OpenShift Container Platform 4.4

Machine management

Adding and maintaining cluster machines

Last Updated: 2021-02-14

OpenShift Container Platform 4.4 Machine management

Adding and maintaining cluster machines

Legal Notice

Abstract

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

Table of Contents

CHAPTER 1. CREATING MACHINE SETS

1.1. CREATING A MACHINE SET IN AWS

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

1.1.1. Machine API overview

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

For OpenShift Container Platform 4.4 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.4 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

In OpenShift Container Platform version 3.11, you could not roll out a multi-zone architecture easily because the cluster did not manage machine provisioning. Beginning with OpenShift Container Platform version 4.1, this process is easier. Each machine set is scoped to a single zone, so the installation

program sends out machine sets across availability zones on your behalf. And then because your compute is dynamic, and in the face of a zone failure, you always have a zone for when you must rebalance your machines. The autoscaler provides best-effort balancing over the life of a cluster.

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructureID> 1
  name: <infrastructureID>-<role>-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructureID>-<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: <infrastructureID>-worker-profile 11
          instanceType: m4.large
          kind: AWSMachineProviderConfig
          placement:
```

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

1 3 5 11 12 13 14 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

```
$ oc 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.

1.1.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

```
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

....

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


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.

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

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

For example:

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

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

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

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

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

NOTE

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

Next steps

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

1.2. CREATING A MACHINE SET IN AZURE

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

1.2.1. Machine API overview

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

For OpenShift Container Platform 4.4 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.4 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructureID> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructureID>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<region> 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" 11
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructureID>-
rg/providers/Microsoft.Compute/images/<infrastructureID>
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: <infrastructureID>-identity 12
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
```

```
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructureID>-rg (13)
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructureID>-<role>-subnet (14) (15)
        userDataSecret:
          name: <role>-user-data (16)
        vmSize: Standard_D2s_v3
        vnet: <infrastructureID>-vnet (17)
        zone: "1" (18)
```

(1) (5) (7) (12) (13) (14) (17) Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

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

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

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

1.2.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

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

....

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


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.

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

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

For example:

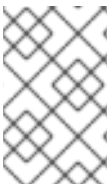
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api
```

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

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

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

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



NOTE

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

Next steps

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

1.3. CREATING A MACHINE SET IN GCP

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

1.3.1. Machine API overview

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

For OpenShift Container Platform 4.4 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.4 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

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

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

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

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

```
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructureID> 1
  name: <infrastructureID>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 9
      providerSpec:
        value:
          apiVersion: gcpprovider.openshift.io/v1beta1
          canIPForward: false
          credentialsSecret:
            name: gcp-cloud-credentials
          deletionProtection: false
          disks:
          - autoDelete: true
            boot: true
            image: <infrastructureID>-rhcos-image 10
            labels: null
            sizeGb: 128
            type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructureID>-network 11
            subnetwork: <infrastructureID>-<role>-subnet 12
          projectID: <project_name> 13
          region: us-central1
          serviceAccounts:
          - email: <infrastructureID>-w@<project_name>.iam.gserviceaccount.com 14 15
            scopes:
            - https://www.googleapis.com/auth/cloud-platform
          tags:
          - <infrastructureID>-<role> 16
```

```
        userDataSecret:
          name: worker-user-data
        zone: us-central1-a
```

1 2 3 4 5 8 10 11 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
```

12 16 Specify the infrastructure ID and node label.

6 7 9 Specify the node label to add.

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

1.3.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

   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

   ....
   ```

```
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


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.

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

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

For example:

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

status:
  addresses:
  - address: 10.0.133.18
    type: InternalIP
  - address: ""
    type: ExternalDNS
  - address: ip-10-0-133-18.ec2.internal
    type: InternalDNS
  lastUpdated: "2019-05-03T10:38:17Z"
  nodeRef:
    kind: Node
```

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

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

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

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

NOTE

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

Next steps

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

1.4. CREATING A MACHINE SET ON OPENSTACK

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

1.4.1. Machine API overview

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

For OpenShift Container Platform 4.4 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.4 offers an elastic, dynamic provisioning method on top of public or private cloud infrastructure.

The two primary resources are:

Machines

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

machine type and required metadata.

Machine sets

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

The following custom resources add more capabilities to your cluster:

Machine autoscaler

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

Cluster autoscaler

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

Machine health check

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

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

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

This sample YAML defines a machine set that runs on Red Hat OpenStack Platform (RHOSP) and creates nodes that are labeled with **node-role.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.

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

