OpenShift Container Platform 4.10

Machine management

Adding and maintaining cluster machines

OpenShift Container Platform 4.10 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. OVERVIEW OF MACHINE MANAGEMENT

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

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

Machine management is implemented as a Custom Resource Definition(CRD). A CRD object defines a new unique object **Kind** in the cluster and enables the Kubernetes API server to handle the object's entire lifecycle.

The Machine API Operator provisions the following resources:

- MachineSet

- Machine

- Cluster Autoscaler

- Machine Autoscaler

- Machine Health Checks

What you can do with machine sets

As a cluster administrator you can:

- Create a machine set on:

  - AWS

  - Azure

  - GCP

  - OpenStack

  - RHV

  - vSphere

- Manually scale a machine set by adding or removing a machine from the machine set.

- Modify a machine set through the MachineSet YAML configuration file.

- Delete a machine.

- Create infrastructure machine sets.

- Configure and deploy a machine health check to automatically fix damaged machines in a machine pool.

Autoscaler

Autoscale your cluster to ensure flexibility to changing workloads. To autoscale your OpenShift Container Platform cluster, you must first deploy a cluster autoscaler, and then deploy a machine

autoscaler for each machine set. The cluster autoscaler increases and decreases the size of the cluster based on deployment needs. The machine autoscaler adjusts the number of machines in the machine sets that you deploy in your OpenShift Container Platform cluster.

User-provisioned infrastructure

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

What you can do with RHEL compute machines

As a cluster administrator, you can:

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

- Add more Red Hat Enterprise Linux (RHEL) compute machines to an existing cluster.

CHAPTER 2. CREATING MACHINE SETS

2.1. CREATING A MACHINE SET ON ALIBABA CLOUD

You can create a different machine set to serve a specific purpose in your OpenShift Container Platform cluster on Alibaba Cloud. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.
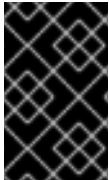
> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

2.1.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.1.2. Sample YAML for a machine set custom resource on Alibaba Cloud

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

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

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

```
    metadata:
      labels:
        node-role.kubernetes.io/<role>: ""
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1
        credentialsSecret:
          name: alibabacloud-credentials
        imageId: <image_id> 11
        instanceType: <instance_type> 12
        kind: AlibabaCloudMachineProviderConfig
        ramRoleName: <infrastructure_id>-role-worker 13
        regionId: <region> 14
        resourceGroup: 15
          id: <resource_group_id>
          type: ID
        securityGroups:
        - tags: 16
          - Key: Name
            Value: <infrastructure_id>-sg-<role>
          type: Tags
        systemDisk: 17
          category: cloud_essd
          size: <disk_size>
        tag: 18
        - Key: kubernetes.io/cluster/<infrastructure_id>
          Value: owned
        userDataSecret:
          name: <user_data_secret> 19
        vSwitch:
          tags: 20
          - Key: Name
            Value: <infrastructure_id>-vswitch-<zone>
          type: Tags
        vpcId: ""
        zoneId: <zone> 21
```

[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 (**oc**) 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] Specify the node label to add.

[4] [6] [10] Specify the infrastructure ID, node label, and zone.

[11] Specify the image to use. Use an image from an existing default machine set for the cluster.

[12] Specify the instance type you want to use for the machine set.

[13] Specify the name of the RAM role to use for the machine set. Use the value that the installer populates in the default machine set.

14 Specify the region to place machines on.

15 Specify the resource group and type for the cluster. You can use the value that the installer populates in the default machine set, or specify a different one.

16 18 20 Specify the tags to use for the machine set. Minimally, you must include the tags shown in this example, with appropriate values for your cluster. You can include additional tags, including the tags that the installer populates in the default machine set it creates, as needed.

17 Specify the type and size of the root disk. Use the **category** value that the installer populates in the default machine set it creates. If required, specify a different value in gigabytes for **size**.

19 Specify the name of the secret in the user data YAML file that is in the **openshift-machine-api** namespace. Use the value that the installer populates in the default machine set.

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

2.1.2.1. Machine set parameters for Alibaba Cloud usage statistics

The default machine sets that the installer creates for Alibaba Cloud clusters include nonessential tag values that Alibaba Cloud uses internally to track usage statistics. These tags are populated in the **securityGroups**, **tag**, and **vSwitch** parameters of the **spec.template.spec.providerSpec.value** list.

When creating machine sets to deploy additional machines, you must include the required Kubernetes tags. The usage statistics tags are applied by default, even if they are not specified in the machine sets you create. You can also include additional tags as needed.

The following YAML snippets indicate which tags in the default machine sets are optional and which are required.

Tags in spec.template.spec.providerSpec.value.securityGroups

```
spec:
  template:
    spec:
      providerSpec:
        value:
          securityGroups:
          - tags:
            - Key: kubernetes.io/cluster/<infrastructure_id> 1
              Value: owned
            - Key: GISV
              Value: ocp
            - Key: sigs.k8s.io/cloud-provider-alibaba/origin 2
              Value: ocp
            - Key: Name
              Value: <infrastructure_id>-sg-<role> 3
            type: Tags
```

1 2 Optional: This tag is applied even when not specified in the machine set.

3 Required.

where:

- **<infrastructure_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.

- **<role>** is the node label to add.

Tags in spec.template.spec.providerSpec.value.tag

```
spec:
  template:
    spec:
      providerSpec:
        value:
          tag:
          - Key: kubernetes.io/cluster/<infrastructure_id> 1
            Value: owned
          - Key: GISV 2
            Value: ocp
          - Key: sigs.k8s.io/cloud-provider-alibaba/origin 3
            Value: ocp
```

2 **3** Optional: This tag is applied even when not specified in the machine set.

1 Required.

where **<infrastructure_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.

Tags in spec.template.spec.providerSpec.value.vSwitch

```
spec:
  template:
    spec:
      providerSpec:
        value:
          vSwitch:
            tags:
            - Key: kubernetes.io/cluster/<infrastructure_id> 1
              Value: owned
            - Key: GISV 2
              Value: ocp
            - Key: sigs.k8s.io/cloud-provider-alibaba/origin 3
              Value: ocp
            - Key: Name
              Value: <infrastructure_id>-vswitch-<zone> 4
            type: Tags
```

1 **2** **3** Optional: This tag is applied even when not specified in the machine set.

4 Required.

where:

- **<infrastructure_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.

- **<zone>** is the zone within your region to place machines on.

2.1.3. 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 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 ❷
machine.openshift.io/cluster-api-machine-type: worker
machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

❶ The cluster ID.

❷ 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.

2.2. 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.

IMPORTANT

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

2.2.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.2.2. 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, **<infrastructure_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>  1
  name: <infrastructure_id>-<role>-<zone>  2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>  3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone>  4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>  5
        machine.openshift.io/cluster-api-machine-role: <role>  6
        machine.openshift.io/cluster-api-machine-type: <role>  7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone>  8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: ""  9
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9  10
          apiVersion: awsproviderconfig.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
          credentialsSecret:
            name: aws-cloud-credentials
          deviceIndex: 0
          iamInstanceProfile:
            id: <infrastructure_id>-worker-profile  11
          instanceType: m6i.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: us-east-1a
            region: us-east-1
          securityGroups:
            - filters:
                - name: tag:Name
                  values:
```

```
            - <infrastructure_id>-worker-sg 12
        subnet:
          filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-private-us-east-1a 13
        tags:
          - name: kubernetes.io/cluster/<infrastructure_id> 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 get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

[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.

2.2.3. 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 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 ❶
      machine.openshift.io/cluster-api-machine-role: worker ❷
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: agl030519-vplxk-worker-us-east-1a
```

❶ The cluster ID.

❷ 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.

2.2.4. 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 utilize unused AWS EC2 capacity and are less expensive than On-Demand Instances. You can use Spot Instances for workloads that can tolerate interruptions, such as batch or stateless, horizontally scalable workloads.

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

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

- The instance price exceeds your maximum price

- The demand for Spot Instances increases

- The supply of Spot Instances decreases

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

2.2.5. 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: {}
  ```

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

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

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

2.2.6. Machine sets that deploy machines as Dedicated Instances

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

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

2.2.7. Creating Dedicated Instances by using machine sets

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

Procedure

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

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

2.3. 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.
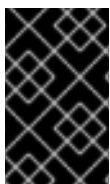
> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

2.3.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.3.2. 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 a region and creates nodes that are labeled with **node-role.kubernetes.io/<role>: ""**.

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
 labels:
   machine.openshift.io/cluster-api-cluster: <infrastructure_id>     1
   machine.openshift.io/cluster-api-machine-role: <role>     2
   machine.openshift.io/cluster-api-machine-type: <role>     3
 name: <infrastructure_id>-<role>-<region>     4
```

```yaml
  namespace: openshift-machine-api
spec:
 replicas: 1
 selector:
   matchLabels:
     machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
     machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
 template:
   metadata:
     creationTimestamp: null
     labels:
       machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
       machine.openshift.io/cluster-api-machine-role: <role> 8
       machine.openshift.io/cluster-api-machine-type: <role> 9
       machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
   spec:
     metadata:
       creationTimestamp: null
       labels:
         machine.openshift.io/cluster-api-machineset: <machineset_name> 11
         node-role.kubernetes.io/<role>: "" 12
     providerSpec:
       value:
         apiVersion: azureproviderconfig.openshift.io/v1beta1
         credentialsSecret:
           name: azure-cloud-credentials
           namespace: openshift-machine-api
         image:
           offer: ""
           publisher: ""
           resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 13
           sku: ""
           version: ""
         internalLoadBalancer: ""
         kind: AzureMachineProviderSpec
         location: <region> 14
         managedIdentity: <infrastructure_id>-identity 15
         metadata:
           creationTimestamp: null
         natRule: null
         networkResourceGroup: ""
         osDisk:
           diskSizeGB: 128
           managedDisk:
             storageAccountType: Premium_LRS
           osType: Linux
         publicIP: false
         publicLoadBalancer: ""
         resourceGroup: <infrastructure_id>-rg 16
         sshPrivateKey: ""
         sshPublicKey: ""
         subnet: <infrastructure_id>-<role>-subnet 17 18
         userDataSecret:
```

```
        name: worker-user-data 19
    vmSize: Standard_D4s_v3
    vnet: <infrastructure_id>-vnet 20
    zone: "1" 21
```

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

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

You can obtain the subnet by running the following command:

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

You can obtain the vnet by running the following command:

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

2 **3** **8** **9** **12** **18** **19** Specify the node label to add.

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

11 Optional: Specify the machine set name to enable the use of availability sets. This setting only applies to new compute machines.

14 Specify the region to place machines on.

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

2.3.3. 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 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.

2.3.4. Machine sets that deploy machines as Spot VMs

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

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

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

- The instance price exceeds your maximum price

- The supply of Spot VMs decreases

- Azure needs capacity back

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

2.3.5. Creating Spot VMs by using machine sets

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

Procedure

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

    ```
    providerSpec:
      value:
        spotVMOptions: {}
    ```

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

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

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

2.3.6. Machine sets that deploy machines on Ephemeral OS disks

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

Additional resources

- For more information, see the Microsoft Azure documentation about Ephemeral OS disks for Azure VMs.

2.3.7. Creating machines on Ephemeral OS disks by using machine sets

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

Prerequisites

- Have an existing Microsoft Azure cluster.

Procedure

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

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

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

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

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

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

   4 Ephemeral OS disks are only supported for VMs or scale set instances that use the Standard LRS storage account type.
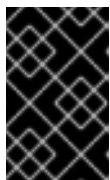
   > **IMPORTANT**
   >
   > The implementation of Ephemeral OS disk support in OpenShift Container Platform only supports the **CacheDisk** placement type. Do not change the **placement** configuration setting.

3. Create a machine set using the updated configuration:

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

Verification

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

2.3.8. Enabling customer-managed encryption keys for a machine set

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

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

Prerequisites

- [Create an Azure Key Vault instance](#) .

- [Create an instance of a disk encryption set](#) .

- [Grant the disk encryption set access to key vault](#) .

Procedure

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

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

Additional resources

- You can learn more about [customer-managed keys](#) in the Azure documentation.

2.3.9. Accelerated Networking for Microsoft Azure VMs

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

2.3.9.1. Limitations

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

- Accelerated Networking is only supported on clusters where the Machine API is operational.

- Although the minimum requirement for an Azure worker node is two vCPUs, Accelerated Networking requires an Azure VM size that includes at least four vCPUs. To satisfy this requirement, you can change the value of **vmSize** in your machine set. For information about Azure VM sizes, see Microsoft Azure documentation.

- When this feature is enabled on an existing Azure cluster, only newly provisioned nodes are affected. Currently running nodes are not reconciled. To enable the feature on all nodes, you must replace each existing machine. This can be done for each machine individually, or by scaling the replicas down to zero, and then scaling back up to your desired number of replicas.

Additional resources

- For information about enabling Accelerated Networking during installation, see Enabling Accelerated Networking during installation.

2.3.10. Enabling Accelerated Networking on an existing Microsoft Azure cluster

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

Prerequisites

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

Procedure

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

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

   The machine sets are listed in the form of **<cluster-id>-worker-<region>**.

   Example output

   ```
   NAME                          DESIRED  CURRENT  READY  AVAILABLE  AGE
   jmywbfb-8zqpx-worker-centralus1   1        1        1      1          15m
   jmywbfb-8zqpx-worker-centralus2   1        1        1      1          15m
   jmywbfb-8zqpx-worker-centralus3   1        1        1      1          15m
   ```

2. For each machine set:

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

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

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

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

1 This line enables Accelerated Networking.

2 Specify an Azure VM size that includes at least four vCPUs. For information about VM sizes, see Microsoft Azure documentation.

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

Verification

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

Additional resources

- For more details about scaling a machine set, see Manually scaling a machine set .

2.4. CREATING A MACHINE SET ON AZURE STACK HUB

You can create a different machine set to serve a specific purpose in your OpenShift Container Platform cluster on Microsoft Azure Stack Hub. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.
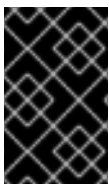
> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

2.4.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.4.2. Sample YAML for a machine set custom resource on Azure Stack Hub

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" 11
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          availabilitySet: <availability_set> 12
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 13
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 14
          managedIdentity: <infrastructure_id>-identity 15
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
```

```
      diskSizeGB: 128
      managedDisk:
        storageAccountType: Premium_LRS
      osType: Linux
    publicIP: false
    publicLoadBalancer: ""
    resourceGroup: <infrastructure_id>-rg
    sshPrivateKey: ""
    sshPublicKey: ""
    subnet: <infrastructure_id>-<role>-subnet
    userDataSecret:
      name: worker-user-data
    vmSize: Standard_DS4_v2
    vnet: <infrastructure_id>-vnet
    zone: "1"
```

