OpenShift Container Platform 4.2

Machine management

Adding and maintaining cluster machines

Last Updated: 2020-06-15

OpenShift Container Platform 4.2 Machine management

Adding and maintaining cluster machines

Abstract

This document provides instructions for managing the machines that make up an OpenShift Container Platform 4.2 cluster. Some tasks make use of the enhanced automatic machine management functions of an OpenShift Container Platform 4.2 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 MACHINESETS

1.1. CREATING A MACHINESET IN AWS

You can create a different MachineSet to serve a specific purpose in your OpenShift Container Platform cluster on Amazon Web Services (AWS). For example, you might create infrastructure MachineSets 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 4.2 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.2 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, 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.

MachineSets

Groups of machines. MachineSets are to machines as ReplicaSets are to Pods. If you need more machines or must scale them down, you change the **replicas** field on the MachineSet to meet your compute need.

The following custom resources add more capabilities to your cluster:

MachineAutoscaler

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

ClusterAutoscaler

This resource is based on the upstream ClusterAutoscaler project. In the OpenShift Container Platform implementation, it is integrated with the Machine API by extending the MachineSet 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 ScalingPolicy so you can scale up nodes but not scale them down.

MachineHealthCheck

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

> **NOTE**
>
> In version 4.2, MachineHealthChecks is a Technology Preview feature

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 MachineSet is scoped to a single zone, so the installation program sends out MachineSets 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 MachineSet Custom Resource on AWS

This sample YAML defines a MachineSet 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 and **jq** package 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 MachineSet

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

Prerequisites

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

Procedure

1. Create a new YAML file that contains the MachineSet Custom Resource 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 MachineSet 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 MachineSet:

```
$ 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**:

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

3. View the list of MachineSets:

```
$ 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 MachineSet is available, the **DESIRED** and **CURRENT** values match. If the MachineSet is not available, wait a few minutes and run the command again.

4. After the new MachineSet 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 MachineSet is not applied to existing machines owned by the MachineSet. For example, labels edited or added to an existing MachineSet are not propagated to existing machines and Nodes associated with the MachineSet.

Next steps

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

1.2. CREATING A MACHINESET IN AZURE

You can create a different MachineSet to serve a specific purpose in your OpenShift Container Platform cluster on Microsoft Azure. For example, you might create infrastructure MachineSets 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 4.2 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.2 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, 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.

MachineSets

Groups of machines. MachineSets are to machines as ReplicaSets are to Pods. If you need more machines or must scale them down, you change the **replicas** field on the MachineSet to meet your compute need.

The following custom resources add more capabilities to your cluster:

MachineAutoscaler

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

ClusterAutoscaler

This resource is based on the upstream ClusterAutoscaler project. In the OpenShift Container Platform implementation, it is integrated with the Machine API by extending the MachineSet 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 ScalingPolicy so you can scale up nodes but not scale them down.

MachineHealthCheck

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

NOTE

In version 4.2, MachineHealthChecks is a Technology Preview feature

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 MachineSet is scoped to a single zone, so the installation program sends out MachineSets 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 MachineSet Custom Resource on Azure

This sample YAML defines a MachineSet 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 and **jq** package 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 MachineSet

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

Prerequisites

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

Procedure

1. Create a new YAML file that contains the MachineSet Custom Resource 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 MachineSet 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 MachineSet:

   ```
   $ 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**:

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

3. View the list of MachineSets:

   ```
   $ 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 MachineSet is available, the **DESIRED** and **CURRENT** values match. If the MachineSet is not available, wait a few minutes and run the command again.

4. After the new MachineSet 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 MachineSet is not applied to existing machines owned by the MachineSet. For example, labels edited or added to an existing MachineSet are not propagated to existing machines and Nodes associated with the MachineSet.

Next steps

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

1.3. CREATING A MACHINESET IN GCP

You can create a different MachineSet to serve a specific purpose in your OpenShift Container Platform cluster on Google Cloud Platform (GCP). For example, you might create infrastructure MachineSets 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 4.2 clusters, the Machine API performs all node host provisioning management actions after the cluster installation finishes. Because of this system, OpenShift Container Platform 4.2 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, 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.

MachineSets

Groups of machines. MachineSets are to machines as ReplicaSets are to Pods. If you need more machines or must scale them down, you change the **replicas** field on the MachineSet to meet your compute need.

The following custom resources add more capabilities to your cluster:

MachineAutoscaler

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

ClusterAutoscaler

This resource is based on the upstream ClusterAutoscaler project. In the OpenShift Container Platform implementation, it is integrated with the Machine API by extending the MachineSet 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 ScalingPolicy so you can scale up nodes but not scale them down.

