
01-worker-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            31d
01-worker-kubelet                                  365c1cfd14de5b0e3b85e0fc815b0060f36ab955
2.2.0            31d
99-master-1ae2a1e0-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            31d
99-master-ssh                                                            2.2.0            31d
99-worker-1ae64748-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            31d
99-worker-ssh                                                            2.2.0            31d
rendered-infra-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            19s
rendered-master-072d4b2da7f88162636902b074e9e28e
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0            31d
rendered-master-3e88ec72aed3886dec061df60d16d1af
02c07496ba0417b3e12b78fb32baf6293d314f79   2.2.0            31d
rendered-master-419bee7de96134963a15fdf9dd473b25
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            17d
rendered-master-53f5c91c7661708adce18739cc0f40fb
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            13d
rendered-master-a6a357ec18e5bce7f5ac426fc7c5ffcd
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            7d3h
rendered-master-dc7f874ec77fc4b969674204332da037
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0            31d
rendered-worker-1a75960c52ad18ff5dfa6674eb7e533d
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0            31d
rendered-worker-2640531be11ba43c61d72e82dc634ce6
5b6fb8349a29735e48446d435962dec4547d3090   2.2.0            31d
rendered-worker-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            7d3h
rendered-worker-4f110718fe88e5f349987854a1147755
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            17d
rendered-worker-afc758e194d6188677eb837842d3b379
02c07496ba0417b3e12b78fb32baf6293d314f79   2.2.0            31d
rendered-worker-daa08cc1e8f5fcdeba24de60cd955cc3
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   2.2.0            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:
  labels:
    machineconfiguration.openshift.io/role: infra 1
  name: 51-infra
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:,infra
        filesystem: root
        mode: 0644
        path: /etc/infratest
```

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
DEGRADEDMACHINECOUNT   AGE
infra    rendered-infra-60e35c2e99f42d976e084fa94da4d0fc    True      False      False      1
1              1                   0                   4m20s
master   rendered-master-9360fdb895d4c131c7c4bebbae099c90   True      False      False
3          3                   3                   0                    91m
worker   rendered-worker-60e35c2e99f42d976e084fa94da4d0fc   True      False      False
2          2                   2                   0                    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.

6.3. 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.

6.3.1. 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-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
   Name:               ci-ln-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>:<effect>
   ```

For example:

```
$ oc adm taint nodes node1 node-role.kubernetes.io/infra:NoSchedule
```

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.

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
   operator: Exists 3
```

1 Specify the effect that you added to the node.

2 Specify the key that you added to the node.

3 Specify the **Exists** Operator to require a taint with the key **node-role.kubernetes.io/infra** to be present on 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.

6.4. 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.

6.4.1. Moving the router

You can deploy the router pod to a different machine set. By default, the pod is deployed to a worker node.

Prerequisites

- Configure additional machine sets in your OpenShift Container Platform cluster.

Procedure

1. 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
```

2. Edit the **ingresscontroller** resource and change the **nodeSelector** to use the **infra** label:

```
$ oc edit ingresscontroller default -n openshift-ingress-operator
```

Add the **nodeSelector** stanza that references the **infra** label to the **spec** section, as shown:

```
  spec:
    nodePlacement:
      nodeSelector:
        matchLabels:
          node-role.kubernetes.io/infra: ""
```

3. Confirm that the router pod is running on the **infra** node.

   a. 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   <none>           <none>
   router-default-955d875f4-255g8    0/1     Terminating   0          19h     10.129.2.4   ip-10-
   0-148-172.ec2.internal   <none>           <none>
   ```

   In this example, the running pod is on the **ip-10-0-217-226.ec2.internal** node.

   b. View the node status of the running pod:

   ```
   $ oc get node <node_name> 1
   ```

   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   v1.18.3
   ```

   Because the role list includes **infra**, the pod is running on the correct node.

6.4.2. Moving the default registry

You configure the registry Operator to deploy its pods to different nodes.