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

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

You can obtain the subnet by running the following command:

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

You can obtain the vnet by running the following command:

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

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

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

14 Specify the region to place machines on.

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

12 Specify the availability set for the cluster.

2.4.3. 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.

- Create an availability set in which to deploy Azure Stack Hub machines.

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 **<availabilitySet>**, **<clusterID>**, and **<role>** parameter values.

   a. If you are not sure 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.

2.4.4. Enabling customer-managed encryption keys for a machine set

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

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

Prerequisites

- Create an Azure Key Vault instance .

- Create an instance of a disk encryption set .

- Grant the disk encryption set access to key vault .

Procedure

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

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

Additional resources

- You can learn more about customer-managed keys in the Azure documentation.

2.5. 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.
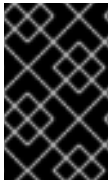
> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

2.5.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.5.2. 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, **<infrastructure_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<role>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>  1
  name: <infrastructure_id>-w-a  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>-w-a  4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>  5
        machine.openshift.io/cluster-api-machine-role: <role>  6
        machine.openshift.io/cluster-api-machine-type: <role>  7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-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
        gcpMetadata: 11
        - key: <custom_metadata_key>
          value: <custom_metadata_value>
        kind: GCPMachineProviderSpec
        machineType: n1-standard-4
        metadata:
          creationTimestamp: null
        networkInterfaces:
        - network: <infrastructure_id>-network 12
          subnetwork: <infrastructure_id>-worker-subnet 13
        projectID: <project_name> 14
        region: us-central1
        serviceAccounts:
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com 15 16
          scopes:
          - https://www.googleapis.com/auth/cloud-platform
        tags:
        - <infrastructure_id>-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 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/<infrastructure_id>-worker-a
```

11

Optional: Specify custom metadata in the form of a **key:value** pair. For example use cases, see the GCP documentation for [setting custom metadata](#).

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

2.5.3. 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 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.

2.5.4. Machine sets that deploy machines as preemptible VM instances

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

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

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

- There is a system or maintenance event

- The supply of preemptible VM instances decreases

- The instance reaches the end of the allotted 24-hour period for preemptible VM instances

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

2.5.5. Creating preemptible VM instances by using machine sets

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

Procedure

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

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

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

2.5.6. Enabling customer-managed encryption keys for a machine set

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

You can enable encryption with a customer-managed key by using the Machine API. You must first create a KMS key and assign the correct permissions to a service account. The KMS key name, key ring name, and location are required to allow a service account to use your key.

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

Procedure

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

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

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

```
providerSpec:
  value:
    # ...
    disks:
    - type:
      # ...
      encryptionKey:
        kmsKey:
          name: machine-encryption-key 1
          keyRing: openshift-encrpytion-ring 2
          location: global 3
          projectID: openshift-gcp-project 4
        kmsKeyServiceAccount: openshift-service-account@openshift-gcp-
project.iam.gserviceaccount.com 5
```

1 The name of the customer-managed encryption key that is used for the disk encryption.

2 The name of the KMS key ring that the KMS key belongs to.

3 The GCP location in which the KMS key ring exists.

4 Optional: The ID of the project in which the KMS key ring exists. If a project ID is not set, the machine set **projectID** in which the machine set was created is used.

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

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

2.5.7. Enabling GPU support for a machine set

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

Table 2.1. Supported GPU configurations

Model name	GPU type	Machine types[1]
NVIDIA A100	**nvidia-tesla-a100**	**a2-highgpu-1g** **a2-highgpu-2g** **a2-highgpu-4g** **a2-highgpu-8g** **a2-megagpu-16g**

Model name	GPU type	Machine types[1]
NVIDIA K80	**nvidia-tesla-k80**	• **n1-standard-1** • **n1-standard-2** • **n1-standard-4** • **n1-standard-8** • **n1-standard-16** • **n1-standard-32** • **n1-standard-64** • **n1-standard-96** • **n1-highmem-2** • **n1-highmem-4** • **n1-highmem-8** • **n1-highmem-16** • **n1-highmem-32** • **n1-highmem-64** • **n1-highmem-96** • **n1-highcpu-2** • **n1-highcpu-4** • **n1-highcpu-8** • **n1-highcpu-16** • **n1-highcpu-32** • **n1-highcpu-64** • **n1-highcpu-96**
NVIDIA P100	**nvidia-tesla-p100**	
NVIDIA P4	**nvidia-tesla-p4**	
NVIDIA T4	**nvidia-tesla-t4**	
NVIDIA V100	**nvidia-tesla-v100**	

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

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

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

> **NOTE**
>
> GPUs for graphics workloads are not supported.

Procedure

1. In a text editor, open the YAML file for an existing machine set or create a new one.

2. Specify a GPU configuration under the **providerSpec** field in your machine set YAML file. See the following examples of valid configurations:

   Example configuration for the A2 machine series:

   ```
   providerSpec:
    value:
      machineType: a2-highgpu-1g ❶
      onHostMaintenance: Terminate ❷
      restartPolicy: Always ❸
   ```

   ❶ Specify the machine type. Ensure that the machine type is included in the A2 machine series.

   ❷ When using GPU support, you must set **onHostMaintenance** to **Terminate**.

   ❸ Specify the restart policy for machines deployed by the machine set. Allowed values are **Always** or **Never**.

   Example configuration for the N1 machine series:

   ```
   providerSpec:
    value:
      gpus:
      - count: 1 ❶
        type: nvidia-tesla-p100 ❷
      machineType: n1-standard-1 ❸
      onHostMaintenance: Terminate ❹
      restartPolicy: Always ❺
   ```

   ❶ Specify the number of GPUs to attach to the machine.

   ❷ Specify the type of GPUs to attach to the machine. Ensure that the machine type and GPU type are compatible.

   ❸ Specify the machine type. Ensure that the machine type and GPU type are compatible.

   ❹ When using GPU support, you must set **onHostMaintenance** to **Terminate**.

   ❺ Specify the restart policy for machines deployed by the machine set. Allowed values are **Always** or **Never**.

2.6. CREATING A MACHINE SET ON IBM CLOUD

You can create a different machine set to serve a specific purpose in your OpenShift Container Platform cluster on IBM Cloud. For example, you might create infrastructure machine sets and related machines so that you can move supporting workloads to the new machines.
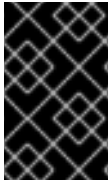
> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

2.6.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.6.2. Sample YAML for a machine set custom resource on IBM Cloud

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>  1
    machine.openshift.io/cluster-api-machine-role: <role>  2
    machine.openshift.io/cluster-api-machine-type: <role>  3
  name: <infrastructure_id>-<role>-<region>  4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>  5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region>  6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>  7
        machine.openshift.io/cluster-api-machine-role: <role>  8
        machine.openshift.io/cluster-api-machine-type: <role>  9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region>  10
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: ""
      providerSpec:
        value:
          apiVersion: ibmcloudproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: ibmcloud-credentials
          image: <infrastructure_id>-rhcos  11
          kind: IBMCloudMachineProviderSpec
```

```
          primaryNetworkInterface:
            securityGroups:
            - <infrastructure_id>-sg-cluster-wide
            - <infrastructure_id>-sg-openshift-net
            subnet: <infrastructure_id>-subnet-compute-<zone> 12
          profile: <instance_profile> 13
          region: <region> 14
          resourceGroup: <resource_group> 15
          userDataSecret:
            name: <role>-user-data 16
          vpc: <vpc_name> 17
          zone: <zone> 18
```

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

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

2 3 8 9 16 The node label to add.

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

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

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

13 Specify the IBM Cloud instance profile .

14 Specify the region to place machines on.

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

17 The VPC name.

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

2.6.3. 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.

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 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.

2.7. 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.

IMPORTANT

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

2.7.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

WARNING

Control plane machines cannot be managed by machine sets.

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.7.2. 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 **<role>** is the node label to add.

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

```
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machine-role: <role> 8
      machine.openshift.io/cluster-api-machine-type: <role> 9
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 10
  spec:
    providerSpec:
      value:
        apiVersion: 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> 13
        primarySubnet: <rhosp_subnet_UUID> 14
        securityGroups:
        - filter: {}
          name: <infrastructure_id>-worker 15
        serverMetadata:
          Name: <infrastructure_id>-worker 16
          openshiftClusterID: <infrastructure_id> 17
        tags:
        - openshiftClusterID=<infrastructure_id> 18
        trunk: true
        userDataSecret:
          name: worker-user-data 19
        availabilityZone: <optional_openstack_availability_zone>
```

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

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

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

4 6 10 Specify the infrastructure ID and node label.

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

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

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.

the same subnet that is used as the value of **machinesSubnet** in the **install-config.yaml** file.

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

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

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

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

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

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

An example machine set that uses SR-IOV networks

```yaml
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
    spec:
      metadata:
      providerSpec:
        value:
          apiVersion: 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>
    kind: OpenstackProviderSpec
    networks:
      - subnets:
        - UUID: <machines_subnet_UUID>
    ports:
      - networkID: <radio_network_UUID>  1
        nameSuffix: radio
        fixedIPs:
          - subnetID: <radio_subnet_UUID>  2
        tags:
          - sriov
          - radio
        vnicType: direct  3
        portSecurity: false  4
      - networkID: <uplink_network_UUID>  5
        nameSuffix: uplink
        fixedIPs:
          - subnetID: <uplink_subnet_UUID>  6
        tags:
          - sriov
          - uplink
        vnicType: direct  7
        portSecurity: false  8
    primarySubnet: <machines_subnet_UUID>
    securityGroups:
    - filter: {}
      name: <infrastructure_id>-<node_role>
    serverMetadata:
      Name: <infrastructure_id>-<node_role>
      openshiftClusterID: <infrastructure_id>
    tags:
    - openshiftClusterID=<infrastructure_id>
    trunk: true
    userDataSecret:
      name: <node_role>-user-data
    availabilityZone: <optional_openstack_availability_zone>
    configDrive: true  9
```

1 5 Enter a network UUID for each port.

2 6 Enter a subnet UUID for each port.

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

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

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

9 The value of the **configDrive** parameter must be **true**.

> **NOTE**
>
> Trunking is enabled for ports that are created by entries in the networks and subnets lists. The names of ports that are created from these lists follow the pattern **<machine_name>-<nameSuffix>**. The **nameSuffix** field is required in port definitions.
>
> You can enable trunking for each port.
>
> Optionally, you can add tags to ports as part of their **tags** lists.

