OpenShift Container Platform 4.4

机器管理

添加和维护集群机器

Last Updated: 2021-03-11

OpenShift Container Platform 4.4 机器管理

添加和维护集群机器

法律通告

摘要

本文说明如何管理构成 OpenShift Container Platform 集群的机器。某些任务利用 OpenShift Container Platform 集群的增强型自动机器管理功能，另一些任务则要手动完成。本文所述的任务并非对所有安装类型都适用。

目录

第 1 章 创建机器集

1.1. 在 AWS 中创建机器集

您可以在 Amazon Web Services (AWS) 上的 OpenShift Container Platform 集群中创建不同的机器集来满足特定目的。例如，您可以创建基础架构机器集和相关的机器，以便将支持型工作负载转移到新机器上。

1.1.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.4 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。因此，OpenShift Container Platform 4.4 可以在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

Machine

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，Amazon Web Services (AWS) 上的 worker 节点的机器类型可能会定义特定的机器类型和所需的元数据。

机器集

MachineSet 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

以下自定义资源可为集群添加更多功能：

机器自动扩展

MachineAutoscaler 资源自动扩展云中的机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，但不会缩减节点。

机器健康检查

MachineHealthCheck 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。自动扩展器在集群生命周期内尽可能提供平衡。

1.1.2. AWS 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 **us-east-1a** Amazon Web Services (AWS) 区域中运行的机器集，并创建通过 **node-role.kubernetes.io/<role>: ""** 标记的节点。

在本例中，**<infrastructureID>** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **<role>** 则是要添加的节点标签。

```
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 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

2 4 8 指定基础架构 ID、节点标签和区域。

6 7 9 指定要添加的节点标签。

10 为您的 OpenShift Container Platform 节点的 AWS 区域指定有效的 Red Hat Enterprise Linux CoreOS (RHCOS) AMI。

1.1.3. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集来动态管理您选择的特定工作负载的机器计算资源。

先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 创建一个包含机器集自定义资源（CR）示例的新 YAML 文件，并将其命名为 **<file_name>.yaml**。
   确保设置 **<clusterID>** 和 **<role>** 参数值。

   a. 如果不确定要为特定字段设置哪个值，您可以从集群中检查现有的机器集。

```
$ 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. 检查特定机器集的值：

```
$ 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) 集群 ID。

(2) 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

```
$ 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
```

当新机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果机器集不可用，请等待几分钟，然后再次运行命令。

4. 新机器设置可用后，检查机器及其引用的节点的状态：

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

例如：

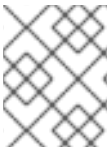
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

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. 查看新节点，并确认新节点具有您指定的标签：

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

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。



注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

后续步骤

如果需要其他可用区中的机器集，请重复此过程来创建更多 MachineSet。

1.2. 在 AZURE 中创建机器集

您可以在 Microsoft Azure 上的 OpenShift Container Platform 集群中创建不同的机器集来满足特定目的。例如，您可以创建基础架构机器集和相关的机器，以便将支持型工作负载转移到新机器上。

1.2.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.4 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。因此，OpenShift Container Platform 4.4 可以在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

Machine

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，Amazon Web Services (AWS) 上的 worker 节点的机器类型可能会定义特定的机器类型和所需的元数据。

机器集

MachineSet 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

以下自定义资源可为集群添加更多功能：

机器自动扩展

MachineAutoscaler 资源自动扩展云中的机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，但不会缩减节点。

机器健康检查

MachineHealthCheck 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。自动扩展器在集群生命周期内尽可能提供平衡。

1.2.2. Azure 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 **centralus** 地区（region）的 **1** Microsoft Azure 区域（zone）中运行的机器集，并创建了通过 **node-role.kubernetes.io/<role>: ""** 标记的节点。

在本例中，**<infrastructureID>** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **<role>** 则是要添加的节点标签。

```
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 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

2 3 8 9 11 15 16 指定要添加的节点标签。

4 6 10 指定基础架构 ID、节点标签和地区。

18 指定您所在地区（region）内要放置 Machine 的区域 (zone)。确保您的地区支持您指定的区域。

1.2.3. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集来动态管理您选择的特定工作负载的机器计算资源。

先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 创建一个包含机器集自定义资源（CR）示例的新 YAML 文件，并将其命名为 **<file_name>.yaml**。
   确保设置 **<clusterID>** 和 **<role>** 参数值。

   a. 如果不确定要为特定字段设置哪个值，您可以从集群中检查现有的机器集。

```
$ 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. 检查特定机器集的值：