```
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 7
        machine.openshift.io/cluster-api-machine-role: <node_role> 8
        machine.openshift.io/cluster-api-machine-type: <node_role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role> 10
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          kind: OpenstackProviderSpec
          networks:
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_ID>
          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 11
```

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

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

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

4 6 10 Specify the infrastructure ID and node label.

1.4.3. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

   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

   ....

   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
   ```

```
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.

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

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

For example:

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

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

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

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

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

NOTE

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

Next steps

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

CHAPTER 2. MANUALLY SCALING A MACHINE SET

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

NOTE

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

2.1. PREREQUISITES

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

IMPORTANT

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

2.2. SCALING A MACHINE SET MANUALLY

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

Prerequisites

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

Procedure

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

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

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

2. Scale the machine set:

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

   Or:

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

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

2.3. THE MACHINE SET DELETION POLICY

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

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

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

IMPORTANT

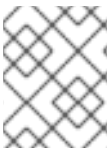
By default, the OpenShift Container Platform router pods are deployed on workers. Because the router is required to access some cluster resources, including the web console, do not scale the worker machine set to **0** unless you first relocate the router pods.

NOTE

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

CHAPTER 3. MODIFYING A MACHINE SET

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



NOTE

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

3.1. MODIFYING A MACHINE SET

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

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

NOTE

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

Prerequisites

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

Procedure

1. Edit the machine set:

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

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

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

   Or:

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

   Wait for the machines to be removed.

3. Scale up the machine set as needed:

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

Or:

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

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

CHAPTER 4. DELETING A MACHINE

You can delete a specific machine.

4.1. DELETING A SPECIFIC MACHINE

You can delete a specific machine.

Prerequisites

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

Procedure

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

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

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

2. Delete the machine:

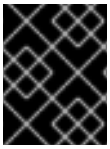
```
$ oc delete machine <machine> -n openshift-machine-api
```

IMPORTANT

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

CHAPTER 5. APPLYING AUTOSCALING TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

Applying autoscaling to an OpenShift Container Platform cluster involves deploying a cluster autoscaler and then deploying machine autoscalers for each machine type in your cluster.



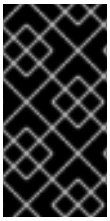
IMPORTANT

You can configure the cluster autoscaler only in clusters where the machine API is operational.

5.1. ABOUT THE CLUSTER AUTOSCALER

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

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



IMPORTANT

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

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

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

- Pods with restrictive pod disruption budgets (PDBs).
- Kube-system pods that do not run on the node by default.
- Kube-system pods that do not have a PDB or have a PDB that is too restrictive.
- Pods that are not backed by a controller object such as a deployment, replica set, or stateful set.
- Pods with local storage.
- Pods that cannot be moved elsewhere because of a lack of resources, incompatible node selectors or affinity, matching anti-affinity, and so on.
- Unless they also have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** annotation, pods that have a **"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** annotation.

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

- Do not modify the nodes that are in autoscaled node groups directly. All nodes within the same node group have the same capacity and labels and run the same system pods.
- Specify requests for your pods.
- If you have to prevent pods from being deleted too quickly, configure appropriate PDBs.
- Confirm that your cloud provider quota is large enough to support the maximum node pools that you configure.
- Do not run additional node group autoscalers, especially the ones offered by your cloud provider.

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

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

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

5.2. ABOUT THE MACHINE AUTOSCALER

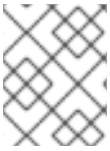
The machine autoscaler adjusts the number of Machines in the machine sets that you deploy in an OpenShift Container Platform cluster. You can scale both the default **worker** machine set and any other machine sets that you create. The machine autoscaler makes more Machines when the cluster runs out of resources to support more deployments. Any changes to the values in **MachineAutoscaler** resources, such as the minimum or maximum number of instances, are immediately applied to the machine set they target.

IMPORTANT

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

5.3. CONFIGURING THE CLUSTER AUTOSCALER

First, deploy the cluster autoscaler to manage automatic resource scaling in your OpenShift Container Platform cluster.



NOTE

Because the cluster autoscaler is scoped to the entire cluster, you can make only one cluster autoscaler for the cluster.

5.3.1. ClusterAutoscaler resource definition

This **ClusterAutoscaler** resource definition shows the parameters and sample values for the cluster autoscaler.

```
apiVersion: "autoscaling.openshift.io/v1"
kind: "ClusterAutoscaler"
metadata:
  name: "default"
spec:
  podPriorityThreshold: -10 (1)
  resourceLimits:
    maxNodesTotal: 24 (2)
    cores:
      min: 8 (3)
      max: 128 (4)
    memory:
      min: 4 (5)
      max: 256 (6)
    gpus:
      - type: nvidia.com/gpu (7)
        min: 0 (8)
        max: 16 (9)
      - type: amd.com/gpu (10)
        min: 0 (11)
        max: 4 (12)
  scaleDown: (13)
    enabled: true (14)
    delayAfterAdd: 10m (15)
    delayAfterDelete: 5m (16)
    delayAfterFailure: 30s (17)
    unneededTime: 60s (18)
```

(1) Specify the priority that a pod must exceed to cause the cluster autoscaler to deploy additional nodes. Enter a 32-bit integer value. The **podPriorityThreshold** value is compared to the value of the **PriorityClass** that you assign to each pod.

(2) Specify the maximum number of nodes to deploy. This value is the total number of machines that are deployed in your cluster, not just the ones that the autoscaler controls. Ensure that this value is large enough to account for all of your control plane and compute machines and the total number of replicas that you specify in your **MachineAutoscaler** resources.

(3) Specify the minimum number of cores to deploy.

(4) Specify the maximum number of cores to deploy.

(5) Specify the minimum amount of memory, in GiB, per node.

6 Specify the maximum amount of memory, in GiB, per node.

7 10 Optionally, specify the type of GPU node to deploy. Only **nvidia.com/gpu** and **amd.com/gpu** are valid types.

8 11 Specify the minimum number of GPUs to deploy.

9 12 Specify the maximum number of GPUs to deploy.

13 In this section, you can specify the period to wait for each action by using any valid ParseDuration interval, including **ns**, **us**, **ms**, **s**, **m**, and **h**.

14 Specify whether the cluster autoscaler can remove unnecessary nodes.

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

16 Specify the period to wait before deleting a node after a node has recently been *deleted*. If you do not specify a value, the default value of **10s** is used.

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

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

5.3.2. Deploying the cluster autoscaler

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

Procedure

1. Create a YAML file for the **ClusterAutoscaler** resource that contains the customized resource definition.
2. Create the resource in the cluster:

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

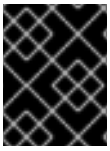
1 **<filename>** is the name of the resource file that you customized.

5.4. NEXT STEPS

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

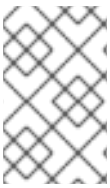
5.5. CONFIGURING THE MACHINE AUTOSCALERS

After you deploy the cluster autoscaler, deploy **MachineAutoscaler** resources that reference the machine sets that are used to scale the cluster.



IMPORTANT

You must deploy at least one **MachineAutoscaler** resource after you deploy the **ClusterAutoscaler** resource.



NOTE

You must configure separate resources for each machine set. Remember that machine sets are different in each region, so consider whether you want to enable machine scaling in multiple regions. The machine set that you scale must have at least one machine in it.

5.5.1. MachineAutoscaler resource definition

This **MachineAutoscaler** resource definition shows the parameters and sample values for the machine autoscaler.