Additional resources

- [Installing a cluster on OpenStack that supports SR-IOV-connected compute machines](#)

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

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

Ports that you define for machines subnets require:

- Allowed address pairs for the API and ingress virtual IP ports

- The compute security group

- Attachment to the machines network and subnet

> **NOTE**
>
> Only parameters that are specific to SR-IOV deployments where port security is disabled are described in this sample. To review a more general sample, see Sample YAML for a machine set custom resource that uses SR-IOV on RHOSP".

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
```

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

❶   Specify allowed address pairs for the API and ingress ports.

❷ ❸ Specify the machines network and subnet.

❹   Specify the compute machines security group.

> **NOTE**
>
> Trunking is enabled for ports that are created by entries in the networks and subnets lists. The names of ports that are created from these lists follow the pattern **<machine_name>-<nameSuffix>**. The **nameSuffix** field is required in port definitions.
>
> You can enable trunking for each port.
>
> Optionally, you can add tags to ports as part of their **tags** lists.

If your cluster uses Kuryr and the RHOSP SR-IOV network has port security disabled, the primary port for compute machines must have:

- The value of the **spec.template.spec.providerSpec.value.networks.portSecurityEnabled** parameter set to **false**.

- For each subnet, the value of the **spec.template.spec.providerSpec.value.networks.subnets.portSecurityEnabled** parameter set to **false**.

- The value of **spec.template.spec.providerSpec.value.securityGroups** set to empty: **[]**.

An example section of a machine set for a cluster on Kuryr that uses SR-IOV and has port security disabled

```
...
        networks:
        - subnets:
          - uuid: <machines_subnet_UUID>
            portSecurityEnabled: false
          portSecurityEnabled: false
        securityGroups: []
...
```

In that case, you can apply the compute security group to the primary VM interface after the VM is created. For example, from a command line:

```
$ openstack port set --enable-port-security --security-group <infrastructure_id>-<node_role> <main_port_ID>
```

2.7.5. 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 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.

2.8. 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.

> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

2.8.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.8.2. 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
        guaranteed_memory_mb:  <memory_size> 22
        os_disk: 23
          size_gb: <disk_size> 24
        network_interfaces: 25
          vnic_profile_id:  <vnic_profile_id> 26
        credentialsSecret:
          name: ovirt-credentials 27
        kind: OvirtMachineProviderSpec
        type: <workload_type> 28
        auto_pinning_policy: <auto_pinning_policy> 29
        hugepages: <hugepages> 30
        affinityGroupsNames:
          - compute 31
        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.

> **WARNING**
>
> The **instance_type_id** field is deprecated and will be removed in a future release.

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: Specify the size of a virtual machine's guaranteed memory in MiB. This is the amount of memory that is guaranteed not to be drained by the ballooning mechanism. For more information, see Memory Ballooning and Optimization Settings Explained.

> **NOTE**
>
> If you are using a version earlier than RHV 4.4.8, see Guaranteed memory requirements for OpenShift on Red Hat Virtualization clusters.

23 Optional: Root disk of the node.

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

25 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.

26 Optional: Specify the vNIC profile ID.

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

28 Optional: Specify the workload type for which the instance is optimized. This value affects the **RHV VM** parameter. Supported values: **desktop**, **server** (default), **high_performance**. **high_performance** improves performance on the VM, but there are limitations. For example, you cannot access the VM with a graphical console. For more information, see Configuring High Performance Virtual Machines, Templates, and Pools in the *Virtual Machine Management Guide*.

29 Optional: AutoPinningPolicy defines the policy that automatically sets CPU and NUMA settings, including pinning to the host for this instance. Supported values: **none**, **resize_and_pin**. For more information, see Setting NUMA Nodes in the *Virtual Machine Management Guide*.

30 Optional: Hugepages is the size in KiB for defining hugepages in a VM. Supported values: **2048** or **1048576**. For more information, see Configuring Huge Pages in the *Virtual Machine Management*

1048576. For more information, see Configuring Huge Pages in the *Virtual Machine Management Guide*.

31 Optional: A list of affinity group names that should be applied to the VMs. The affinity groups must exist in oVirt.

> **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.

2.8.3. 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 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.

2.9. 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.

IMPORTANT

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

2.9.1. 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 4.10 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.10 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

The **MachineAutoscaler** resource automatically scales 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. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability. The autoscaler provides best-effort balancing over the life of a cluster.

2.9.2. 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>  1
  name: <infrastructure_id>-<role>  2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>  3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>  4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>  5
        machine.openshift.io/cluster-api-machine-role: <role>  6
        machine.openshift.io/cluster-api-machine-type: <role>  7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>  8
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: ""  9
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
            - networkName: "<vm_network_name>"  10
          numCPUs: 4
          numCoresPerSocket: 1
```

```
        snapshot: ""
        template: <vm_template_name> (11)
        userDataSecret:
          name: worker-user-data
        workspace:
          datacenter: <vcenter_datacenter_name> (12)
          datastore: <vcenter_datastore_name> (13)
          folder: <vcenter_vm_folder_path> (14)
          resourcepool: <vsphere_resource_pool> (15)
          server: <vcenter_server_ip> (16)
```

(1) (3) (5) Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. 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. This VM network must be where other compute machines reside in the cluster.

(11) Specify the vSphere VM clone of the template to use, such as **user-5ddjd-rhcos**.
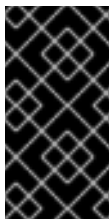
> **IMPORTANT**
>
> Do not specify the original VM template. The VM template must remain off and must be cloned for new RHCOS machines. Starting the VM template configures the VM template as a VM on the platform, which prevents it from being used as a template that machine sets can apply configurations 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.

2.9.3. Minimum required vCenter privileges for machine set management

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

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

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

vSphere object for role	When required	Required privileges
vSphere vCenter	Always	**InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.Update**[1] **StorageProfile.View**[1]
vSphere vCenter Cluster	Always	**Resource.AssignVMToPool**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse**
vSphere Port Group	Always	**Network.Assign**
Virtual Machine Folder	Always	**VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone**

vSphere object for role	When required	Required privileges
vSphere vCenter Datacenter	If the installation program creates the virtual machine folder	**Resource.AssignVMToPool VirtualMachine.Provisioning.DeployTemplate**

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

> **IMPORTANT**
>
> Some CSI drivers and features are in Technology Preview in OpenShift Container Platform 4.10. For more information, see *CSI drivers supported by OpenShift Container Platform*.

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

Example 2.2. Required permissions and propagation settings

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter	Always	Not required	Listed required privileges

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter Datacenter	Existing folder	Not required	**ReadOnly** permission
	Installation program creates the folder	Required	Listed required privileges
vSphere vCenter Cluster	Always	Required	Listed required privileges
vSphere vCenter Datastore	Always	Not required	Listed required privileges
vSphere Switch	Always	Not required	**ReadOnly** permission
vSphere Port Group	Always	Not required	Listed required privileges
vSphere vCenter Virtual Machine Folder	Existing folder	Required	Listed required privileges

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

Additional resources

- For more information about CSI driver and feature support, see CSI drivers supported by OpenShift Container Platform.

2.9.4. Creating a machine set

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

Prerequisites

- Deploy an OpenShift Container Platform cluster.

- Install the OpenShift CLI (**oc**).

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

- 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 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 3. 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.

3.1. 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 for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

3.2. SCALING A MACHINE SET MANUALLY

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

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

Prerequisites

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

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

Procedure

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

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

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

2. View the machines that are in the cluster:

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

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

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

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

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

5. Scale the machine set:

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

Or:

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

TIP

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

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

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

Verification

- Verify the deletion of the intended machine:

```
$ oc get machines
```

3.3. 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.
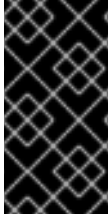
IMPORTANT

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 4. MODIFYING A MACHINE SET

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

On Red Hat Virtualization (RHV), you can also change a machine set to provision new nodes on a different storage domain.

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

4.1. 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
   ```

TIP

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machineset>
  namespace: openshift-machine-api
spec:
  replicas: 0
```

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
```

TIP

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

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

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

4.2. MIGRATING NODES TO A DIFFERENT STORAGE DOMAIN ON RHV

You can migrate the OpenShift Container Platform control plane and compute nodes to a different storage domain in a Red Hat Virtualization (RHV) cluster.

4.2.1. Migrating compute nodes to a different storage domain in RHV

Prerequisites

- You are logged in to the Manager.

- You have the name of the target storage domain.

Procedure

1. Identify the virtual machine template:

   ```
   $ oc get -o jsonpath='{.items[0].spec.template.spec.providerSpec.value.template_name}{"\n"}'
   machineset -A
   ```

2. Create a new virtual machine in the Manager, based on the template you identified. Leave all other settings unchanged. For details, see Creating a Virtual Machine Based on a Template in the Red Hat Virtualization *Virtual Machine Management Guide* .

   TIP

   You do not need to start the new virtual machine.

3. Create a new template from the new virtual machine. Specify the target storage domain under **Target**. For details, see Creating a Template in the Red Hat Virtualization *Virtual Machine Management Guide*.

4. Add a new machine set to the OpenShift Container Platform cluster with the new template.

   a. Get the details of the current machine set:

      ```
      $ oc get machineset -o yaml
      ```

   b. Use these details to create a machine set. For more information see *Creating a machine set* . Enter the new virtual machine template name in the **template_name** field. Use the same template name you used in the **New template** dialog in the Manager.

   c. Note the names of both the old and new machine sets. You need to refer to them in subsequent steps.

5. Migrate the workloads.

   a. Scale up the new machine set. For details on manually scaling machine sets, see *Scaling a machine set manually*.
      OpenShift Container Platform moves the pods to an available worker when the old machine is removed.

   b. Scale down the old machine set.

6. Remove the old machine set:

   ```
   $ oc delete machineset <machineset-name>
   ```

Additional resources

- Creating a machine set.

- Scaling a machine set manually

- Controlling pod placement using the scheduler

4.2.2. Migrating control plane nodes to a different storage domain on RHV

OpenShift Container Platform does not manage control plane nodes, so they are easier to migrate than compute nodes. You can migrate them like any other virtual machine on Red Hat Virtualization (RHV).

Perform this procedure for each node separately.

Prerequisites

- You are logged in to the Manager.

- You have identified the control plane nodes. They are labeled **master** in the Manager.

Procedure

1. Select the virtual machine labeled **master**.

2. Shut down the virtual machine.

3. Click the **Disks** tab.

4. Click the virtual machine's disk.

5. Click **More Actions** and select **Move**.

6. Select the target storage domain and wait for the migration process to complete.

7. Start the virtual machine.

8. Verify that the OpenShift Container Platform cluster is stable:

   ```
   $ oc get nodes
   ```

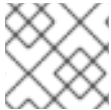   The output should display the node with the status **Ready**.

9. Repeat this procedure for each control plane node.

CHAPTER 5. DELETING A MACHINE

You can delete a specific machine.

5.1. DELETING A SPECIFIC MACHINE

You can delete a specific machine.

> **NOTE**
>
> You cannot delete a control plane machine.

Prerequisites

- Install an OpenShift Container Platform cluster.

- Install the OpenShift CLI (**oc**).

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

Procedure

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

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

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

2. Delete the machine:

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

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

5.2. ADDITIONAL RESOURCES

- [Replacing an unhealthy etcd member](#).

CHAPTER 6. 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.

6.1. ABOUT THE CLUSTER AUTOSCALER

The cluster autoscaler adjusts the size of an OpenShift Container Platform cluster to meet its current deployment needs. It uses declarative, Kubernetes-style arguments to provide infrastructure management that does not rely on objects of a specific cloud provider. The cluster autoscaler has a cluster scope, and is not associated with a particular namespace.

The cluster autoscaler increases the size of the cluster when there are pods that fail to schedule on any of the current worker nodes due to insufficient resources or when another node is necessary to meet deployment needs. The cluster autoscaler does not increase the cluster resources beyond the limits that you specify.

The cluster autoscaler computes the total memory, CPU, and GPU on all nodes the cluster, even though it does not manage the control plane nodes. These values are not single-machine oriented. They are an aggregation of all the resources in the entire cluster. For example, if you set the maximum memory resource limit, the cluster autoscaler includes all the nodes in the cluster when calculating the current memory usage. That calculation is then used to determine if the cluster autoscaler has the capacity to add more worker resources.

> **IMPORTANT**
>
> Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** resource definition that you create is large enough to account for the total possible number of machines in your cluster. This value must encompass the number of control plane machines and the possible number of compute machines that you might scale to.

Every 10 seconds, the cluster autoscaler checks which nodes are unnecessary in the cluster and removes them. The cluster autoscaler considers a node for removal if the following conditions apply:

- The node utilization is less than the *node utilization level* threshold for the cluster. The node utilization level is the sum of the requested resources divided by the allocated resources for the node. If you do not specify a value in the **ClusterAutoscaler** custom resource, the cluster autoscaler uses a default value of **0.5**, which corresponds to 50% utilization.

- The cluster autoscaler can move all pods running on the node to the other nodes.

- The cluster autoscaler does not have scale down disabled annotation.

If the following types of pods are present on a node, the cluster autoscaler will not remove the node:

- Pods with restrictive pod disruption budgets (PDBs).

- Kube-system pods that do not run on the node by default.

- Kube-system pods that do not have a PDB or have a PDB that is too restrictive.

- Pods that are not backed by a controller object such as a deployment, replica set, or stateful set.

- Pods with local storage.

- Pods that cannot be moved elsewhere because of a lack of resources, incompatible node selectors or affinity, matching anti-affinity, and so on.

- Unless they also have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** annotation, pods that have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** annotation.

For example, you set the maximum CPU limit to 64 cores and configure the cluster autoscaler to only create machines that have 8 cores each. If your cluster starts with 30 cores, the cluster autoscaler can add up to 4 more nodes with 32 cores, for a total of 62.

If you configure the cluster autoscaler, additional usage restrictions apply:

- Do not modify the nodes that are in autoscaled node groups directly. All nodes within the same node group have the same capacity and labels and run the same system pods.

- Specify requests for your pods.

- If you have to prevent pods from being deleted too quickly, configure appropriate PDBs.

- Confirm that your cloud provider quota is large enough to support the maximum node pools that you configure.

- Do not run additional node group autoscalers, especially the ones offered by your cloud provider.

The horizontal pod autoscaler (HPA) and the cluster autoscaler modify cluster resources in different ways. The HPA changes the deployment's or replica set's number of replicas based on the current CPU load. If the load increases, the HPA creates new replicas, regardless of the amount of resources available to the cluster. If there are not enough resources, the cluster autoscaler adds resources so that the HPA-created pods can run. If the load decreases, the HPA stops some replicas. If this action causes some nodes to be underutilized or completely empty, the cluster autoscaler deletes the unnecessary nodes.

The cluster autoscaler takes pod priorities into account. The Pod Priority and Preemption feature enables scheduling pods based on priorities if the cluster does not have enough resources, but the cluster autoscaler ensures that the cluster has resources to run all pods. To honor the intention of both features, the cluster autoscaler includes a priority cutoff function. You can use this cutoff to schedule "best-effort" pods, which do not cause the cluster autoscaler to increase resources but instead run only when spare resources are available.

Pods with priority lower than the cutoff value do not cause the cluster to scale up or prevent the cluster from scaling down. No new nodes are added to run the pods, and nodes running these pods might be deleted to free resources.

6.2. CONFIGURING THE CLUSTER AUTOSCALER

First, deploy the cluster autoscaler to manage automatic resource scaling in your OpenShift Container Platform cluster.
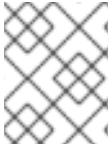
> **NOTE**
>
> Because the cluster autoscaler is scoped to the entire cluster, you can make only one cluster autoscaler for the cluster.

6.2.1. 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
    utilizationThreshold: "0.4"    19
```