```
$ 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 集群 ID。

2 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

```
$ 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
```

当新机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果机器集不可用，请等待几分钟，然后再次运行命令。

4. 新机器设置可用后，检查机器及其引用的节点的状态：

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

例如：

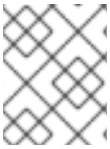
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

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. 查看新节点，并确认新节点具有您指定的标签：

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

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。



注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

后续步骤

如果需要其他可用区中的机器集，请重复此过程来创建更多 MachineSet。

1.3. 在 GCP 中创建机器集

您可以在 Google Cloud Platform (GCP) 上的 OpenShift Container Platform 集群中创建不同的机器集来满足特定目的。例如，您可以创建基础架构机器集和相关的机器，以便将支持型工作负载转移到新机器上。

1.3.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.4 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。因此，OpenShift Container Platform 4.4 可以在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

Machine

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，Amazon Web Services (AWS) 上的 worker 节点的机器类型可能会定义特定的机器类型和所需的元数据。

机器集

MachineSet 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

以下自定义资源可为集群添加更多功能：

机器自动扩展

MachineAutoscaler 资源自动扩展云中的机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，但不会缩减节点。

机器健康检查

MachineHealthCheck 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。自动扩展器在集群生命周期内尽可能提供平衡。

1.3.2. GCP 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 Google Cloud Platform (GCP) 中运行的机器集，并创建通过 **node-role.kubernetes.io/<role>: ""** 标记的节点。

在本例中，**<infrastructureID>** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **<role>** 则是要添加的节点标签。

```
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) 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

(12) (16) 指定基础架构 ID 和节点标签。

(6) (7) (9) 指定要添加的节点标签。

(13) (15) 指定用于集群的 GCP 项目的名称。

1.3.3. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集来动态管理您选择的特定工作负载的机器计算资源。

先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 创建一个包含机器集自定义资源（CR）示例的新 YAML 文件，并将其命名为 **<file_name>.yaml**。
   确保设置 **<clusterID>** 和 **<role>** 参数值。

   a. 如果不确定要为特定字段设置哪个值，您可以从集群中检查现有的机器集。

```
$ 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. 检查特定机器集的值：

```
$ 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 集群 ID。

   2 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

```
$ 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
```

当新机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果机器集不可用，请等待几分钟，然后再次运行命令。

4. 新机器设置可用后，检查机器及其引用的节点的状态：

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

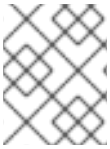
例如：

```
$ 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. 查看新节点，并确认新节点具有您指定的标签：

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

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。



注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

后续步骤

如果需要其他可用区中的机器集，请重复此过程来创建更多 MachineSet。

1.4. 在 OPENSTACK 上创建机器集

您可以在 Red Hat OpenStack Platform（RHOSP）上的 OpenShift Container Platform 集群中创建不同的机器集来满足特定目的。例如，您可以创建基础架构机器集和相关的机器，以便将支持型工作负载转移到新机器上。

1.4.1. Machine API 概述

Machine API 将基于上游 Cluster API 项目的主要资源与自定义 OpenShift Container Platform 资源相结合。

对于 OpenShift Container Platform 4.4 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。因此，OpenShift Container Platform 4.4 可以在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

Machine

描述节点主机的基本单元。机器具有 **providerSpec** 规格，用于描述为不同云平台提供的计算节点的类型。例如，Amazon Web Services (AWS) 上的 worker 节点的机器类型可能会定义特定的机器类型和所需的元数据。

机器集

MachineSet 资源是机器组。机器集适用于机器，复制集则适用于 pod。如果需要更多机器或必须缩减规模，则可以更改机器集的 **replicas** 字段来满足您的计算需求。

以下自定义资源可为集群添加更多功能：

机器自动扩展

MachineAutoscaler 资源自动扩展云中的机器。您可以为指定机器集中的节点设置最小和最大扩展界限，机器自动扩展就会维护此范围内的节点。**ClusterAutoscaler** 对象存在后，**MachineAutoscaler** 对象生效。**ClusterAutoscaler** 和 **MachineAutoscaler** 资源都由 **ClusterAutoscalerOperator** 对象提供。

集群自动扩展

此资源基于上游集群自动扩展项目。在 OpenShift Container Platform 实现中，它通过扩展机器集 API 来与 Machine API 集成。您可以为核心、节点、内存和 GPU 等资源设置集群范围的扩展限制。您可以设置优先级，使集群对 Pod 进行优先级排序，以便不针对不太重要的 Pod 使新节点上线。您还可以设置扩展策略，以便可以扩展节点，但不会缩减节点。