Prerequisites

- Configure additional machine sets in your OpenShift Container Platform cluster.

Procedure

1. 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
```

3. Add the following lines of text the **spec** section of the object:

```
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

4. Verify the registry pod has been moved to the infrastructure node.

   a. Run the following command to identify the node where the registry pod is located:

   ```
   $ oc get pods -o wide -n openshift-image-registry
   ```

   b. 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.

6.4.3. Moving the monitoring solution

By default, the Prometheus Cluster Monitoring stack, which contains Prometheus, Grafana, and AlertManager, is deployed to provide cluster monitoring. It is managed by the Cluster Monitoring Operator. To move its components to different machines, you create and apply a custom config map.

Procedure

1. Save the following **ConfigMap** definition as the **cluster-monitoring-configmap.yaml** file:

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: cluster-monitoring-config
     namespace: openshift-monitoring
   data:
     config.yaml: |+
       alertmanagerMain:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       prometheusK8s:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       prometheusOperator:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       grafana:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       k8sPrometheusAdapter:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       kubeStateMetrics:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       telemeterClient:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       openshiftStateMetrics:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
       thanosQuerier:
         nodeSelector:
           node-role.kubernetes.io/infra: ""
   ```

   Running this config map forces the components of the monitoring stack to redeploy to infrastructure nodes.

2. Apply the new config map:

   ```
   $ oc create -f cluster-monitoring-configmap.yaml
   ```

3. Watch the monitoring pods move to the new machines:

   ```
   $ watch 'oc get pod -n openshift-monitoring -o wide'
   ```

4. 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.

6.4.4. Moving the cluster logging resources

You can configure the Cluster Logging Operator to deploy the pods for any or all of the Cluster Logging components, Elasticsearch, Kibana, and Curator to different nodes. You cannot move the Cluster Logging Operator pod from its installed location.

For example, you can move the Elasticsearch pods to a separate node because of high CPU, memory, and disk requirements.

Prerequisites

- Cluster logging and Elasticsearch must be installed. These features are not installed by default.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit ClusterLogging instance
```

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: ''
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: 3
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

...
```

1 2 3 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.

Verification

To verify that a component has moved, you can use the **oc get pod -o wide** command.

For example:

- You want to move the Kibana pod from the **ip-10-0-147-79.us-east-2.compute.internal** node:

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

Example output

```
NAME                      READY   STATUS    RESTARTS   AGE   IP          NODE
NOMINATED NODE   READINESS GATES
kibana-5b8bdf44f9-ccpq9   2/2     Running   0          27s   10.129.2.18   ip-10-0-147-79.us-
east-2.compute.internal   <none>           <none>
```

- You want to move the Kibana Pod to the **ip-10-0-139-48.us-east-2.compute.internal** node, a dedicated infrastructure node:

```
$ oc get nodes
```

Example output

```
NAME                                         STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.18.3
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.18.3
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.18.3
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.18.3
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.18.3
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.18.3
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.18.3
```

Note that the node has a **node-role.kubernetes.io/infra: ''** label:

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

Example output

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: ''
...
```

- To move the Kibana pod, edit the **ClusterLogging** CR to add a node selector:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:

...

  visualization:
    kibana:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 Add a node selector to match the label in the node specification.

- After you save the CR, the current Kibana pod is terminated and new pod is deployed:

```
$ oc get pods
```

Example output

```
NAME                                            READY   STATUS        RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running       0          29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running       0          28m
fluentd-42dzz                                   1/1     Running       0          28m
fluentd-d74rq                                   1/1     Running       0          28m
fluentd-m5vr9                                   1/1     Running       0          28m
fluentd-nkxl7                                   1/1     Running       0          28m
fluentd-pdvqb                                   1/1     Running       0          28m
fluentd-tflh6                                   1/1     Running       0          28m
kibana-5b8bdf44f9-ccpq9                         2/2     Terminating   0          4m11s
kibana-7d85dcffc8-bfpfp                         2/2     Running       0          33s
```