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 Optional: 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 Optional: Specify the period to wait before deleting a node after a node has recently been *added*. If you do not specify a value, the default value of **10m** is used.

16 Optional: Specify the period to wait before deleting a node after a node has recently been *deleted*. If you do not specify a value, the default value of **0s** is used.

17 Optional: Specify the period to wait before deleting a node after a scale down failure occurred. If you do not specify a value, the default value of **3m** is used.

18 Optional: 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.

19 Optional: Specify the *node utilization level* below which an unnecessary node is eligible for deletion. The node utilization level is the sum of the requested resources divided by the allocated resources for the node, and must be a value greater than **"0"** but less than **"1"**. If you do not specify a value, the cluster autoscaler uses a default value of **"0.5"**, which corresponds to 50% utilization. This value must be expressed as a string.

> NOTE
>
> When performing a scaling operation, the cluster autoscaler remains within the ranges set in the **ClusterAutoscaler** resource definition, such as the minimum and maximum number of cores to deploy or the amount of memory in the cluster. However, the cluster autoscaler does not correct the current values in your cluster to be within those ranges.
>
> The minimum and maximum CPUs, memory, and GPU values are determined by calculating those resources on all nodes in the cluster, even if the cluster autoscaler does not manage the nodes. For example, the control plane nodes are considered in the total memory in the cluster, even though the cluster autoscaler does not manage the control plane nodes.

6.2.2. 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.

6.3. NEXT STEPS

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

6.4. 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.
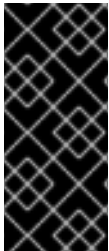
> **IMPORTANT**
>
> You must deploy a machine autoscaler for the cluster autoscaler to scale your machines. The cluster autoscaler uses the annotations on machine sets that the machine autoscaler sets to determine the resources that it can scale. If you define a cluster autoscaler without also defining machine autoscalers, the cluster autoscaler will never scale your cluster.

6.5. 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.

6.5.1. 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 ❷
  maxReplicas: 12 ❸
  scaleTargetRef: ❹
    apiVersion: machine.openshift.io/v1beta1
    kind: MachineSet ❺
    name: worker-us-east-1a ❻
```

❶ 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>-<region>**.

❷ Specify the minimum number machines of the specified type that must remain in the specified zone after the cluster autoscaler initiates cluster scaling. If running in AWS, GCP, Azure, RHOSP, or vSphere, this value can be set to **0**. For other providers, do not set this value to **0**.

You can save on costs by setting this value to **0** for use cases such as running expensive or limited-usage hardware that is used for specialized workloads, or by scaling a machine set with extra large machines. The cluster autoscaler scales the machine set down to zero if the machines are not in use.

> **IMPORTANT**
>
> Do not set the **spec.minReplicas** value to **0** for the three compute machine sets that are created during the OpenShift Container Platform installation process for an installer provisioned infrastructure.

❸ Specify the maximum number machines of the specified type that the cluster autoscaler can deploy in the specified zone after it initiates cluster scaling. Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** resource definition is large enough to allow the machine autoscaler to deploy this number of machines.

❹ In this section, provide values that describe the existing machine set to scale.

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

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

6.5.2. 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:
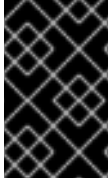
```
$ oc create -f <filename>.yaml ❶
```

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

6.6. ADDITIONAL RESOURCES

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

CHAPTER 7. CREATING INFRASTRUCTURE MACHINE SETS

> **IMPORTANT**
>
> This process is not applicable for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

You can use infrastructure machine sets to create machines that host only infrastructure components, such as the default router, the integrated container image registry, and the components for cluster metrics and monitoring. These infrastructure machines are not counted toward the total number of subscriptions that are required to run the environment.

7.1. 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 HAProxy-based Ingress Controller

- 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 Data Foundation

- Red Hat Advanced Cluster Manager

- Red Hat Advanced Cluster Security for Kubernetes

- Red Hat OpenShift GitOps

- Red Hat OpenShift Pipelines

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

For information about infrastructure nodes and which components can run on infrastructure nodes, see the "Red Hat OpenShift control plane and infrastructure nodes" section in the OpenShift sizing and subscription guide for enterprise Kubernetes document.

7.2. CREATING INFRASTRUCTURE MACHINE SETS FOR PRODUCTION ENVIRONMENTS

In a production deployment, it is recommended that you deploy at least three machine sets to hold infrastructure components. Both OpenShift Logging and Red Hat OpenShift Service Mesh deploy Elasticsearch, which requires three instances to be installed on different nodes. Each of these nodes can be deployed to different availability zones for high availability. A configuration like this requires three different machine sets, one for each availability zone. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability.

7.2.1. Creating machine sets for different clouds

Use the sample machine set for your cloud.

7.2.1.1. Sample YAML for a machine set custom resource on Alibaba Cloud

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-<infra>-<zone> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 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>-<zone> 10
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1
          credentialsSecret:
            name: alibabacloud-credentials
          imageId: <image_id> 11
          instanceType: <instance_type> 12
          kind: AlibabaCloudMachineProviderConfig
```

```
        ramRoleName: <infrastructure_id>-role-worker 13
        regionId: <region> 14
        resourceGroup: 15
          id: <resource_group_id>
          type: ID
        securityGroups:
        - tags: 16
          - Key: Name
            Value: <infrastructure_id>-sg-<role>
          type: Tags
        systemDisk: 17
          category: cloud_essd
          size: <disk_size>
        tag: 18
        - Key: kubernetes.io/cluster/<infrastructure_id>
          Value: owned
        userDataSecret:
          name: <user_data_secret> 19
        vSwitch:
          tags: 20
          - Key: Name
            Value: <infrastructure_id>-vswitch-<zone>
          type: Tags
        vpcId: ""
        zoneId: <zone> 21
    taints: 22
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
```

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 (**oc**) 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 Specify the **<infra>** node label.

4 6 10 Specify the infrastructure ID, **<infra>** node label, and zone.

11 Specify the image to use. Use an image from an existing default machine set for the cluster.

12 Specify the instance type you want to use for the machine set.

13 Specify the name of the RAM role to use for the machine set. Use the value that the installer populates in the default machine set.

14 Specify the region to place machines on.

15 Specify the resource group and type for the cluster. You can use the value that the installer populates in the default machine set, or specify a different one.

16 18 20 Specify the tags to use for the machine set. Minimally, you must include the tags shown in this example, with appropriate values for your cluster. You can include additional tags, including the tags that the installer populates in the default machine set it creates, as needed.

17 Specify the type and size of the root disk. Use the **category** value that the installer populates in the default machine set it creates. If required, specify a different value in gigabytes for **size**.

19 Specify the name of the secret in the user data YAML file that is in the **openshift-machine-api** namespace. Use the value that the installer populates in the default machine set.

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

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

Machine set parameters for Alibaba Cloud usage statistics

The default machine sets that the installer creates for Alibaba Cloud clusters include nonessential tag values that Alibaba Cloud uses internally to track usage statistics. These tags are populated in the **securityGroups**, **tag**, and **vSwitch** parameters of the **spec.template.spec.providerSpec.value** list.

When creating machine sets to deploy additional machines, you must include the required Kubernetes tags. The usage statistics tags are applied by default, even if they are not specified in the machine sets you create. You can also include additional tags as needed.

The following YAML snippets indicate which tags in the default machine sets are optional and which are required.

Tags in spec.template.spec.providerSpec.value.securityGroups

```
spec:
  template:
    spec:
      providerSpec:
        value:
          securityGroups:
          - tags:
            - Key: kubernetes.io/cluster/<infrastructure_id> 1
              Value: owned
            - Key: GISV
              Value: ocp
            - Key: sigs.k8s.io/cloud-provider-alibaba/origin 2
              Value: ocp
            - Key: Name
              Value: <infrastructure_id>-sg-<role> 3
            type: Tags
```

1 **2** Optional: This tag is applied even when not specified in the machine set.

3 Required.

where:

- **<infrastructure_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.

- **<role>** is the node label to add.

Tags in spec.template.spec.providerSpec.value.tag

```
spec:
  template:
    spec:
      providerSpec:
        value:
          tag:
          - Key: kubernetes.io/cluster/<infrastructure_id>    1
            Value: owned
          - Key: GISV    2
            Value: ocp
          - Key: sigs.k8s.io/cloud-provider-alibaba/origin    3
            Value: ocp
```

2 **3** Optional: This tag is applied even when not specified in the machine set.

1 Required.

where **<infrastructure_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.

Tags in spec.template.spec.providerSpec.value.vSwitch

```
spec:
  template:
    spec:
      providerSpec:
        value:
          vSwitch:
            tags:
            - Key: kubernetes.io/cluster/<infrastructure_id>    1
              Value: owned
            - Key: GISV    2
              Value: ocp
            - Key: sigs.k8s.io/cloud-provider-alibaba/origin    3
              Value: ocp
            - Key: Name
              Value: <infrastructure_id>-vswitch-<zone>    4
            type: Tags
```

1 **2** **3** Optional: This tag is applied even when not specified in the machine set.

4 Required.

where:

- **<infrastructure_id>** is the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster.

- **<zone>** is the zone within your region to place machines on.

7.2.1.2. 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/infra: ""**.

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
 labels:
   machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
 name: <infrastructure_id>-infra-<zone> 2
 namespace: openshift-machine-api
spec:
 replicas: 1
 selector:
   matchLabels:
     machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
     machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 4
 template:
   metadata:
     labels:
       machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
       machine.openshift.io/cluster-api-machine-role: <infra> 6
       machine.openshift.io/cluster-api-machine-type: <infra> 7
       machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 8
   spec:
     metadata:
       labels:
         node-role.kubernetes.io/infra: "" 9
     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: <infrastructure_id>-worker-profile 12
         instanceType: m6i.large
         kind: AWSMachineProviderConfig
         placement:
           availabilityZone: us-east-1a
           region: us-east-1
         securityGroups:
```

```
          - filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-worker-sg 13
          subnet:
            filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-private-us-east-1a 14
          tags:
            - name: kubernetes.io/cluster/<infrastructure_id> 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 get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

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.