```
apiVersion: "autoscaling.openshift.io/v1beta1"
kind: "MachineAutoscaler"
metadata:
  name: "worker-us-east-1a" (1)
  namespace: "openshift-machine-api"
spec:
  minReplicas: 1 (2)
  maxReplicas: 12 (3)
  scaleTargetRef: (4)
    apiVersion: machine.openshift.io/v1beta1
    kind: MachineSet (5)
    name: worker-us-east-1a (6)
```

(1) Specify the machine autoscaler name. To make it easier to identify which machine set this machine autoscaler scales, specify or include the name of the machine set to scale. The machine set name takes the following form: **<clusterid>-<machineset>-<aws-region-az>**

(2) Specify the minimum number machines of the specified type that must remain in the specified zone after the cluster autoscaler initiates cluster scaling. Do not set this value to **0**.

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

(4) In this section, provide values that describe the existing machine set to scale.

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

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

5.5.2. Deploying the machine autoscaler

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

Procedure

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

2. Create the resource in the cluster:

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

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

5.6. ADDITIONAL RESOURCES

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

CHAPTER 6. CREATING INFRASTRUCTURE MACHINE SETS

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

IMPORTANT

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

6.1. OPENSHIFT CONTAINER PLATFORM INFRASTRUCTURE COMPONENTS

The following OpenShift Container Platform components are infrastructure components:

- Kubernetes and OpenShift Container Platform control plane services that run on masters
- The default router
- The container image registry
- The cluster metrics collection, or monitoring service
- Cluster aggregated logging
- Service brokers

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

6.2. CREATING INFRASTRUCTURE MACHINE SETS FOR PRODUCTION ENVIRONMENTS

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

6.2.1. Creating machine sets for different clouds

Use the sample machine set for your cloud.

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

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

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

-

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructureID> 1
  name: <infrastructureID>-<role>-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-<role>-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructureID>-<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: <infrastructureID>-worker-profile 11
          instanceType: m4.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: us-east-1a
            region: us-east-1
          securityGroups:
            - filters:
                - name: tag:Name
                  values:
                    - <infrastructureID>-worker-sg 12
          subnet:
            filters:
              - name: tag:Name
                values:
                  - <infrastructureID>-private-us-east-1a 13
```

```
        tags:
          - name: kubernetes.io/cluster/<infrastructureID> 14
            value: owned
        userDataSecret:
          name: worker-user-data
```

1 3 5 11 12 13 14 Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

```
$ oc 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.

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

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

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

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

```
      labels:
        node-role.kubernetes.io/<role>: "" (11)
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image:
          offer: ""
          publisher: ""
          resourceID: /resourceGroups/<infrastructureID>-
rg/providers/Microsoft.Compute/images/<infrastructureID>
          sku: ""
          version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: centralus
        managedIdentity: <infrastructureID>-identity (12)
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructureID>-rg (13)
        sshPrivateKey: ""
        sshPublicKey: ""
        subnet: <infrastructureID>-<role>-subnet (14) (15)
        userDataSecret:
          name: <role>-user-data (16)
        vmSize: Standard_D2s_v3
        vnet: <infrastructureID>-vnet (17)
        zone: "1" (18)
```

(1) (5) (7) (12) (13) (14) (17) Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

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

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

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

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

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

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

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

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructureID> 1
  name: <infrastructureID>-w-a 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructureID> 3
      machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructureID> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructureID>-w-a 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 9
      providerSpec:
        value:
          apiVersion: gcpprovider.openshift.io/v1beta1
          canIPForward: false
          credentialsSecret:
            name: gcp-cloud-credentials
          deletionProtection: false
          disks:
          - autoDelete: true
            boot: true
            image: <infrastructureID>-rhcos-image 10
            labels: null
            sizeGb: 128
            type: pd-ssd
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructureID>-network 11
            subnetwork: <infrastructureID>-<role>-subnet 12
```

```
projectID: <project_name> 13
region: us-central1
serviceAccounts:
- email: <infrastructureID>-w@<project_name>.iam.gserviceaccount.com 14 15
  scopes:
  - https://www.googleapis.com/auth/cloud-platform
tags:
- <infrastructureID>-<role> 16
userDataSecret:
  name: worker-user-data
zone: us-central1-a
```