- The new pod is on the **ip-10-0-139-48.us-east-2.compute.internal** node:

  ```
  $ oc get pod kibana-7d85dcffc8-bfpfp -o wide
  ```

  Example output

  ```
  NAME                    READY   STATUS      RESTARTS   AGE   IP            NODE
  NOMINATED NODE   READINESS GATES
  kibana-7d85dcffc8-bfpfp   2/2     Running      0         43s   10.131.0.22   ip-10-0-139-48.us-
  east-2.compute.internal   <none>           <none>
  ```

- After a few moments, the original Kibana pod is removed.

  ```
  $ oc get pods
  ```

  Example output

  ```
  NAME                                            READY   STATUS    RESTARTS   AGE
  cluster-logging-operator-84d98649c4-zb9g7       1/1     Running   0          30m
  elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
  elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
  elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running   0          29m
  fluentd-42dzz                                   1/1     Running   0          29m
  fluentd-d74rq                                   1/1     Running   0          29m
  fluentd-m5vr9                                   1/1     Running   0          29m
  fluentd-nkxl7                                   1/1     Running   0          29m
  fluentd-pdvqb                                   1/1     Running   0          29m
  fluentd-tflh6                                   1/1     Running   0          29m
  kibana-7d85dcffc8-bfpfp                         2/2     Running   0          62s
  ```

Additional resources

- See the monitoring documentation for the general instructions on moving OpenShift Container Platform components.

CHAPTER 7. ADDING RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

In OpenShift Container Platform, you can add Red Hat Enterprise Linux (RHEL) compute, or worker, machines to a user-provisioned infrastructure cluster or a installation-provisioned infrastructure cluster. You can use RHEL as the operating system on only compute machines.

7.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.5, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines, which are also known as worker machines, in your cluster if you use a user-provisioned infrastructure installation. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane, or master, machines in your cluster.

As with all installations that use user-provisioned infrastructure, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks.

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.

IMPORTANT

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.

You must add any RHEL compute machines to the cluster after you initialize the control plane.

7.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

The Red Hat Enterprise Linux (RHEL) compute machine hosts, which are also known as worker 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 percent 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 7.7-7.8 with "Minimal" installation option.

IMPORTANT

Only RHEL 7.7-7.8 is supported in OpenShift Container Platform 4.5. You must not upgrade your compute machines to RHEL 8.

  - If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See Enabling FIPS Mode in the RHEL 7 documentation.
  - 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 the system's temporary directory. The system's temporary directory is determined according to the rules defined in the tempfile module in Python's 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 host names. Any network security access control that is in place must allow the system access to the cluster's API service endpoints.

7.2.1. 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.

7.3. 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.

7.3.1. 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 7.9 Amazon Machine Images (AMI):

  ```
  $ aws ec2 describe-images --owners 309956199498 \ ❶
  --query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \ ❷
  --filters "Name=name,Values=RHEL-7.9*" \ ❸
  --region us-east-1 \ ❹
  --output table ❺
  ```

  ❶ 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.

  ❷ 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.

  ❸ The **--filter** command option sets which version of RHEL is shown. In this example, since the filter is set by **"Name=name,Values=RHEL-7.9*"**, then RHEL 7.9 AMIs are shown.

  ❹ The **--region** command option sets where the region where an AMI is stored.

  ❺ 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 7.9.

Example output

```
-------------------------------------------------------------------------------------------------------
|                                          DescribeImages                                             |
+---------------------------+-----------------------------------------------------+-----------------------+
|  2020-05-13T09:50:36.000Z |  RHEL-7.9_HVM_BETA-20200422-x86_64-0-Hourly2-GP2  |  ami-
038714142142a6a64 |
|  2020-09-18T07:51:03.000Z |  RHEL-7.9_HVM_GA-20200917-x86_64-0-Hourly2-GP2    |  ami-
005b7876121b7244d |
|  2021-02-09T09:46:19.000Z |  RHEL-7.9_HVM-20210208-x86_64-0-Hourly2-GP2       |  ami-
030e754805234517e |
+---------------------------+-----------------------------------------------------+-----------------------+
```

Additional resources

- You may also manually import RHEL images to AWS .