MachineHealthCheck

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

> **NOTE**
>
> In version 4.2, MachineHealthChecks is a Technology Preview feature

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 MachineSet is scoped to a single zone, so the installation program sends out MachineSets 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 MachineSet Custom Resource on GCP

This sample YAML defines a MachineSet 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 and **jq** package 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 MachineSet

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

Prerequisites

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

Procedure

1. Create a new YAML file that contains the MachineSet Custom Resource 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 MachineSet 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 MachineSet:

```
$ 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**:

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

3. View the list of MachineSets:

```
$ 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 MachineSet is available, the **DESIRED** and **CURRENT** values match. If the MachineSet is not available, wait a few minutes and run the command again.

4. After the new MachineSet 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 MachineSet is not applied to existing machines owned by the MachineSet. For example, labels edited or added to an existing MachineSet are not propagated to existing machines and Nodes associated with the MachineSet.

Next steps

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

CHAPTER 2. MANUALLY SCALING A MACHINESET

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

NOTE

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

Prerequisites

- If you enabled the cluster-wide proxy and scale up workers not included in **networking.machineCIDR** 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.

2.1. SCALING A MACHINESET MANUALLY

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

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 MachineSets that are in the cluster:

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

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

2. Scale the MachineSet:

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

   Or:

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

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

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 MachineSet to **0** unless you first relocate the router pods.

CHAPTER 3. MODIFYING A MACHINESET

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

NOTE

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

3.1. MODIFYING A MACHINESET

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

If you need to scale a MachineSet 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 MachineSet 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 MachineSet:

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

2. Scale down the MachineSet 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 MachineSet 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 Machineset.

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 Command-line Interface (CLI), commonly known as **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 MachineSet, 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 ClusterAutoscaler and then deploying MachineAutoscalers for each Machine type in your cluster.

IMPORTANT

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

5.1. ABOUT THE CLUSTERAUTOSCALER

The ClusterAutoscaler 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 ClusterAutoscaler has a cluster scope, and is not associated with a particular namespace.

The ClusterAutoscaler 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 ClusterAutoscaler does not increase the cluster resources beyond the limits that you specify.

IMPORTANT

Ensure that the **maxNodesTotal** value in the **ClusterAutoscaler** 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 ClusterAutoscaler 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 ClusterAutoscaler will not remove the node:

- Pods with restrictive PodDisruptionBudgets (PDBs).
- Kube-system Pods that do not run on the node by default.
- Kube-system Pods that do not have a PDBB or have a PDB that is too restrictive.
- Pods that are not backed by a controller object such as a Deployment, ReplicaSet, or StatefulSet.
- 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 ClusterAutoscaler, 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 ClusterAutoscaler modify cluster resources in different ways. The HPA changes the deployment’s or ReplicaSet’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 ClusterAutoscaler 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 ClusterAutoscaler deletes the unnecessary nodes.

The ClusterAutoscaler 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 ClusterAutoscaler ensures that the cluster has resources to run all Pods. To honor the intention of both features, the ClusterAutoscaler inclues a priority cutoff function. You can use this cutoff to schedule "best-effort" Pods, which do not cause the ClusterAutoscaler 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 MACHINEAUTOSCALER

The MachineAutoscaler adjusts the number of Machines in the MachineSets that you deploy in an OpenShift Container Platform cluster. You can scale both the default **worker** MachineSet and any other MachineSets that you create. The MachineAutoscaler 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 MachineSet they target.

IMPORTANT

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

5.3. CONFIGURING THE CLUSTERAUTOSCALER

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

NOTE

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

5.3.1. ClusterAutoscaler resource definition

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

```
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 ClusterAutoscaler 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 ClusterAutoscaler 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 ClusterAutoscaler

To deploy the ClusterAutoscaler, 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.

Next steps

- After you configure the ClusterAutoscaler, you must configure at least one MachineAutoscaler.

5.4. CONFIGURING THE MACHINEAUTOSCALERS

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

5.4.1. MachineAutoscaler resource definition

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

```
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 **MachineAutoscaler** name. To make it easier to identify which MachineSet this MachineAutoscaler scales, specify or include the name of the MachineSet to scale. The MachineSet 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 AWS zone after the ClusterAutoscaler initiates cluster scaling. Do not set this value to **0**.

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

4. In this section, provide values that describe the existing MachineSet to scale.

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

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

5.4.2. Deploying the MachineAutoscaler

To deploy the MachineAutoscaler, 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.