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

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.

7.2.1.3. 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 a region and creates nodes that are labeled with **node-role.kubernetes.io/infra: ""**.

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

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

```
    name: <infrastructure_id>-infra-<region> 4
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          machine.openshift.io/cluster-api-machineset: <machineset_name> 11
          node-role.kubernetes.io/infra: "" 12
      taints: 13
      - 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/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 14
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 15
          managedIdentity: <infrastructure_id>-identity 16
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: ""
          resourceGroup: <infrastructure_id>-rg 17
```

```
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructure_id>-<role>-subnet 18 19
        userDataSecret:
          name: worker-user-data 20
        vmSize: Standard_D4s_v3
        vnet: <infrastructure_id>-vnet 21
        zone: "1" 22
```

1 5 7 14 16 17 18 21 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

You can obtain the subnet by running the following command:

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

You can obtain the vnet by running the following command:

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

2 3 8 9 12 19 20 Specify the **<infra>** node label.

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

11 Optional: Specify the machine set name to enable the use of availability sets. This setting only applies to new compute machines.

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

15 Specify the region to place machines on.

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

Machine sets running on Azure support non-guaranteed Spot VMs. You can save on costs by using Spot VMs at a lower price compared to standard VMs on Azure. You can configure Spot VMs by adding **spotVMOptions** to the **MachineSet** YAML file.

7.2.1.4. Sample YAML for a machine set custom resource on Azure Stack Hub

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region> 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: "" 11
      taints: 12
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          availabilitySet: <availability_set> 13
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 14
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 15
          managedIdentity: <infrastructure_id>-identity 16
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
```

```
      diskSizeGB: 128
      managedDisk:
        storageAccountType: Premium_LRS
      osType: Linux
  publicIP: false
  publicLoadBalancer: ""
  resourceGroup: <infrastructure_id>-rg 17
  sshPrivateKey: ""
  sshPublicKey: ""
  subnet: <infrastructure_id>-<role>-subnet 18 19
  userDataSecret:
    name: worker-user-data 20
  vmSize: Standard_DS4_v2
  vnet: <infrastructure_id>-vnet 21
  zone: "1" 22
```

1 5 7 14 16 17 18 21 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

You can obtain the subnet by running the following command:

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

You can obtain the vnet by running the following command:

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

2 3 8 9 11 19 20 Specify the **<infra>** node label.

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

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

15 Specify the region to place machines on.

13 Specify the availability set for the cluster.

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

> **NOTE**
>
> Machine sets running on Azure Stack Hub do not support non-guaranteed Spot VMs.

7.2.1.5. Sample YAML for a machine set custom resource on IBM Cloud

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

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

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

(19)
```
        taints:
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
```

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

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

(2) (3) (8) (9) (16) The **<infra>** node label.

(4) (6) (10) The infrastructure ID, **<infra>** node label, and region.

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

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

(13) Specify the IBM Cloud instance profile .

(14) Specify the region to place machines on.

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

(17) The VPC name.

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

(19) The taint to prevent user workloads from being scheduled on infra nodes.

7.2.1.6. 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, **<infrastructure_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<infra>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>  1
  name: <infrastructure_id>-w-a  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>-w-a 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>-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
          gcpMetadata: 12
          - key: <custom_metadata_key>
            value: <custom_metadata_value>
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructure_id>-network 13
            subnetwork: <infrastructure_id>-worker-subnet 14
          projectID: <project_name> 15
          region: us-central1
          serviceAccounts:
          - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com 16 17
            scopes:
            - https://www.googleapis.com/auth/cloud-platform
          tags:
          - <infrastructure_id>-worker 18
          userDataSecret:
            name: worker-user-data
          zone: us-central1-a
```

1 2 3 4 5 8 13 14 16 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
```

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/<infrastructure_id>-worker-a
```

12 Optional: Specify custom metadata in the form of a **key:value** pair. For example use cases, see the GCP documentation for setting custom metadata.

15 17 Specify the name of the GCP project that you use for your cluster.

Machine sets running on GCP support non-guaranteed preemptible VM instances. You can save on costs by using preemptible VM instances at a lower price compared to normal instances on GCP. You can configure preemptible VM instances by adding **preemptible** to the **MachineSet** YAML file.

7.2.1.7. 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> 14
        primarySubnet: <rhosp_subnet_UUID> 15
        securityGroups:
        - filter: {}
          name: <infrastructure_id>-worker 16
        serverMetadata:
          Name: <infrastructure_id>-worker 17
          openshiftClusterID: <infrastructure_id> 18
        tags:
        - openshiftClusterID=<infrastructure_id> 19
        trunk: true
        userDataSecret:
          name: worker-user-data 20
        availabilityZone: <optional_openstack_availability_zone>
```

1 **5** **7** **14** **16** **17** **18** **19** Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

2 **3** **8** **9** **20** Specify the **<infra>** node label.

4 **6** **10** Specify the infrastructure ID and **<infra>** node label.

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

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

13 Required for deployments to multiple networks. If deploying to multiple networks, this list must include the network that is used as the **primarySubnet** value.

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

7.2.1.8. 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
          guaranteed_memory_mb:  <memory_size> 22
          os_disk: 23
            size_gb: <disk_size> 24
          network_interfaces: 25
            vnic_profile_id:  <vnic_profile_id> 26
          credentialsSecret:
            name: ovirt-credentials 27
          kind: OvirtMachineProviderSpec
          type: <workload_type> 28
          auto_pinning_policy: <auto_pinning_policy> 29
          hugepages: <hugepages> 30
          affinityGroupsNames:
            - compute 31
          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.

> ⚠️ **WARNING**
>
> The **instance_type_id** field is deprecated and will be removed in a future release.

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: Specify the size of a virtual machine's guaranteed memory in MiB. This is the amount of memory that is guaranteed not to be drained by the ballooning mechanism. For more information, see Memory Ballooning and Optimization Settings Explained.

> **NOTE**
>
> If you are using a version earlier than RHV 4.4.8, see Guaranteed memory requirements for OpenShift on Red Hat Virtualization clusters.

23 Optional: Root disk of the node.

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

25 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.

26 Optional: Specify the vNIC profile ID.

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

28 Optional: Specify the workload type for which the instance is optimized. This value affects the **RHV VM** parameter. Supported values: **desktop**, **server** (default), **high_performance**. **high_performance** improves performance on the VM, but there are limitations. For example, you cannot access the VM with a graphical console. For more information, see Configuring High Performance Virtual Machines, Templates, and Pools in the *Virtual Machine Management Guide*.

29 Optional: AutoPinningPolicy defines the policy that automatically sets CPU and NUMA settings, including pinning to the host for this instance. Supported values: **none**, **resize_and_pin**. For more information, see Setting NUMA Nodes in the *Virtual Machine Management Guide*.

30 Optional: Hugepages is the size in KiB for defining hugepages in a VM. Supported values: **2048** or **1048576**. For more information, see Configuring Huge Pages in the *Virtual Machine Management Guide*.

31 Optional: A list of affinity group names that should be applied to the VMs. The affinity groups must exist in oVirt.

> **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.

7.2.1.9. 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. This VM network must be where other compute machines reside in the cluster.

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

13 Specify the vCenter 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.

7.2.2. 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 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.

7.2.3. Creating an infrastructure node
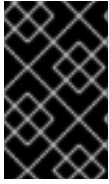
> **IMPORTANT**
>
> See Creating infrastructure machine sets for installer-provisioned infrastructure environments or for any cluster where the control plane nodes are managed by the machine API.

Requirements of the cluster dictate that infrastructure, also called **infra** nodes, be provisioned. The installer only provides provisions for control plane and worker nodes. Worker nodes can be designated as infrastructure nodes or application, also called **app**, nodes through labeling.

Procedure

1. Add a label to the worker node that you want to act as application node:

   ```
   $ oc label node <node-name> node-role.kubernetes.io/app=""
   ```

2. Add a label to the worker nodes that you want to act as infrastructure nodes:

   ```
   $ oc label node <node-name> node-role.kubernetes.io/infra=""
   ```

3. Check to see if applicable nodes now have the **infra** role and **app** roles:

   ```
   $ oc get nodes
   ```

4. Create a default cluster-wide node selector. The default node selector is applied to pods created in all namespaces. This creates an intersection with any existing node selectors on a pod, which additionally constrains the pod's selector.
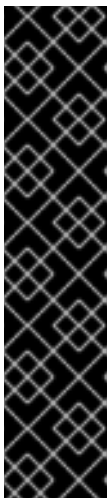
   > **IMPORTANT**
   >
   > If the default node selector key conflicts with the key of a pod's label, then the default node selector is not applied.
   >
   > However, do not set a default node selector that might cause a pod to become unschedulable. For example, setting the default node selector to a specific node role, such as **node-role.kubernetes.io/infra=""**, when a pod's label is set to a different node role, such as **node-role.kubernetes.io/master=""**, can cause the pod to become unschedulable. For this reason, use caution when setting the default node selector to specific node roles.
   >
   > You can alternatively use a project node selector to avoid cluster-wide node selector key conflicts.

   a. Edit the **Scheduler** object:

      ```
      $ oc edit scheduler cluster
      ```

b. Add the **defaultNodeSelector** field with the appropriate node selector:

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
...
spec:
  defaultNodeSelector: topology.kubernetes.io/region=us-east-1 1
...
```

1　This example node selector deploys pods on nodes in the **us-east-1** region by default.

c. Save the file to apply the changes.

You can now move infrastructure resources to the newly labeled **infra** nodes.

Additional resources

- [Moving resources to infrastructure machine sets](#)

7.2.4. 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
3.2.0          31d
00-worker                                             365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0          31d
01-master-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          31d
01-master-kubelet                                     365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0          31d
01-worker-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          31d
01-worker-kubelet                                     365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0          31d
99-master-1ae2a1e0-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          31d
99-master-ssh                                                               3.2.0          31d
99-worker-1ae64748-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          31d
99-worker-ssh                                                               3.2.0          31d
rendered-infra-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          23m
rendered-master-072d4b2da7f88162636902b074e9e28e
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0          31d
rendered-master-3e88ec72aed3886dec061df60d16d1af
02c07496ba0417b3e12b78fb32baf6293d314f79   3.2.0          31d
rendered-master-419bee7de96134963a15fdf9dd473b25
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          17d
rendered-master-53f5c91c7661708adce18739cc0f40fb
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          13d
```

```
rendered-master-a6a357ec18e5bce7f5ac426fc7c5ffcd
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0         7d3h
rendered-master-dc7f874ec77fc4b969674204332da037
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0          31d
rendered-worker-1a75960c52ad18ff5dfa6674eb7e533d
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0          31d
rendered-worker-2640531be11ba43c61d72e82dc634ce6
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0          31d
rendered-worker-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0         7d3h
rendered-worker-4f110718fe88e5f349987854a1147755
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          17d
rendered-worker-afc758e194d6188677eb837842d3b379
02c07496ba0417b3e12b78fb32baf6293d314f79   3.2.0          31d
rendered-worker-daa08cc1e8f5fcdeba24de60cd955cc3
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0          13d
```

You should see a new machine config, with the **rendered-infra-*** prefix.

5. Optional: To deploy changes to a custom pool, create a machine config that uses the custom pool name as the label, such as **infra**. Note that this is not required and only shown for instructional purposes. In this manner, you can apply any custom configurations specific to only your infra nodes.

> **NOTE**
>
> After you create the new machine config pool, the MCO generates a new rendered config for that pool, and associated nodes of that pool reboot to apply the new configuration.

a. Create a machine config:

```
$ cat infra.mc.yaml
```

Example output

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 51-infra
  labels:
    machineconfiguration.openshift.io/role: infra ❶
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - path: /etc/infratest
        mode: 0644
        contents:
          source: data:,infra
```

❶   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.

7.3. 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.

7.3.1. 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
   ```

   TIP

   You can alternatively apply the following YAML to add the taint:

   ```
   kind: Node
   apiVersion: v1
   metadata:
     name: <node_name>
     labels:
       ...
   spec:
     taints:
       - key: node-role.kubernetes.io/infra
         effect: NoSchedule
     ...
   ```

   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 ❶
    key: node-role.kubernetes.io/infra ❷
    operator: Exists ❸
```

❶ Specify the effect that you added to the node.

❷ Specify the key that you added to the node.

❸ 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.