7.4. PREPARING THE MACHINE TO RUN THE PLAYBOOK

Before you can add compute machines that use Red Hat Enterprise Linux as the operating system to an OpenShift Container Platform 4.5 cluster, you must prepare a machine to run the playbook from. 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 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 4.5:

```
# subscription-manager repos \
```

```
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms" \
    --enable="rhel-7-server-ose-4.5-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.

7.5. 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 4.5:

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.5-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.

7.6. 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. The following permissions are required:

    - **sts:AssumeRole**

    - **ec2:DescribeInstances**

    - **ec2:DescribeRegions**

7.7. 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.

7.8. 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 4.5 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-rhel7-0.example.com
mycluster-rhel7-1.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 host name 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.

7.9. 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  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

The output lists all of the machines that you created.

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. Once 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.

- 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
  ```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests.

7.10. 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.

Paramter	Description	Values
ansible_user	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 **root**.
ansible_become	If the values of **ansible_user** is not root, you must set **ansible_become** to **True**, and the user that you specify as the **ansible_user** must be configured for passwordless sudo access.	**True**. If the value is not **True**, do not specify and define this parameter.
openshift_kubeconfig_path	Specifies a path and file name to a local directory that contains the **kubeconfig** file for your cluster.	The path and name of the configuration file.

7.10.1. 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-local-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 8. 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.

8.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.5, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines, which are also known as worker machines, in your cluster if you use a user-provisioned infrastructure installation. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane, or master, machines in your cluster.

As with all installations that use user-provisioned infrastructure, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks.

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.

IMPORTANT

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.

You must add any RHEL compute machines to the cluster after you initialize the control plane.

8.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

The Red Hat Enterprise Linux (RHEL) compute machine hosts, which are also known as worker 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 percent 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 7.7-7.8 with "Minimal" installation option.

IMPORTANT

Only RHEL 7.7-7.8 is supported in OpenShift Container Platform 4.5. You must not upgrade your compute machines to RHEL 8.

- If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See Enabling FIPS Mode in the RHEL 7 documentation.
- 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 the system’s temporary directory. The system’s temporary directory is determined according to the rules defined in the tempfile module in Python’s 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 host names. Any network security access control that is in place must allow the system access to the cluster’s API service endpoints.

8.2.1. 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.

8.3. 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.

8.3.1. 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 7.9 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-7.9*" \ 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-7.9*"**, then RHEL 7.9 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 7.9.

Example output

```
--------------------------------------------------------------------------------------------------------------
|                                    DescribeImages                                          |
+---------------------------+----------------------------------------------------+-------------------------+
|  2020-05-13T09:50:36.000Z |  RHEL-7.9_HVM_BETA-20200422-x86_64-0-Hourly2-GP2  |  ami-
038714142142a6a64 |
|  2020-09-18T07:51:03.000Z |  RHEL-7.9_HVM_GA-20200917-x86_64-0-Hourly2-GP2    |  ami-
005b7876121b7244d |
|  2021-02-09T09:46:19.000Z |  RHEL-7.9_HVM-20210208-x86_64-0-Hourly2-GP2       |  ami-
030e754805234517e |
+---------------------------+----------------------------------------------------+-------------------------+
```

Additional resources

- You may also manually import RHEL images to AWS .