1 2 3 4 5 8 10 11 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
```

12 16 Specify the infrastructure ID and node label.

6 7 9 Specify the node label to add.

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

6.2.2. Creating a machine set

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

Prerequisites

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

Procedure

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

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

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

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

....

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


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.

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

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

For example:

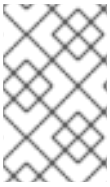
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api
```

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

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

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

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



NOTE

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

Next steps

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

6.3. ASSIGNING MACHINE SET RESOURCES TO INFRASTRUCTURE NODES

After creating an infrastructure machine set, the **worker** and **infra** roles are applied to new infra nodes. Nodes with the **infra** role applied are not counted toward the total number of subscriptions that are required to run the environment, even when the **worker** role is also applied.

However, with an infra node being assigned as a worker, there is a chance user workloads could get inadvertently assigned to an infra node. To avoid this, you can apply a taint to the infra node and tolerations for the pods you want to control.

6.3.1. Binding infrastructure node workloads using taints and tolerations

If you have an infra node that has the **infra** and **worker** roles assigned, you must configure the node so that user workloads are not assigned to it.

Prerequisites

- Configure additional **MachineSet** objects in your OpenShift Container Platform cluster.

Procedure

1. Use the following command to add a taint to the infra node to prevent scheduling user workloads on it:

```
$ oc adm taint nodes <node_name> <key>:<effect>
```

For example:

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

This example places a taint on **node1** that has key **node-role.kubernetes.io/infra** and taint effect **NoSchedule**. Nodes with the **NoSchedule** effect schedule only pods that tolerate the taint, but allow existing pods to remain scheduled on the node.

NOTE

If a descheduler is used, pods violating node taints could be evicted from the cluster.

2. Add tolerations for the pod configurations you want to schedule on the infra node, like router, registry, and monitoring workloads. Add the following code to the **Pod** object specification:

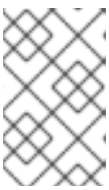
```
tolerations:
  - effect: NoSchedule (1)
    key: node-role.kubernetes.io/infra (2)
    operator: Exists (3)
```

(1) Specify the effect that you added to the node.

(2) Specify the key that you added to the node.

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

This toleration matches the taint created by the **oc adm taint** command. A pod with this toleration can be scheduled onto the infra node.



NOTE

Moving pods for an Operator installed via OLM to an infra node is not always possible. The capability to move Operator pods depends on the configuration of each Operator.

3. Schedule the pod to the infra node using a scheduler. See the documentation for *Controlling pod placement onto nodes* for details.

Additional resources

- See Controlling pod placement using the scheduler for general information on scheduling a pod to a node.
- See Moving resources to infrastructure machine sets for instructions on scheduling pods to infra nodes.

6.4. MOVING RESOURCES TO INFRASTRUCTURE MACHINE SETS

Some of the infrastructure resources are deployed in your cluster by default. You can move them to the infrastructure machine sets that you created.

6.4.1. Moving the router

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

Prerequisites

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

Procedure

1. View the **IngressController** custom resource for the router Operator:

```
$ oc get ingresscontroller default -n openshift-ingress-operator -o yaml
```

The command output resembles the following text:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: 2019-04-18T12:35:39Z
  finalizers:
  - ingresscontroller.operator.openshift.io/finalizer-ingresscontroller
  generation: 1
  name: default
  namespace: openshift-ingress-operator
  resourceVersion: "11341"
  selfLink: /apis/operator.openshift.io/v1/namespaces/openshift-ingress-
operator/ingresscontrollers/default
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a
spec: {}
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: 2019-04-18T12:36:15Z
    status: "True"
    type: Available
  domain: apps.<cluster>.example.com
```

```
  endpointPublishingStrategy:
    type: LoadBalancerService
  selector: ingresscontroller.operator.openshift.io/deployment-ingresscontroller=default
```

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

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

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

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

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

   a. View the list of router pods and note the node name of the running pod:

```
$ oc get pod -n openshift-ingress -o wide

NAME                              READY   STATUS        RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
router-default-86798b4b5d-bdlvd   1/1     Running       0          28s     10.130.2.4   ip-10-
0-217-226.ec2.internal   <none>           <none>
router-default-955d875f4-255g8    0/1     Terminating   0          19h     10.129.2.4   ip-10-
0-148-172.ec2.internal   <none>           <none>
```

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

   b. View the node status of the running pod:

```
$ oc get node <node_name> 1

NAME                           STATUS  ROLES        AGE   VERSION
ip-10-0-217-226.ec2.internal   Ready   infra,worker  17h   v1.17.1
```

1 Specify the **<node_name>** that you obtained from the pod list.

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

6.4.2. Moving the default registry

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

Prerequisites

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

Procedure

1. View the **config/instance** object:

```
$ oc get config/cluster -o yaml
```

The output resembles the following text:

```
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 config/cluster
```

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

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

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

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

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

   b. Confirm the node has the label you specified:

   ```
   $ oc describe node <node_name>
   ```

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

6.4.3. Moving the monitoring solution

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

Procedure

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

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

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

2. Apply the new config map:

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

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

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

4. If a component has not moved to the **infra** node, delete the pod with this component:

```
$ oc delete pod -n openshift-monitoring <pod>
```

The component from the deleted pod is re-created on the **infra** node.

Additional resources

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

6.4.4. Moving the cluster logging resources

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

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

NOTE

You should set your machine set to use at least 6 replicas.

Prerequisites

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

Procedure

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

```
$ oc edit ClusterLogging instance
```

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

....

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  curation:
    curator:
      nodeSelector: 1
          node-role.kubernetes.io/infra: ''
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
```

```
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: ''
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: 3
        node-role.kubernetes.io/infra: '' 4
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

....
```

1 2 3 4 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 steps

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
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
NAME                                         STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
```

```
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker       51m   v1.17.1
ip-10-0-152-241.us-east-2.compute.internal   Ready    master       60m   v1.17.1
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra        51m   v1.17.1
```

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

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: '' 2
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 2 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
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
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
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
```

CHAPTER 7. USER-PROVISIONED INFRASTRUCTURE

7.1. ADDING RHEL COMPUTE MACHINES TO AN OPENSHIFT CONTAINER PLATFORM CLUSTER

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

7.1.1. About adding RHEL compute nodes to a cluster

In OpenShift Container Platform 4.4, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines, which are also known as worker machines, in your cluster if you use a user-provisioned infrastructure installation. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane, or master, machines in your cluster.

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

IMPORTANT

Because removing OpenShift Container Platform from a machine in the cluster requires destroying the operating system, you must use dedicated hardware for any RHEL machines that you add to the cluster.

IMPORTANT

Swap memory is disabled on all RHEL machines that you add to your OpenShift Container Platform cluster. You cannot enable swap memory on these machines.

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

7.1.2. System requirements for RHEL compute nodes

The Red Hat Enterprise Linux (RHEL) compute machine hosts, which are also known as worker machine hosts, in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements.

- You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.
- Production environments must provide compute machines to support your expected workloads. As a cluster administrator, you must calculate the expected workload and add about 10 percent for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.
- Each system must meet the following hardware requirements:
    - Physical or virtual system, or an instance running on a public or private IaaS.
    - Base OS: RHEL 7.6-7.8 with "Minimal" installation option.

IMPORTANT

Only RHEL 7.6-7.8 is supported in OpenShift Container Platform 4.4. You must not upgrade your compute machines to RHEL 8.

  - If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See Enabling FIPS Mode in the RHEL 7 documentation.
  - NetworkManager 1.0 or later.
  - 1 vCPU.
  - Minimum 8 GB RAM.
  - Minimum 15 GB hard disk space for the file system containing /**var**/.
  - Minimum 1 GB hard disk space for the file system containing /**usr**/**local**/**bin**/.
  - Minimum 1 GB hard disk space for the file system containing the system's temporary directory. The system's temporary directory is determined according to the rules defined in the tempfile module in Python's standard library.

- Each system must meet any additional requirements for your system provider. For example, if you installed your cluster on VMware vSphere, your disks must be configured according to its storage guidelines and the **disk.enableUUID=true** attribute must be set.
- Each system must be able to access the cluster's API endpoints by using DNS-resolvable host names. Any network security access control that is in place must allow the system access to the cluster's API service endpoints.

7.1.2.1. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

7.1.3. Preparing the machine to run the playbook

Before you can add compute machines that use Red Hat Enterprise Linux as the operating system to an OpenShift Container Platform 4.4 cluster, you must prepare a machine to run the playbook from. This machine is not part of the cluster but must be able to access it.

Prerequisites

- Install the OpenShift CLI (**oc**) on the machine that you run the playbook on.
- Log in as a user with **cluster-admin** permission.

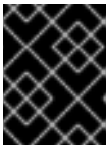
Procedure

1. For each of the **inventory** file for the cluster and the installation

1. Ensure that the **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the machine. One way to accomplish this is to use the same machine that you used to install the cluster.

2. Configure the machine to access all of the RHEL hosts that you plan to use as compute machines. You can use any method that your company allows, including a bastion with an SSH proxy or a VPN.

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



IMPORTANT

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

4. If you have not already done so, register the machine with RHSM and attach a pool with an **OpenShift** subscription to it:

   a. Register the machine with RHSM:

   ```
   # subscription-manager register --username=<user_name> --password=<password>
   ```

   b. Pull the latest subscription data from RHSM:

   ```
   # subscription-manager refresh
   ```

   c. List the available subscriptions:

   ```
   # subscription-manager list --available --matches '*OpenShift*'
   ```

   d. In the output for the previous command, find the pool ID for an OpenShift Container Platform subscription and attach it:

   ```
   # subscription-manager attach --pool=<pool_id>
   ```

5. Enable the repositories required by OpenShift Container Platform 4.4:

```
# 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-4.4-rpms"
```

6. Install the required packages, including **openshift-ansible**:

```
# yum install openshift-ansible openshift-clients jq
```

The **openshift-ansible** package provides installation program utilities and pulls in other packages that you require to add a RHEL compute node to your cluster, such as Ansible, playbooks, and related configuration files. The **openshift-clients** provides the **oc** CLI, and the **jq** package improves the display of JSON output on your command line.