7.4. 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.

7.4.1. 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>  ❶
```

❶ 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   v1.23.0
```

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

7.4.2. 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. Modify the **spec** section of the object to resemble the following YAML:

   ```
   spec:
     affinity:
       podAntiAffinity:
         preferredDuringSchedulingIgnoredDuringExecution:
         - podAffinityTerm:
             namespaces:
             - openshift-image-registry
             topologyKey: kubernetes.io/hostname
           weight: 100
     logLevel: Normal
     managementState: Managed
     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.

7.4.3. 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.

7.4.4. Moving OpenShift Logging resources

You can configure the Cluster Logging Operator to deploy the pods for logging subsystem components, such as Elasticsearch and Kibana, 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

- The logging subsystem for Red Hat OpenShift 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
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 1
        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: 2
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
```

```
    type: kibana

...
```

①② 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   v1.23.0
  ip-10-0-139-146.us-east-2.compute.internal  Ready   master    60m   v1.23.0
  ip-10-0-139-192.us-east-2.compute.internal  Ready   worker    51m   v1.23.0
  ip-10-0-139-241.us-east-2.compute.internal  Ready   worker    51m   v1.23.0
  ip-10-0-147-79.us-east-2.compute.internal   Ready   worker    51m   v1.23.0
  ip-10-0-152-241.us-east-2.compute.internal  Ready   master    60m   v1.23.0
  ip-10-0-139-48.us-east-2.compute.internal   Ready   infra     51m   v1.23.0
  ```

  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.
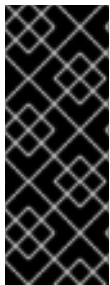
CHAPTER 8. ADDING RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

In OpenShift Container Platform, you can add Red Hat Enterprise Linux (RHEL) compute machines to a user-provisioned infrastructure cluster or an installation-provisioned infrastructure cluster on the **x86_64** architecture. You can use RHEL as the operating system only on compute machines.

8.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.10, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines in your cluster if you use a user-provisioned infrastructure installation on the **x86_64** architecture. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane machines in your cluster.

If you choose to use RHEL compute machines in your cluster, you are responsible for all operating system life cycle management and maintenance. You must perform system updates, apply patches, and complete all other required tasks.

> **IMPORTANT**
>
> - Because removing OpenShift Container Platform from a machine in the cluster requires destroying the operating system, you must use dedicated hardware for any RHEL machines that you add to the cluster.
>
> - Swap memory is disabled on all RHEL machines that you add to your OpenShift Container Platform cluster. You cannot enable swap memory on these machines.

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 in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements:

- You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.

- Production environments must provide compute machines to support your expected workloads. As a cluster administrator, you must calculate the expected workload and add about 10% for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.

- Each system must meet the following hardware requirements:

  - Physical or virtual system, or an instance running on a public or private IaaS.

  - Base OS: RHEL 8.4 or 8.5 with "Minimal" installation option.

> **IMPORTANT**
>
> Adding RHEL 7 compute machines to an OpenShift Container Platform cluster is not supported.
>
> If you have RHEL 7 compute machines that were previously supported in a past OpenShift Container Platform version, you cannot upgrade them to RHEL 8. You must deploy new RHEL 8 hosts, and the old RHEL 7 hosts should be removed. See the "Deleting nodes" section for more information.
>
> For the most recent list of major functionality that has been deprecated or removed within OpenShift Container Platform, refer to the *Deprecated and removed features* section of the OpenShift Container Platform release notes.

- If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See Installing a RHEL 8 system with FIPS mode enabled in the RHEL 8 documentation.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

- NetworkManager 1.0 or later.

- 1 vCPU.

- Minimum 8 GB RAM.

- Minimum 15 GB hard disk space for the file system containing **/var/**.

- Minimum 1 GB hard disk space for the file system containing **/usr/local/bin/**.

- Minimum 1 GB hard disk space for the file system containing its temporary directory. The temporary system directory is determined according to the rules defined in the tempfile module in the Python standard library.

  - Each system must meet any additional requirements for your system provider. For example, if you installed your cluster on VMware vSphere, your disks must be configured according to its storage guidelines and the **disk.enableUUID=true** attribute must be set.

  - Each system must be able to access the cluster's API endpoints by using DNS-resolvable hostnames. Any network security access control that is in place must allow system access to the cluster's API service endpoints.

Additional resources

- Deleting nodes

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 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.

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 8.4 Amazon Machine Images (AMI):

  ```
  $ aws ec2 describe-images --owners 309956199498 \     1
  --query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \     2
  --filters "Name=name,Values=RHEL-8.4*" \     3
  --region us-east-1 \     4
  --output table     5
  ```

  1　The **--owners** command option shows Red Hat images based on the account ID **309956199498**.

  > **IMPORTANT**
  >
  > This account ID is required to display AMI IDs for images that are provided by Red Hat.

  2　The **--query** command option sets how the images are sorted with the parameters **'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]'**. In this case, the images are sorted by the creation date, and the table is structured to show the creation date, the name of the image, and the AMI IDs.

  3　The **--filter** command option sets which version of RHEL is shown. In this example, since the filter is set by **"Name=name,Values=RHEL-8.4*"**, then RHEL 8.4 AMIs are shown.

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

  5　The **--output** command option sets how the results are displayed.

NOTE

When creating a RHEL compute machine for AWS, ensure that the AMI is RHEL 8.4 or 8.5.

Example output

```
-------------------------------------------------------------------------------------------------------
|                                          DescribeImages                                             |
+--------------------------+--------------------------------------------------------+----------------------+
|  2021-03-18T14:23:11.000Z |  RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2  |  ami-
07eeb4db5f7e5a8fb |
|  2021-03-18T14:38:28.000Z |  RHEL-8.4.0_HVM_BETA-20210309-arm64-1-Hourly2-GP2   |  ami-
069d22ec49577d4bf |
|  2021-05-18T19:06:34.000Z |  RHEL-8.4.0_HVM-20210504-arm64-2-Hourly2-GP2        |  ami-
01fc429821bf1f4b4 |
|  2021-05-18T20:09:47.000Z |  RHEL-8.4.0_HVM-20210504-x86_64-2-Hourly2-GP2       |  ami-
0b0af3577fe5e3532 |
+--------------------------+--------------------------------------------------------+----------------------+
```

Additional resources

- You may also manually import RHEL images to AWS .

8.4. PREPARING THE MACHINE TO RUN THE PLAYBOOK

Before you can add compute machines that use Red Hat Enterprise Linux (RHEL) as the operating system to an OpenShift Container Platform 4.10 cluster, you must prepare a RHEL 7 or 8 machine to run an Ansible playbook that adds the new node to the cluster. This machine is not part of the cluster but must be able to access it.

Prerequisites

- Install the OpenShift CLI (**oc**) on the machine that you run the playbook on.

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

Procedure

1. Ensure that the **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the RHEL 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 4.10:

   - On a RHEL 8 system, run the following command:

   ```
   # subscription-manager repos \
       --enable="rhel-8-for-x86_64-baseos-rpms" \
       --enable="rhel-8-for-x86_64-appstream-rpms" \
       --enable="ansible-2.9-for-rhel-8-x86_64-rpms" \
       --enable="rhocp-4.10-for-rhel-8-x86_64-rpms"
   ```

   - On a RHEL 7 system, run the following command:

   ```
   # 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-4.10-rpms"
   ```

   > **IMPORTANT**
   >
   > As of OpenShift Container Platform 4.10.23, running the Ansible playbooks on RHEL 7 is deprecated, and suggested only for the purpose of updating existing installations. Starting with OpenShift Container Platform 4.11, the Ansible playbooks are provided only for RHEL 8.

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.

8.5. 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
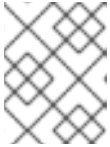
6. Enable only the repositories required by OpenShift Container Platform 4.10:

   ```
   # subscription-manager repos \
       --enable="rhel-8-for-x86_64-baseos-rpms" \
       --enable="rhel-8-for-x86_64-appstream-rpms" \
       --enable="rhocp-4.10-for-rhel-8-x86_64-rpms" \
       --enable="fast-datapath-for-rhel-8-x86_64-rpms"
   ```

7. Stop and disable firewalld on the host:

```
# systemctl disable --now firewalld.service
```

> **NOTE**
>
> You must not enable firewalld later. If you do, you cannot access OpenShift
> Container Platform logs on the worker.

8.6. ATTACHING THE ROLE PERMISSIONS TO RHEL INSTANCE IN AWS

Using the Amazon IAM console in your browser, you may select the needed roles and assign them to a worker node.

Procedure

1. From the AWS IAM console, create your desired IAM role.

2. Attach the IAM role to the desired worker node.

Additional resources

- See Required AWS permissions for IAM roles .

8.7. 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.

8.8. 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 4.10 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 ❶
   #ansible_become=True ❷

   openshift_kubeconfig_path="~/.kube/config" ❸

   [new_workers] ❹
   mycluster-rhel8-0.example.com
   mycluster-rhel8-1.example.com
   ```

   ❶ Specify the user name that runs the Ansible tasks on the remote compute machines.

   ❷ If you do not specify **root** for the **ansible_user**, you must set **ansible_become** to **True** and assign the user sudo permissions.

   ❸ Specify the path and file name of the **kubeconfig** file for your cluster.

   ❹ List each RHEL machine to add to your cluster. You must provide the fully-qualified domain name for each host. This name is the hostname that the cluster uses to access the machine, so set the correct public or private name to access the machine.

2. Navigate to the Ansible playbook directory:

   ```
   $ cd /usr/share/ansible/openshift-ansible
   ```

3. Run the playbook:

   ```
   $ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml ❶
   ```

   ❶ For **<path>**, specify the path to the Ansible inventory file that you created.

8.9. 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  v1.23.0
master-1  Ready    master  63m  v1.23.0
master-2  Ready    master  64m  v1.23.0
```

The output lists all of the machines that you created.

> **NOTE**
>
> The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME       AGE    REQUESTOR                                            CONDITION
csr-8b2br  15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
csr-8vnps  15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

> **NOTE**
>
> Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

  ```
  $ oc adm certificate approve <csr_name> ❶
  ```

  ❶ **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

  ```
  $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
  {{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
  ```

  > **NOTE**
  >
  > Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME        AGE     REQUESTOR                                              CONDITION
   csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
   Pending
   csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
   Pending
   ...
   ```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

   - To approve them individually, run the following command for each valid CSR:

     ```
     $ oc adm certificate approve <csr_name> ❶
     ```

     ❶ **<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  v1.23.0
master-1  Ready    master  73m  v1.23.0
master-2  Ready    master  74m  v1.23.0
worker-0  Ready    worker  11m  v1.23.0
worker-1  Ready    worker  11m  v1.23.0
```

> **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 .

8.10. REQUIRED PARAMETERS FOR THE ANSIBLE HOSTS FILE

You must define the following parameters in the Ansible hosts file before you add Red Hat Enterprise Linux (RHEL) compute machines to your cluster.

Parameter	Description	Values
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.

8.10.1. 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>  ❶
   ```