8.4. 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 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 4.5:

   ```
   # subscription-manager repos \
       --enable="rhel-7-server-rpms" \
       --enable="rhel-7-server-extras-rpms" \
       --enable="rhel-7-server-ose-4.5-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.

8.5. 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. The following permissions are required:

   - **sts:AssumeRole**
   - **ec2:DescribeInstances**
   - **ec2:DescribeRegions**

8.6. 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.

8.7. 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 4.5 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-rhel7-0.example.com
mycluster-rhel7-1.example.com

[new_workers]
mycluster-rhel7-2.example.com
mycluster-rhel7-3.example.com
```

   In this example, the **mycluster-rhel7-0.example.com** and **mycluster-rhel7-1.example.com** machines are in the cluster and you add the **mycluster-rhel7-2.example.com** and **mycluster-rhel7-3.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.

8.8. 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  v1.18.3
   master-1  Ready     master  63m  v1.18.3
   master-2  Ready     master  64m  v1.18.3
   worker-0  NotReady  worker  76s  v1.18.3
   worker-1  NotReady  worker  70s  v1.18.3
   ```

   The output lists all of the machines that you created.

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. Once 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.

- 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
  ```

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  v1.20.0
master-1  Ready    master  73m  v1.20.0
master-2  Ready    master  74m  v1.20.0
worker-0  Ready    worker  11m  v1.20.0
worker-1  Ready    worker  11m  v1.20.0
```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests.

8.9. 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.

Paramter	Description	Values
ansible_user	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 **root**.
ansible_become	If the values of **ansible_user** is not root, you must set **ansible_become** to **True**, and the user that you specify as the **ansible_user** must be configured for passwordless sudo access.	**True**. If the value is not**True**, do not specify and define this parameter.
openshift_kubeconfig_path	Specifies a path and file name to a local directory that contains the **kubeconfig** file for your cluster.	The path and name of the configuration file.

CHAPTER 9. USER-PROVISIONED INFRASTRUCTURE

9.1. 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.

9.1.1. 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.

9.1.2. 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.

9.1.3. 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  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

The output lists all of the machines that you created.

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. Once 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.

- 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
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests.

9.2. ADDING COMPUTE MACHINES TO VSPHERE

You can add more compute machines to your OpenShift Container Platform cluster on VMware vSphere.

9.2.1. 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+.

9.2.2. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

You can create more compute machines for your cluster that uses user-provisioned infrastructure on VMware vSphere.

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. After the template deploys, deploy a VM for a machine in the cluster.
   a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
   b. 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**.
   c. On the **Select a name and folder**tab, select the name of the folder that you created for the cluster.
   d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
   e. Optional: On the **Select storage** tab, customize the storage options.
   f. On the **Select clone options**, select **Customize this virtual machine's hardware**.
   g. On the **Customize hardware** tab, click **VM Options → Advanced**.
      - From the **Latency Sensitivity** list, select **High**.
      - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
        - **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**.
   h. 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. Also, make sure to select the correct network under **Add network adapter** if there are multiple networks available.

i. Complete the configuration and power on the VM.

2. Continue to create more compute machines for your cluster.

9.2.3. 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  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

The output lists all of the machines that you created.

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. Once 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.

- 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
```

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  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests.

9.3. ADDING COMPUTE MACHINES TO BARE METAL

You can add more compute machines to your OpenShift Container Platform cluster on bare metal.

9.3.1. 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.

9.3.2. 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.

9.3.2.1. Creating more 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.

- Obtain the URL of the BIOS or UEFI RHCOS image file that you uploaded to your HTTP server during cluster installation.

Procedure

1. 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.
2. After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
3. Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda ①
coreos.inst.image_url=<bare_metal_image_URL> ②
coreos.inst.ignition_url=http://example.com/worker.ign ③
```

① Specify the block device of the system to install to.

② Specify the URL of the UEFI or BIOS image that you uploaded to your server.

③ Specify the URL of the compute Ignition config file.

4. Press **Enter** to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
5. Continue to create more compute machines for your cluster.

9.3.2.2. Creating more 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>-installer-kernel-<architecture> ①
  APPEND ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-metal.<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/worker.ign ② ③
```

① Specify the location of the **kernel** file that you uploaded to your HTTP server.