7.1.4. 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

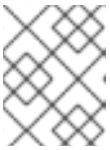
6. Enable only the repositories required by OpenShift Container Platform 4.4:

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.4-rpms"
```

7. Stop and disable firewalld on the host:

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

-



NOTE

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

7.1.5. 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 4.4 cluster.

Prerequisites

- You installed the required packages and performed the necessary configuration on the machine that you run the playbook on.
- You prepared the RHEL hosts for installation.

Procedure

Perform the following steps on the machine that you prepared to run the playbook:

1. Create an Ansible inventory file that is named **/<path>/inventory/hosts** that defines your compute machine hosts and required variables:

```
[all:vars]
ansible_user=root 1
#ansible_become=True 2

openshift_kubeconfig_path="~/.kube/config" 3

[new_workers] 4
mycluster-rhel7-0.example.com
mycluster-rhel7-1.example.com
```

1. Specify the user name that runs the Ansible tasks on the remote compute machines.
2. If you do not specify **root** for the **ansible_user**, you must set **ansible_become** to **True** and assign the user sudo permissions.
3. Specify the path and file name of the **kubeconfig** file for your cluster.
4. List each RHEL machine to add to your cluster. You must provide the fully-qualified domain name for each host. This name is the host name that the cluster uses to access the machine, so set the correct public or private name to access the machine.

2. Run the playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml 1
```

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

7.1.6. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

   The output lists all of the machines that you created.

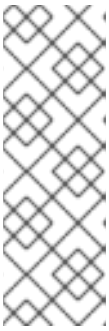
2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

   - 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
```

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

```
$ oc get csr
```

Example output

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

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

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

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

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

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

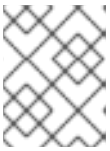
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



NOTE

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

Additional information

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

7.1.7. Required parameters for the Ansible hosts file

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

Paramter	Description	Values
ansible_user	The SSH user that allows SSH-based authentication without requiring a password. If you use SSH key-based authentication, then you must manage the key with an SSH agent.	A user name on the system. The default value is **root**.
ansible_become	If the values of **ansible_user** is not root, you must set **ansible_become** to **True**, and the user that you specify as the **ansible_user** must be configured for passwordless sudo access.	**True**. If the value is not **True**, do not specify and define this parameter.
openshift_kubeconfig_path	Specifies a path and file name to a local directory that contains the **kubeconfig** file for your cluster.	The path and name of the configuration file.

7.1.7.1. Optional: Removing RHCOS compute machines from a cluster

After you add the Red Hat Enterprise Linux (RHEL) compute machines to your cluster, you can optionally remove the Red Hat Enterprise Linux CoreOS (RHCOS) compute machines to free up resources.

Prerequisites

- You have added RHEL compute machines to your cluster.

Procedure

1. View the list of machines and record the node names of the RHCOS compute machines:

```
$ oc get nodes -o wide
```

2. For each RHCOS compute machine, delete the node:

   a. Mark the node as unschedulable by running the **oc adm cordon** command:

```
$ oc adm cordon <node_name> 1
```

1 Specify the node name of one of the RHCOS compute machines.

b. Drain all the pods from the node:

```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets 1
```

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

c. Delete the node:

```
$ oc delete nodes <node_name> 1
```

1 Specify the node name of the RHCOS compute machine that you drained.

3. Review the list of compute machines to ensure that only the RHEL nodes remain:

```
$ oc get nodes -o wide
```

4. Remove the RHCOS machines from the load balancer for your cluster's compute machines. You can delete the virtual machines or reimage the physical hardware for the RHCOS compute machines.

7.2. 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.

7.2.1. About adding RHEL compute nodes to a cluster

In OpenShift Container Platform 4.4, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines, which are also known as worker machines, in your cluster if you use a user-provisioned infrastructure installation. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane, or master, machines in your cluster.

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

IMPORTANT

Because removing OpenShift Container Platform from a machine in the cluster requires destroying the operating system, you must use dedicated hardware for any RHEL machines that you add to the cluster.

IMPORTANT

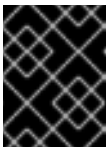
Swap memory is disabled on all RHEL machines that you add to your OpenShift Container Platform cluster. You cannot enable swap memory on these machines.

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

7.2.2. System requirements for RHEL compute nodes

The Red Hat Enterprise Linux (RHEL) compute machine hosts, which are also known as worker machine hosts, in your OpenShift Container Platform environment must meet the following minimum hardware specifications and system-level requirements.

- You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.
- Production environments must provide compute machines to support your expected workloads. As a cluster administrator, you must calculate the expected workload and add about 10 percent for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.
- Each system must meet the following hardware requirements:
  - Physical or virtual system, or an instance running on a public or private IaaS.
  - Base OS: RHEL 7.6-7.8 with "Minimal" installation option.

    IMPORTANT

    Only RHEL 7.6-7.8 is supported in OpenShift Container Platform 4.4. You must not upgrade your compute machines to RHEL 8.

  - If you deployed OpenShift Container Platform in FIPS mode, you must enable FIPS on the RHEL machine before you boot it. See Enabling FIPS Mode in the RHEL 7 documentation.
  - NetworkManager 1.0 or later.
  - 1 vCPU.
  - Minimum 8 GB RAM.
  - Minimum 15 GB hard disk space for the file system containing **/var/**.
  - Minimum 1 GB hard disk space for the file system containing **/usr/local/bin/**.
  - Minimum 1 GB hard disk space for the file system containing the system's temporary directory. The system's temporary directory is determined according to the rules defined in the tempfile module in Python's standard library.
- Each system must meet any additional requirements for your system provider. For example, if you installed your cluster on VMware vSphere, your disks must be configured according to its storage guidelines and the **disk.enableUUID=true** attribute must be set.
- Each system must be able to access the cluster's API endpoints by using DNS-resolvable host names. Any network security access control that is in place must allow the system access to the cluster's API service endpoints.

7.2.2.1. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

7.2.3. 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 \*
```