5.5. ADDITIONAL RESOURCES

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

CHAPTER 6. CREATING INFRASTRUCTURE MACHINESETS

You can create a MachineSet 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 MachineSet.

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 MACHINESETS FOR PRODUCTION ENVIRONMENTS

In a production deployment, deploy at least three MachineSets 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, install deploy these nodes to different availability zones. Since you need different MachineSets for each availability zone, create at least three MachineSets.

6.2.1. Creating MachineSets for different clouds

Use the sample MachineSet for your cloud.

6.2.1.1. Sample YAML for a MachineSet Custom Resource on AWS

This sample YAML defines a MachineSet 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 and **jq** package 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 MachineSet Custom Resource on Azure

This sample YAML defines a MachineSet 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 and **jq** package 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 MachineSet Custom Resource on GCP

This sample YAML defines a MachineSet 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 and **jq** package 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 MachineSet

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

Prerequisites

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

Procedure

1. Create a new YAML file that contains the MachineSet Custom Resource 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 MachineSet 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 MachineSet:

```
$ 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**:

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

3. View the list of MachineSets:

```
$ 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 MachineSet is available, the **DESIRED** and **CURRENT** values match. If the MachineSet is not available, wait a few minutes and run the command again.

4. After the new MachineSet 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 MachineSet is not applied to existing machines owned by the MachineSet. For example, labels edited or added to an existing MachineSet are not propagated to existing machines and Nodes associated with the MachineSet.

Next steps

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

6.3. MOVING RESOURCES TO INFRASTRUCTURE MACHINESETS

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

6.3.1. Moving the router

You can deploy the router Pod to a different MachineSet. By default, the Pod is displayed to a worker node.

Prerequisites

- Configure additional MachineSets 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 -o yaml
   ```

   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      v1.14.6+c4799753c
```

(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.

6.3.2. Moving the default registry

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

Prerequisites

- Configure additional MachineSets 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.

6.3.3. 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 ConfigMap.

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: ""
```

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

2. Apply the new ConfigMap:

```
$ 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.

6.3.4. 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 MachineSet 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 Cluster Logging Custom Resource 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

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   v1.16.2
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.16.2
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.16.2
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.16.2
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.16.2
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.16.2
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.16.2
```

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 Cluster Logging 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. 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. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.2, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines, which are also known as worker machines, in your cluster if you use a user-provisioned infrastructure installation. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane, or master, machines in your cluster.

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

IMPORTANT

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

IMPORTANT

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

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

7.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

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

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

IMPORTANT

Only RHEL 7.6 is supported in OpenShift Container Platform 4.2. You must not upgrade your compute machines to RHEL 8.

  - 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.

7.2.1. Certificate signing requests management

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

7.3. PREPARING 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 4.2 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 Command-line Interface (CLI), commonly known as **oc**, on the machine that you run the playbook on.
- Log in as a user with **cluster-admin** permission.

Procedure

1. Ensure that the **kubeconfig** file for the cluster and the installation program that you used to install the cluster are on the machine. One way to accomplish this is to use the same machine that you used to install the cluster.
2. Configure the machine to access all of the RHEL hosts that you plan to use as compute machines. You can use any method that your company allows, including a bastion with an SSH proxy or a VPN.

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

   IMPORTANT

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

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

   a. Register the machine with RHSM:

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

   b. Pull the latest subscription data from RHSM:

   ```
   # subscription-manager refresh
   ```

   c. List the available subscriptions:

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

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

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

5. Enable the repositories required by OpenShift Container Platform 4.2:

   ```
   # subscription-manager repos \
       --enable="rhel-7-server-rpms" \
       --enable="rhel-7-server-extras-rpms" \
       --enable="rhel-7-server-ansible-2.8-rpms" \
       --enable="rhel-7-server-ose-4.2-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.

7.4. 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 4.2:

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.2-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.

7.5. 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 4.2 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-worker-0.example.com
mycluster-worker-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 to 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.

7.6. APPROVING THE CSRS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

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-0  Ready     master  63m  v1.14.6+c4799753c
   master-1  Ready     master  63m  v1.14.6+c4799753c
   master-2  Ready     master  64m  v1.14.6+c4799753c
   worker-0  NotReady  worker  76s  v1.14.6+c4799753c
   worker-1  NotReady  worker  70s  v1.14.6+c4799753c
   ```

   The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **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 (1)
   csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
   csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal                       Pending (2)
   csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal                       Pending
   ...
   ```

   (1) A client request CSR.

   (2) A server request CSR.