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 locations of the RHCOS files that you uploaded to your HTTP server. The **initrd** parameter value is the location of the **initramfs** file, the **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image, and the **coreos.inst.ignition_url** parameter value is the location of the worker Ignition config file.

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:

```
kernel http://<HTTP_server>/rhcos-<version>-installer-kernel-<architecture> ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-metal.<arhcitectutre>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/worker.ign ① ②
initrd http://<HTTP_server>/rhcos-<version>-installer-initramfs.<architecture>.img ③
boot
```

① Specify 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** parameter value is the location of the **initramfs** file, the **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image, 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?.

2. Use the PXE or iPXE infrastructure to create the required compute machines for your cluster.

9.3.3. 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  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

The output lists all of the machines that you created.

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. Once 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.

- 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
```

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  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```

   NOTE

   It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests.

CHAPTER 10. DEPLOYING MACHINE HEALTH CHECKS

You can configure and deploy a machine health check to automatically repair damaged machines in a machine pool.

IMPORTANT

This process is not applicable to clusters where you manually provisioned the machines yourself. You can use the advanced machine management and scaling capabilities only in clusters where the machine API is operational.

10.1. ABOUT MACHINE HEALTH CHECKS

You can define conditions under which machines in a cluster are considered unhealthy by using a **MachineHealthCheck** resource. Machines matching the conditions are automatically remediated.

To monitor machine health, create a **MachineHealthCheck** custom resource (CR) that includes a label for the set of machines to monitor and a condition to check, such as staying in the **NotReady** status for 15 minutes or displaying a permanent condition in the node-problem-detector.

The controller that observes a **MachineHealthCheck** CR checks for the condition that you defined. If a machine fails the health check, the machine is automatically deleted and a new one is created to take its place. When a machine is deleted, you see a **machine deleted** event.

NOTE

For machines with the master role, the machine health check reports the number of unhealthy nodes, but the machine is not deleted. For example:

Example output

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

```
NAME      MAXUNHEALTHY   EXPECTEDMACHINES   CURRENTHEALTHY
example   40%        3          1
```

To limit the disruptive impact of machine deletions, 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, the controller stops deleting machines and you must manually intervene.

To stop the check, remove the custom resource.

10.1.1. 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. To change the default remediation process from machine deletion to host power-cycle, annotate the MachineHealthCheck resource with the **machine.openshift.io/remediation-strategy: external-baremetal** annotation.

After you set the annotation, unhealthy machines are power-cycled by using BMC credentials.

10.1.2. 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.
- Control plane machines are not currently supported and are not remediated if they are unhealthy.
- 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

- For more information about the node conditions you can define in a **MachineHealthCheck** CR, see About listing all the nodes in a cluster .
- For more information about short-circuiting, see Short-circuiting machine health check remediation.

10.2. SAMPLE MACHINEHEALTHCHECK RESOURCE

The **MachineHealthCheck** resource resembles one of the following YAML files:

MachineHealthCheck for bare metal

```
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 machine set to track in **<cluster_name>-<label>-<zone>** format. For example, **prod-node-us-east-1a**.

6 7 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 a workload on an unhealthy machine.

8 Specify the amount of unhealthy machines allowed in the targeted pool. This can be set as a percentage or an integer.

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.

MachineHealthCheck for all other installation types

```
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 6 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 a workload on an unhealthy machine.

7 Specify the amount of unhealthy machines allowed in the targeted pool. This can be set as a percentage or an integer.

8 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.

10.2.1. 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 **maxUnhealthy** field can be set as either an integer or percentage. There are different remediation implementations depending on the **maxUnhealthy** value.

10.2.1.1. 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.

10.2.1.2. 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.

10.3. CREATING A MACHINEHEALTHCHECK RESOURCE

Additional resources

You can create a **MachineHealthCheck** resource for all **MachineSets** in your cluster. You should not create a **MachineHealthCheck** resource that targets control plane machines.

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
```