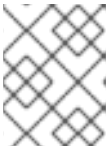
   Note that this might take a few minutes if you have a large number of available repositories

6. Enable only the repositories required by OpenShift Container Platform 4.4:

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.4-rpms"
```

7. Stop and disable firewalld on the host:

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



NOTE

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

7.2.4. 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 4.4 cluster.

Prerequisites

- Your OpenShift Container Platform cluster already contains RHEL compute nodes.
- The **hosts** file that you used to add the first RHEL compute machines to your cluster is on the machine that you use the run the playbook.
- The machine that you run the playbook on must be able to access all of the RHEL hosts. You can use any method that your company allows, including a bastion with an SSH proxy or a VPN.
- The **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the machine that you use the run the playbook.
- You must prepare the RHEL hosts for installation.
- Configure a user on the machine that you run the playbook on that has SSH access to all of the RHEL hosts.
- If you use SSH key-based authentication, you must manage the key with an SSH agent.
- Install the OpenShift CLI (**oc**) on the machine that you run the playbook on.

Procedure

1. Open the Ansible inventory file at **/<path>/inventory/hosts** that defines your compute machine hosts and required variables.
2. Rename the **[new_workers]** section of the file to **[workers]**.
3. Add a **[new_workers]** section to the file and define the fully-qualified domain names for each new host. The file resembles the following example:

```
[all:vars]
ansible_user=root
```

```
#ansible_become=True

openshift_kubeconfig_path="~/.kube/config"

[workers]
mycluster-rhel7-0.example.com
mycluster-rhel7-1.example.com

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

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

4. Run the scaleup playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml 1
```

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

7.2.5. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

The output lists all of the machines that you created.

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

```
$ oc get csr

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- 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
```

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

```
$ oc get csr
```

Example output

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

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

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

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

-

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

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

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

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

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

NOTE

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

Additional information

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

7.2.6. Required parameters for the Ansible hosts file

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

Paramter	Description	Values
ansible_user	The SSH user that allows SSH-based authentication without requiring a password. If you use SSH key-based authentication, then you must manage the key with an SSH agent.	A user name on the system. The default value is **root**.
ansible_become	If the values of **ansible_user** is not root, you must set **ansible_become** to **True**, and the user that you specify as the **ansible_user** must be configured for passwordless sudo access.	**True**. If the value is not **True**, do not specify and define this parameter.

Paramter	Description	Values
openshift_kubeconfig_path	Specifies a path and file name to a local directory that contains the **kubeconfig** file for your cluster.	The path and name of the configuration file.

7.3. ADDING COMPUTE MACHINES TO VSPHERE

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

7.3.1. Prerequisites

- You installed a cluster on vSphere .

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

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

Prerequisites

- Obtain the base64-encoded Ignition file for your compute machines.
- You have access to the vSphere template that you created for your cluster.

Procedure

1. After the template deploys, deploy a VM for a machine in the cluster.
    a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
    b. On the **Select a name and folder**tab, specify a name for the VM. You might include the machine type in the name, such as **compute-1**.
    c. On the **Select a name and folder**tab, select the name of the folder that you created for the cluster.
    d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
    e. Optional: On the **Select storage** tab, customize the storage options.
    f. On the **Select clone options**, select **Customize this virtual machine's hardware**.
    g. On the **Customize hardware** tab, click **VM Options → Advanced**.
        - From the **Latency Sensitivity** list, select **High**.
        - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:

- **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded compute Ignition config file for this machine type.
- **guestinfo.ignition.config.data.encoding**: Specify **base64**.
- **disk.EnableUUID**: Specify **TRUE**.

h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type. Also, make sure to select the correct network under **Add network adapter** if there are multiple networks available.

i. Complete the configuration and power on the VM.

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

7.3.3. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

The output lists all of the machines that you created.

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

```
$ oc get csr

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- 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
```

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

```
$ oc get csr
```

Example output

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

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

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

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

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

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

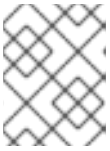
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



NOTE

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

Additional information

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

7.4. ADDING COMPUTE MACHINES TO BARE METAL

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

7.4.1. 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.

7.4.2. 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. Follow either the steps to use an ISO image or network PXE booting to create the machines.

7.4.2.1. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

You can create more compute machines for your bare metal cluster by using an ISO image to create the machines.

Prerequisites

- Obtain the URL of the Ignition config file for the compute machines for your cluster. You uploaded this file to your HTTP server during installation.

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

Procedure

1. Use the ISO file to install RHCOS on more compute machines. Use the same method that you used when you created machines before you installed the cluster:
   - Burn the ISO image to a disk and boot it directly.
   - Use ISO redirection with a LOM interface.
2. After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
3. Add the parameters to the kernel command line:

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

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

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

   (3) Specify the URL of the compute Ignition config file.

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

7.4.2.2. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

You can create more compute machines for your bare metal cluster by using PXE or iPXE booting.

Prerequisites

- Obtain the URL of the Ignition config file for the compute machines for your cluster. You uploaded this file to your HTTP server during installation.
- Obtain the URLs of the RHCOS ISO image, compressed metal BIOS, **kernel**, and **initramfs** files that you uploaded to your HTTP server during cluster installation.
- You have access to the PXE booting infrastructure that you used to create the machines for your OpenShift Container Platform cluster during installation. The machines must boot from their local disks after RHCOS is installed on them.
- If you use UEFI, you have access to the **grub.conf** file that you modified during OpenShift Container Platform installation.

Procedure

1. Confirm that your PXE or iPXE installation for the RHCOS images is correct.

- For PXE:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-
initramfs.<architecture>.img console=tty0 console=ttyS0 coreos.inst=yes
coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-
<version>-metal.<architecture>.raw.gz
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign (2) (3)
```