   ❶   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  ❶
   ```

   ❶   Specify the node name of the RHCOS compute machine that you isolated.

   c. Delete the node:

   ```
   $ oc delete nodes <node_name>  ❶
   ```

   ❶   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 9. 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.

9.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.10, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines in your cluster if you use a user-provisioned infrastructure installation on the **x86_64** architecture. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane machines in your cluster.

If you choose to use RHEL compute machines in your cluster, you are responsible for all operating system life cycle management and maintenance. You must perform system updates, apply patches, and complete all other required tasks.

> **IMPORTANT**
>
> - Because removing OpenShift Container Platform from a machine in the cluster requires destroying the operating system, you must use dedicated hardware for any RHEL machines that you add to the cluster.
>
> - Swap memory is disabled on all RHEL machines that you add to your OpenShift Container Platform cluster. You cannot enable swap memory on these machines.

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

9.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

The Red Hat Enterprise Linux (RHEL) compute machine hosts in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements:

- You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.

- Production environments must provide compute machines to support your expected workloads. As a cluster administrator, you must calculate the expected workload and add about 10% for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.

- Each system must meet the following hardware requirements:

  - Physical or virtual system, or an instance running on a public or private IaaS.

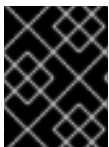 - Base OS: RHEL 8.4 or 8.5 with "Minimal" installation option.

> **IMPORTANT**
>
> Adding RHEL 7 compute machines to an OpenShift Container Platform cluster is not supported.
>
> If you have RHEL 7 compute machines that were previously supported in a past OpenShift Container Platform version, you cannot upgrade them to RHEL 8. You must deploy new RHEL 8 hosts, and the old RHEL 7 hosts should be removed. See the "Deleting nodes" section for more information.
>
> For the most recent list of major functionality that has been deprecated or removed within OpenShift Container Platform, refer to the *Deprecated and removed features* section of the OpenShift Container Platform release notes.

- If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See [Installing a RHEL 8 system with FIPS mode enabled](#) in the RHEL 8 documentation.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

- NetworkManager 1.0 or later.

- 1 vCPU.

- Minimum 8 GB RAM.

- Minimum 15 GB hard disk space for the file system containing **/var**/.

- Minimum 1 GB hard disk space for the file system containing **/usr**/**local**/**bin**/.

- Minimum 1 GB hard disk space for the file system containing its temporary directory. The temporary system directory is determined according to the rules defined in the tempfile module in the Python standard library.

  - Each system must meet any additional requirements for your system provider. For example, if you installed your cluster on VMware vSphere, your disks must be configured according to its [storage guidelines](#) and the **disk.enableUUID=true** attribute must be set.

  - Each system must be able to access the cluster's API endpoints by using DNS-resolvable hostnames. Any network security access control that is in place must allow system access to the cluster's API service endpoints.

Additional resources

- [Deleting nodes](#)

9.2.1. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The

machine-approver cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

9.3. PREPARING AN IMAGE FOR YOUR CLOUD

Amazon Machine Images (AMI) are required 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.

9.3.1. Listing latest available RHEL images on AWS

AMI IDs correspond to native boot images for AWS. Because an AMI must exist before the EC2 instance is provisioned, you will need to know the AMI ID before configuration. The AWS Command Line Interface (CLI) is used to list the available Red Hat Enterprise Linux (RHEL) image IDs.

Prerequisites

- You have installed the AWS CLI.

Procedure

- Use this command to list RHEL 8.4 Amazon Machine Images (AMI):

  ```
  $ aws ec2 describe-images --owners 309956199498 \    1
  --query 'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]' \    2
  --filters "Name=name,Values=RHEL-8.4*" \    3
  --region us-east-1 \    4
  --output table    5
  ```

  1 The **--owners** command option shows Red Hat images based on the account ID **309956199498**.

  > **IMPORTANT**
  >
  > This account ID is required to display AMI IDs for images that are provided by Red Hat.

  2 The **--query** command option sets how the images are sorted with the parameters **'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]'**. In this case, the images are sorted by the creation date, and the table is structured to show the creation date, the name of the image, and the AMI IDs.

  3 The **--filter** command option sets which version of RHEL is shown. In this example, since the filter is set by **"Name=name,Values=RHEL-8.4*"**, then RHEL 8.4 AMIs are shown.

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

  5 The **--output** command option sets how the results are displayed.

> **NOTE**
>
> When creating a RHEL compute machine for AWS, ensure that the AMI is RHEL 8.4 or 8.5.

Example output

```
-------------------------------------------------------------------------------------------------
|                                   DescribeImages                                   |
+--------------------------+-------------------------------------------------+----------------------+
| 2021-03-18T14:23:11.000Z | RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2 | ami-
07eeb4db5f7e5a8fb |
| 2021-03-18T14:38:28.000Z | RHEL-8.4.0_HVM_BETA-20210309-arm64-1-Hourly2-GP2   | ami-
069d22ec49577d4bf |
| 2021-05-18T19:06:34.000Z | RHEL-8.4.0_HVM-20210504-arm64-2-Hourly2-GP2       | ami-
01fc429821bf1f4b4 |
| 2021-05-18T20:09:47.000Z | RHEL-8.4.0_HVM-20210504-x86_64-2-Hourly2-GP2      | ami-
0b0af3577fe5e3532 |
+--------------------------+-------------------------------------------------+----------------------+
```

Additional resources

- You may also manually import RHEL images to AWS .

9.4. PREPARING 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 4.10:

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.10-for-rhel-8-x86_64-rpms" \
    --enable="fast-datapath-for-rhel-8-x86_64-rpms"
```

7. Stop and disable firewalld on the host:

```
# systemctl disable --now firewalld.service
```

> **NOTE**
>
> You must not enable firewalld later. If you do, you cannot access OpenShift Container Platform logs on the worker.

9.5. ATTACHING THE ROLE PERMISSIONS TO RHEL INSTANCE IN AWS

Using the Amazon IAM console in your browser, you may select the needed roles and assign them to a worker node.

Procedure

1. From the AWS IAM console, create your desired IAM role.

2. Attach the IAM role to the desired worker node.

Additional resources

*   See Required AWS permissions for IAM roles .

9.6. 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.

9.7. 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 4.10 cluster.

Prerequisites

- Your OpenShift Container Platform cluster already contains RHEL compute nodes.

- The **hosts** file that you used to add the first RHEL compute machines to your cluster is on the machine that you use the run the playbook.

- The machine that you run the playbook on must be able to access all of the RHEL hosts. You can use any method that your company allows, including a bastion with an SSH proxy or a VPN.

- The **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the machine that you use the run the playbook.

- You must prepare the RHEL hosts for installation.

- Configure a user on the machine that you run the playbook on that has SSH access to all of the RHEL hosts.

- If you use SSH key-based authentication, you must manage the key with an SSH agent.

- Install the OpenShift CLI (**oc**) on the machine that you run the playbook on.

Procedure

1. Open the Ansible inventory file at **/<path>/inventory/hosts** that defines your compute machine hosts and required variables.

2. Rename the **[new_workers]** section of the file to **[workers]**.

3. Add a **[new_workers]** section to the file and define the fully-qualified domain names for each new host. The file resembles the following example:

```
[all:vars]
ansible_user=root
#ansible_become=True

openshift_kubeconfig_path="~/.kube/config"

[workers]
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com

[new_workers]
mycluster-rhel8-2.example.com
mycluster-rhel8-3.example.com
```

In this example, the **mycluster-rhel8-0.example.com** and **mycluster-rhel8-1.example.com** machines are in the cluster and you add the **mycluster-rhel8-2.example.com** and **mycluster-rhel8-3.example.com** machines.

4. Navigate to the Ansible playbook directory:

```
$ cd /usr/share/ansible/openshift-ansible
```

5. Run the scaleup playbook:

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml ❶
```

❶ For **<path>**, specify the path to the Ansible inventory file that you created.

9.8. 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  v1.23.0
master-1  Ready    master  63m  v1.23.0
master-2  Ready    master  64m  v1.23.0
```

The output lists all of the machines that you created.

> **NOTE**
>
> The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE    REQUESTOR                                          CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

> **NOTE**
>
> Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name>  ❶
```

❶ **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

> **NOTE**
>
> Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME       AGE     REQUESTOR                                           CONDITION
csr-bfd72  5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv  5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name>  ❶
```

❶ **<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  v1.23.0
```

```
master-1  Ready    master  73m  v1.23.0
master-2  Ready    master  74m  v1.23.0
worker-0  Ready    worker  11m  v1.23.0
worker-1  Ready    worker  11m  v1.23.0
```

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.

9.9. REQUIRED PARAMETERS FOR THE ANSIBLE HOSTS FILE

You must define the following parameters in the Ansible hosts file before you add Red Hat Enterprise Linux (RHEL) compute machines to your cluster.

Parameter	Description	Values
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 10. USER-PROVISIONED INFRASTRUCTURE

10.1. ADDING COMPUTE MACHINES TO CLUSTERS WITH USER-PROVISIONED INFRASTRUCTURE

You can add compute machines to a cluster on user-provisioned infrastructure either as part of the installation process or after installation. The post-installation process requires some of the same configuration files and parameters that were used during installation.

10.1.1. Adding compute machines to Amazon Web Services

To add more compute machines to your OpenShift Container Platform cluster on Amazon Web Services (AWS), see Adding compute machines to AWS by using CloudFormation templates .

10.1.2. Adding compute machines to Microsoft Azure

To add more compute machines to your OpenShift Container Platform cluster on Microsoft Azure, see Creating additional worker machines in Azure .

10.1.3. Adding compute machines to Azure Stack Hub

To add more compute machines to your OpenShift Container Platform cluster on Azure Stack Hub, see Creating additional worker machines in Azure Stack Hub .

10.1.4. Adding compute machines to Google Cloud Platform

To add more compute machines to your OpenShift Container Platform cluster on Google Cloud Platform (GCP), see Creating additional worker machines in GCP .

10.1.5. Adding compute machines to vSphere

To add more compute machines to your OpenShift Container Platform cluster on vSphere, see Adding compute machines to vSphere.

10.1.6. Adding compute machines to bare metal

To add more compute machines to your OpenShift Container Platform cluster on bare metal, see Adding compute machines to bare metal .

10.2. ADDING COMPUTE MACHINES TO AWS BY USING CLOUDFORMATION TEMPLATES

You can add more compute machines to your OpenShift Container Platform cluster on Amazon Web Services (AWS) that you created by using the sample CloudFormation templates.
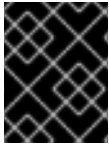
10.2.1. Prerequisites

- You installed your cluster on AWS by using the provided AWS CloudFormation templates.

- You have the JSON file and CloudFormation template that you used to create the compute machines during cluster installation. If you do not have these files, you must recreate them by following the instructions in the installation procedure.

10.2.2. Adding more compute machines to your AWS cluster by using CloudFormation templates

You can add more compute machines to your OpenShift Container Platform cluster on Amazon Web Services (AWS) that you created by using the sample CloudFormation templates.

IMPORTANT

The CloudFormation template creates a stack that represents one compute machine. You must create a stack for each compute machine.

NOTE

If you do not use the provided CloudFormation template to create your compute nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- You installed an OpenShift Container Platform cluster by using CloudFormation templates and have access to the JSON file and CloudFormation template that you used to create the compute machines during cluster installation.

- You installed the AWS CLI.

Procedure

1. Create another compute stack.

    a. Launch the template:

    ```
    $ aws cloudformation create-stack --stack-name <name> \ 1
        --template-body file://<template>.yaml \ 2
        --parameters file://<parameters>.json 3
    ```

    1 **<name>** is the name for the CloudFormation stack, such as **cluster-workers**. You must provide the name of this stack if you remove the cluster.

    2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

    3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

    b. Confirm that the template components exist:

    ```
    $ aws cloudformation describe-stacks --stack-name <name>
    ```

2. Continue to create compute stacks until you have created enough compute machines for your cluster.

10.2.3. 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  v1.23.0
   master-1  Ready    master  63m  v1.23.0
   master-2  Ready    master  64m  v1.23.0
   ```

   The output lists all of the machines that you created.

   > **NOTE**
   >
   > The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME        AGE   REQUESTOR                                                      CONDITION
   csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper   Pending
   csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper   Pending
   ...
   ```

   In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

> **NOTE**
>
> Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

  ```
  $ oc adm certificate approve <csr_name>  ❶
  ```

  ❶ **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

  ```
  $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
  ```

  > **NOTE**
  >
  > Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME        AGE     REQUESTOR                                           CONDITION
   csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
   Pending
   csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
   Pending
   ...
   ```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name>  ❶
```

❶ **<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  v1.23.0
master-1  Ready    master  73m  v1.23.0
master-2  Ready    master  74m  v1.23.0
worker-0  Ready    worker  11m  v1.23.0
worker-1  Ready    worker  11m  v1.23.0
```

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

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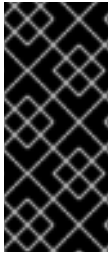
- For more information on CSRs, see Certificate Signing Requests .

10.3. ADDING COMPUTE MACHINES TO VSPHERE

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

10.3.1. Prerequisites

- You installed a cluster on vSphere .

- You have installation media and Red Hat Enterprise Linux CoreOS (RHCOS) images that you used to create your cluster. If you do not have these files, you must obtain them by following the instructions in the installation procedure.

IMPORTANT

If you do not have access to the Red Hat Enterprise Linux CoreOS (RHCOS) images that were used to create your cluster, you can add more compute machines to your OpenShift Container Platform cluster with newer versions of Red Hat Enterprise Linux CoreOS (RHCOS) images. For instructions, see Adding new nodes to UPI cluster fails after upgrading to OpenShift 4.6+.

10.3.2. Adding more compute machines to a cluster in vSphere

You can add more compute machines to a user-provisioned OpenShift Container Platform cluster on VMware vSphere.

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**.

   NOTE

   Ensure that all virtual machine names across a vSphere installation are unique.

   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.