   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
```

7.7. 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 to a local directory that contains the **kubeconfig** file for your cluster.	The path and name of the configuration file.

7.7.1. Removing RHCOS compute machines from a cluster

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

Prerequisites

- You have added RHEL compute machines to your cluster.

Procedure

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

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

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

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

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

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

   b. Drain all the pods from the node:

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

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

   c. Delete the node:

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

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

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

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

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

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

If your OpenShift Container Platform cluster already includes Red Hat Enterprise Linux (RHEL) compute machines, which are also known as worker machines, you can add more RHEL compute machines to it.

8.1. ABOUT ADDING RHEL COMPUTE NODES TO A CLUSTER

In OpenShift Container Platform 4.2, you have the option of using Red Hat Enterprise Linux (RHEL) machines as compute machines, which are also known as worker machines, in your cluster if you use a user-provisioned infrastructure installation. You must use Red Hat Enterprise Linux CoreOS (RHCOS) machines for the control plane, or master, machines in your cluster.

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

IMPORTANT

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

IMPORTANT

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

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

8.2. SYSTEM REQUIREMENTS FOR RHEL COMPUTE NODES

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

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

IMPORTANT

Only RHEL 7.6 is supported in OpenShift Container Platform 4.2. You must not upgrade your compute machines to RHEL 8.

- 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.

8.2.1. Certificate signing requests management

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

8.3. PREPARING 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 4.2:

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-4.2-rpms"
```

7. Stop and disable firewalld on the host:

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

NOTE

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

8.4. ADDING MORE RHEL COMPUTE MACHINES TO YOUR CLUSTER

You can add more compute machines that use Red Hat Enterprise Linux as the operating system to an OpenShift Container Platform 4.2 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 Command-line Interface (CLI), commonly known as **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-worker-0.example.com
mycluster-worker-1.example.com

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

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

8.5. APPROVING THE CSRS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

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-0  Ready     master  63m  v1.14.6+c4799753c
master-1  Ready     master  63m  v1.14.6+c4799753c
master-2  Ready     master  64m  v1.14.6+c4799753c
worker-0  NotReady  worker  76s  v1.14.6+c4799753c
worker-1  NotReady  worker  70s  v1.14.6+c4799753c
```

   The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **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 1
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal                       Pending 2
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal                       Pending
...
```

   1 A client request CSR.

   2 A server request CSR.

   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
  ```

8.6. 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.
openshift_kubeconfig_path	Specifies a path to a local directory that contains the **kubeconfig** file for your cluster.	The path and name of the configuration file.

CHAPTER 9. DEPLOYING MACHINE HEALTH CHECKS

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

IMPORTANT

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

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

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.

Prerequisites

- Enable a FeatureGate so you can access Technology Preview features.

  NOTE

  Turning on Technology Preview features cannot be undone and prevents upgrades.

9.1. ABOUT MACHINEHEALTHCHECKS

MachineHealthChecks automatically repairs unhealthy Machines in a particular MachinePool.

To monitor machine health, you create a resource to define the configuration for a controller. You set a condition to check for, such as staying in the **NotReady** status for 15 minutes or displaying a permanent condition in the node-problem-detector, and a label for the set of machines to monitor.

NOTE

You cannot apply a MachineHealthCheck to a machine with the master role.

The controller that observes a MachineHealthCheck resource checks for the status that you defined. If a machine fails the health check, it is automatically deleted and a new one is created to take its place. When a machine is deleted, you see a **machine deleted** event. To limit disruptive impact of the machine deletion, the controller drains and deletes only one node at a time.

To stop the check, you remove the resource.

9.2. SAMPLE MACHINEHEALTHCHECK RESOURCE

The MachineHealthCheck resource resembles the following YAML file:

MachineHealthCheck

```
apiVersion: healthchecking.openshift.io/v1alpha1
kind: MachineHealthCheck
metadata:
 name: example 1
 namespace: openshift-machine-api
Spec:
  Selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <label> 2
      machine.openshift.io/cluster-api-machine-type: <label> 3
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> 4
```

1 Specify the name of the MachineHealthCheck to deploy. Include the name of the MachinePool to track.

2 3 Specify a label for the MachinePool that you want to check.

4 Specify the MachineSet to track in **<cluster_name>-<label>-<zone>** format. For example, **prod-node-us-east-1a**.

9.3. CREATING A MACHINEHEALTHCHECK RESOURCE

You can create a MachineHealthCheck resource for all MachinePools in your cluster except the **master** pool.

Prerequisites

- Install the **oc** command line interface.

Procedure

1. Create a **healthcheck.yml** file that contains the definition of your MachineHealthCheck.
2. Apply the **healthcheck.yml** file to your cluster:

```
$ oc apply -f healthcheck.yml
```