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

(2) If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

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

- For iPXE:

```
kernel http://<HTTP_server>/rhcos-<version>-installer-kernel-<architecture> ip=dhcp
rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-initramfs.
<architecture>.img console=tty0 console=ttyS0 coreos.inst=yes
coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-
<version>-metal.<arhcitectutre>.raw.gz
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign (1) (2)
initrd http://<HTTP_server>/rhcos-<version>-installer-initramfs.<architecture>.img (3)
boot
```

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

(2) If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

(3) Specify the location of the **initramfs** file that you uploaded to your HTTP server.

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

7.4.3. 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

NAME                 STATUS   ROLES    AGE   VERSION
master-01.example.com   Ready    master   40d   v1.17.1
master-02.example.com   Ready    master   40d   v1.17.1
master-03.example.com   Ready    master   40d   v1.17.1
worker-01.example.com   Ready    worker   40d   v1.17.1
worker-02.example.com   Ready    worker   40d   v1.17.1
```

   The output lists all of the machines that you created.

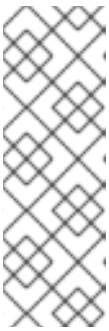
2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

   - 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
```

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

```
$ oc get csr
```

Example output

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

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

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

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

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

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

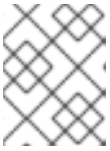
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



NOTE

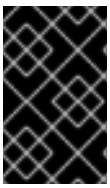
It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

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

CHAPTER 8. DEPLOYING MACHINE HEALTH CHECKS

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



IMPORTANT

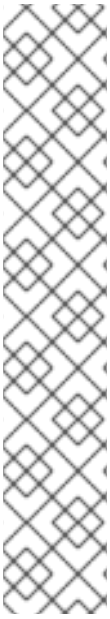
This process is not applicable to clusters where you manually provisioned the machines yourself. You can use the advanced machine management and scaling capabilities only in clusters where the machine API is operational.

8.1. ABOUT MACHINE HEALTH CHECKS

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

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

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



NOTE

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

Example output

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

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

To limit the disruptive impact of machine deletions, the controller drains and deletes only one node at a time. If there are more unhealthy machines than the **maxUnhealthy** threshold allows for in the targeted pool of machines, the controller stops deleting machines and you must manually intervene.

To stop the check, remove the custom resource.

8.1.1. 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

- See Short-circuiting machine health check remediation for more information about short circuiting

8.2. SAMPLE MACHINEHEALTHCHECK RESOURCE

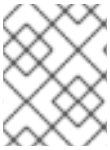
The **MachineHealthCheck** resource resembles the following YAML file:

MachineHealthCheck

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example 1
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> 2
      machine.openshift.io/cluster-api-machine-type: <role> 3
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 4
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s" 5
    status: "False"
  - type:    "Ready"
    timeout: "300s" 6
    status: "Unknown"
  maxUnhealthy: "40%" 7
  nodeStartupTimeout: "10m" 8
```

1. Specify the name of the machine health check to deploy.
2. 3. Specify a label for the machine pool that you want to check.
4. Specify the machine set to track in **<cluster_name>-<label>-<zone>** format. For example, **prod-node-us-east-1a**.
5. 6. Specify the timeout duration for a node condition. If a condition is met for the duration of the timeout, the machine will be remediated. Long timeouts can result in long periods of downtime for a workload on an unhealthy machine.
7. Specify the amount of unhealthy machines allowed in the targeted pool. This can be set as a percentage or an integer.

8 Specify the timeout duration that a machine health check must wait for a node to join the cluster before a machine is determined to be unhealthy.



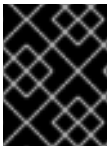
NOTE

The **matchLabels** are examples only; you must map your machine groups based on your specific needs.

8.2.1. Short-circuiting machine health check remediation

Short circuiting ensures that machine health checks remediate machines only when the cluster is healthy. Short-circuiting is configured through the **maxUnhealthy** field in the **MachineHealthCheck** resource.

If the user defines a value for the **maxUnhealthy** field, before remediating any machines, the **MachineHealthCheck** compares the value of **maxUnhealthy** with the number of machines within its target pool that it has determined to be unhealthy. Remediation is not performed if the number of unhealthy machines exceeds the **maxUnhealthy** limit.



IMPORTANT

If **maxUnhealthy** is not set, the value defaults to **100%** and the machines are remediated regardless of the state of the cluster.

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

8.2.1.1. 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.

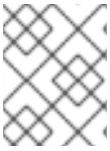
8.2.1.2. Setting maxUnhealthy by using percentages

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.

8.3. 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
```