10.3.3. 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  v1.23.0
   master-1  Ready    master  63m  v1.23.0
   master-2  Ready    master  64m  v1.23.0
   ```

   The output lists all of the machines that you created.

   > **NOTE**
   >
   > The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME       AGE    REQUESTOR                                              CONDITION
   csr-8b2br  15m    system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper  Pending
   csr-8vnps  15m    system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper  Pending
   ...
   ```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

> **NOTE**
>
> Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

  ```
  $ oc adm certificate approve <csr_name>  ❶
  ```

  ❶ **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

  ```
  $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
  {{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
  ```

  > **NOTE**
  >
  > Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

```
NAME       AGE    REQUESTOR                                          CONDITION
csr-bfd72  5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv  5m26s  system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

   - To approve them individually, run the following command for each valid CSR:

     ```
     $ oc adm certificate approve <csr_name>  1
     ```

     1 **<csr_name>** is the name of a CSR from the list of current CSRs.

   - To approve all pending CSRs, run the following command:

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS   ROLES   AGE  VERSION
   master-0  Ready    master  73m  v1.23.0
   master-1  Ready    master  73m  v1.23.0
   master-2  Ready    master  74m  v1.23.0
   worker-0  Ready    worker  11m  v1.23.0
   worker-1  Ready    worker  11m  v1.23.0
   ```

   > **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 .

10.4. ADDING COMPUTE MACHINES TO BARE METAL

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

10.4.1. Prerequisites

- You installed a cluster on bare metal .

- You have installation media and Red Hat Enterprise Linux CoreOS (RHCOS) images that you used to create your cluster. If you do not have these files, you must obtain them by following the instructions in the installation procedure.

- If a DHCP server is available for your user-provisioned infrastructure, you have added the details for the additional compute machines to your DHCP server configuration. This includes a persistent IP address, DNS server information, and a hostname for each machine.

- You have updated your DNS configuration to include the record name and IP address of each compute machine that you are adding. You have validated that DNS lookup and reverse DNS lookup resolve correctly.
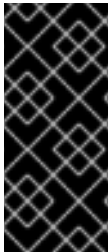
> **IMPORTANT**
>
> If you do not have access to the Red Hat Enterprise Linux CoreOS (RHCOS) images that were used to create your cluster, you can add more compute machines to your OpenShift Container Platform cluster with newer versions of Red Hat Enterprise Linux CoreOS (RHCOS) images. For instructions, see Adding new nodes to UPI cluster fails after upgrading to OpenShift 4.6+.

10.4.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you add more compute machines to a cluster that you installed on bare metal infrastructure, you must create RHCOS machines for it to use. You can either use an ISO image or network PXE booting to create the machines.

> **NOTE**
>
> You must use the same ISO image that you used to install a cluster to deploy all new nodes in a cluster. It is recommended to use the same Ignition config file. The nodes automatically upgrade themselves on the first boot before running the workloads. You can add the nodes before or after the upgrade.

10.4.2.1. 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.

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.install_dev=sda ❶
coreos.inst.ignition_url=http://example.com/worker.ign ❷
```

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

❷ Specify the URL of the compute Ignition config file. Only HTTP and HTTPS protocols are supported.

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.

10.4.2.2. 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>-live-kernel-<architecture> ❶
         APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
     <architecture>.img coreos.inst.install_dev=/dev/sda
     coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
     coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
     <architecture>.img ❷
     ```

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

     ❷ Specify locations of the RHCOS files that you uploaded to your HTTP server. The **initrd** parameter value is the location of the live **initramfs** file, the

coreos.inst.ignition_url parameter value is the location of the worker Ignition config file, and the **coreos.live.rootfs_url** parameter value is the location of the live **rootfs** file. The **coreos.inst.ignition_url** and **coreos.live.rootfs_url** parameters only support HTTP and HTTPS.

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>-live-kernel-<architecture> initrd=main
  coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
  coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
  ```
  1

  ```
  initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
  ```
  2

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=main** argument is needed for booting on UEFI systems, the **coreos.inst.ignition_url** parameter value is the location of the worker Ignition config file, and the **coreos.live.rootfs_url** parameter value is the location of the live **rootfs** file. The **coreos.inst.ignition_url** and **coreos.live.rootfs_url** parameters only support HTTP and HTTPS.

2 Specify the location of the **initramfs** file that you uploaded to your HTTP server.

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?.

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

10.4.3. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

   ```
   $ oc get nodes
   ```

Example output

```
NAME      STATUS   ROLES  AGE  VERSION
master-0  Ready    master 63m  v1.23.0
master-1  Ready    master 63m  v1.23.0
master-2  Ready    master 64m  v1.23.0
```

The output lists all of the machines that you created.

> **NOTE**
>
> The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME       AGE    REQUESTOR                                             CONDITION
csr-8b2br  15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
csr-8vnps  15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

> **NOTE**
>
> Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

  ```
  $ oc adm certificate approve <csr_name> ❶
  ```

  ❶ **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

  ```
  $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
  {{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
  ```

  > **NOTE**
  >
  > Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME        AGE     REQUESTOR                                              CONDITION
   csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
   Pending
   csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
   Pending
   ...
   ```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

   - To approve them individually, run the following command for each valid CSR:

     ```
     $ oc adm certificate approve <csr_name> ❶
     ```

     ❶ **<csr_name>** is the name of a CSR from the list of current CSRs.

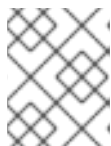- To approve all pending CSRs, run the following command:

```
$ 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  v1.23.0
master-1  Ready    master  73m  v1.23.0
master-2  Ready    master  74m  v1.23.0
worker-0  Ready    worker  11m  v1.23.0
worker-1  Ready    worker  11m  v1.23.0
```
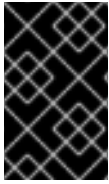
> **NOTE**
>
> It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

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

CHAPTER 11. 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 for clusters with manually provisioned machines. You can use the advanced machine management and scaling capabilities only in clusters where the Machine API is operational.

11.1. ABOUT MACHINE HEALTH CHECKS

Machine health checks automatically repair unhealthy machines in a particular machine pool.

To monitor machine health, create a resource to define the configuration for a controller. Set a condition to check, such as staying in the **NotReady** status for five minutes or displaying a permanent condition in the node-problem-detector, and a label for the set of machines to monitor.

> **NOTE**
>
> You cannot apply a machine health check to a machine with the master role.

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

To limit disruptive impact of the machine deletion, the controller drains and deletes only one node at a time. If there are more unhealthy machines than the **maxUnhealthy** threshold allows for in the targeted pool of machines, remediation stops and therefore enables manual intervention.

> **NOTE**
>
> Consider the timeouts carefully, accounting for workloads and requirements.
>
> - Long timeouts can result in long periods of downtime for the workload on the unhealthy machine.
>
> - Too short timeouts can result in a remediation loop. For example, the timeout for checking the **NotReady** status must be long enough to allow the machine to complete the startup process.

To stop the check, remove the resource.

11.1.1. Limitations when deploying machine health checks

There are limitations to consider before deploying a machine health check:

- Only machines owned by a machine set are remediated by a machine health check.

- 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.

11.2. SAMPLE MACHINEHEALTHCHECK RESOURCE

The **MachineHealthCheck** resource for all cloud-based installation types, and other than bare metal, resembles the following YAML file:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
 name: example 1
 namespace: openshift-machine-api
spec:
 selector:
  matchLabels:
   machine.openshift.io/cluster-api-machine-role: <role> 2
   machine.openshift.io/cluster-api-machine-type: <role> 3
   machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 4
 unhealthyConditions:
 - type:    "Ready"
   timeout: "300s" 5
   status: "False"
 - type:    "Ready"
   timeout: "300s" 6
   status: "Unknown"
 maxUnhealthy: "40%" 7
 nodeStartupTimeout: "10m" 8
```

1 Specify the name of the machine health check to deploy.

2 3 Specify a label for the machine pool that you want to check.

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

5 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 machines allowed to be concurrently remediated in the targeted pool. This can be set as a percentage or an integer. If the number of unhealthy machines exceeds the limit set

by **maxUnhealthy**, remediation is not performed.

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.

11.2.1. Short-circuiting machine health check remediation

Short circuiting ensures that machine health checks remediate machines only when the cluster is healthy. Short-circuiting is configured through the **maxUnhealthy** field in the **MachineHealthCheck** resource.

If the user defines a value for the **maxUnhealthy** field, before remediating any machines, the **MachineHealthCheck** compares the value of **maxUnhealthy** with the number of machines within its target pool that it has determined to be unhealthy. Remediation is not performed if the number of unhealthy machines exceeds the **maxUnhealthy** limit.
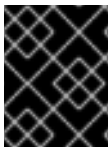
> **IMPORTANT**
>
> If **maxUnhealthy** is not set, the value defaults to **100%** and the machines are remediated regardless of the state of the cluster.

The appropriate **maxUnhealthy** value depends on the scale of the cluster you deploy and how many machines the **MachineHealthCheck** covers. For example, you can use the **maxUnhealthy** value to cover multiple machine sets across multiple availability zones so that if you lose an entire zone, your **maxUnhealthy** setting prevents further remediation within the cluster. In global Azure regions that do not have multiple availability zones, you can use availability sets to ensure high availability.

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

11.2.1.1. 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.

11.2.1.2. 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
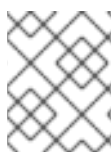
> **NOTE**
>
> The allowed number of machines is rounded down when the percentage of **maxUnhealthy** machines that are checked is not a whole number.

11.3. CREATING A MACHINEHEALTHCHECK RESOURCE

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
```

You can configure and deploy a machine health check to detect and repair unhealthy bare metal nodes.

11.4. ABOUT POWER-BASED REMEDIATION OF BARE METAL

In a bare metal cluster, remediation of nodes is critical to ensuring the overall health of the cluster. Physically remediating a cluster can be challenging and any delay in putting the machine into a safe or an operational state increases the time the cluster remains in a degraded state, and the risk that subsequent failures might bring the cluster offline. Power-based remediation helps counter such challenges.

Instead of reprovisioning the nodes, power-based remediation uses a power controller to power off an inoperable node. This type of remediation is also called power fencing.

OpenShift Container Platform uses the **MachineHealthCheck** controller to detect faulty bare metal nodes. Power-based remediation is fast and reboots faulty nodes instead of removing them from the cluster.

Power-based remediation provides the following capabilities:

- Allows the recovery of control plane nodes

- Reduces the risk data loss in hyperconverged environments

- Reduces the downtime associated with recovering physical machines

11.4.1. MachineHealthChecks on bare metal

Machine deletion on bare metal cluster triggers reprovisioning of a bare metal host. Usually bare metal

reprovisioning is a lengthy process, during which the cluster is missing compute resources and applications might be interrupted. 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.

11.4.2. Understanding the remediation process

The remediation process operates as follows:

1. The MachineHealthCheck (MHC) controller detects that a node is unhealthy.

2. The MHC notifies the bare metal machine controller which requests to power-off the unhealthy node.

3. After the power is off, the node is deleted, which allows the cluster to reschedule the affected workload on other nodes.

4. The bare metal machine controller requests to power on the node.

5. After the node is up, the node re-registers itself with the cluster, resulting in the creation of a new node.

6. After the node is recreated, the bare metal machine controller restores the annotations and labels that existed on the unhealthy node before its deletion.
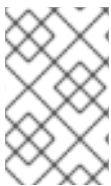
> **NOTE**
>
> If the power operations did not complete, the bare metal machine controller triggers the reprovisioning of the unhealthy node unless this is a control plane node or a node that was provisioned externally.

11.4.3. Creating a MachineHealthCheck resource for bare metal

Prerequisites

- The OpenShift Container Platform is installed using installer-provisioned infrastructure (IPI).

- Access to Baseboard Management Controller (BMC) credentials (or BMC access to each node)

- Network access to the BMC interface of the unhealthy node.

Procedure

1. Create a **healthcheck.yaml** file that contains the definition of your machine health check.

2. Apply the **healthcheck.yaml** file to your cluster using the following command:

```
$ oc apply -f healthcheck.yaml
```

Sample **MachineHealthCheck** resource 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 the node condition. If the condition is met for the duration of the timeout, the machine will be remediated. Long timeouts can result in long periods of downtime for a workload on an unhealthy machine.

8 Specify the amount of machines allowed to be concurrently remediated in the targeted pool. This can be set as a percentage or an integer. If the number of unhealthy machines exceeds the limit set by **maxUnhealthy**, remediation is not performed.

9 Specify the timeout duration that a machine health check must wait for a node to join the cluster before a machine is determined to be unhealthy.

> **NOTE**
>
> The **matchLabels** are examples only; you must map your machine groups based on your specific needs.

<mgmt-troubleshooting-issue-power-remediation_deploying-machine-health-checks><title>Troubleshooting issues with power-based remediation</title>
To troubleshoot an issue with power-based remediation, verify the following:

- You have access to the BMC.

- BMC is connected to the control plane node that is responsible for running the remediation task.

</mgmt-troubleshooting-issue-power-remediation_deploying-machine-health-checks>