机器健康检查

MachineHealthCheck 资源可检测机器何时处于不健康状态并将其删除，然后在支持的平台上生成新的机器。

在 OpenShift Container Platform 版本 3.11 中，您无法轻松地推出多区架构，因为集群不负责管理机器置备。自 OpenShift Container Platform 版本 4.1 起，此过程变得更加容易。每个机器集限定在一个区域，因此安装程序可以代表您将机器集分发到多个可用区。然后，由于您的计算是动态的，因此在面对区域故障时，您始终都有一个区域来应对必须重新平衡机器的情况。自动扩展器在集群生命周期内尽可能提供平衡。

1.4.2. RHOSP 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 Red Hat OpenStack Platform（RHOSP）上运行的机器集，并创建带有 **node-role.openshift.io/<node_role>: ""** 标记的节点

在本例中，**infrastructure_ID** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **node_role** 则是要添加的节点标签。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 1
    machine.openshift.io/cluster-api-machine-role: <node_role> 2
    machine.openshift.io/cluster-api-machine-type: <node_role> 3
  name: <infrastructure_ID>-<node_role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_ID> 7
        machine.openshift.io/cluster-api-machine-role: <node_role> 8
        machine.openshift.io/cluster-api-machine-type: <node_role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_ID>-<node_role> 10
    spec:
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          kind: OpenstackProviderSpec
          networks:
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
                tags: openshiftClusterID=<infrastructure_ID>
          securityGroups:
          - filter: {}
            name: <infrastructure_ID>-<node_role>
          serverMetadata:
            Name: <infrastructure_ID>-<node_role>
            openshiftClusterID: <infrastructure_ID>
          tags:
          - openshiftClusterID=<infrastructure_ID>
          trunk: true
          userDataSecret:
            name: <node_role>-user-data 11
```

1 5 7 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

2 3 8 9 11 指定要添加的节点标签。

4 6 10 指定基础架构 ID 和节点标签。

1.4.3. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集来动态管理您选择的特定工作负载的机器计算资源。

先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 创建一个包含机器集自定义资源（CR）示例的新 YAML 文件，并将其命名为 **<file_name>.yaml**。
   确保设置 **<clusterID>** 和 **<role>** 参数值。

   a. 如果不确定要为特定字段设置哪个值，您可以从集群中检查现有的机器集。

   ```
   $ 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. 检查特定机器集的值：

   ```
   $ 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 集群 ID。

2 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

```
$ 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
```

当新机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果机器集不可用，请等待几分钟，然后再次运行命令。

4. 新机器设置可用后，检查机器及其引用的节点的状态：

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

例如：

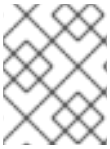
```
$ oc describe machine agl030519-vplxk-infra-us-east-1a -n openshift-machine-api

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. 查看新节点，并确认新节点具有您指定的标签：

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

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。



注意

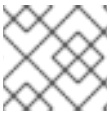
对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

后续步骤

如果需要其他可用区中的机器集，请重复此过程来创建更多 MachineSet。

第 2 章 手动扩展机器集

您可以在机器集中添加或删除机器的实例。

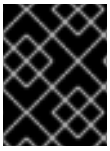


注意

如果您需要在扩展之外修改机器集的各个方面，请参阅 修改机器集。

2.1. 先决条件

- 如果启用了集群范围代理并要扩展未包含在安装配置的 **networking.machineNetwork[].cidr** 中的 worker，您必须将 worker 添加到 Proxy 对象的 **noProxy** 字段，以防发生连接问题。



重要

此过程不适用于自己手动置备机器的集群。您只能在使用机器 API 的集群中使用高级机器管理和扩展功能。

2.2. 手动扩展机器集

如果您必须在机器集中添加或移除机器实例，则可以手动扩展机器集。

先决条件

- 安装 OpenShift Container Platform 集群和 **oc** 命令行。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 查看集群中的机器集：

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

机器集以 **<clusterid>-worker-<aws-region-az>** 的形式列出。

2. 扩展机器集：

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

或者：

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

您可以扩展或缩减机器集。需要过几分钟以后新机器才可用。

2.3. 机器集删除策略

Random、**Newest** 和 **Oldest** 是三个支持的删除选项。默认值为 **Random**，表示在扩展机器时随机选择并删除机器。通过修改特定机器集，可以根据用例设置删除策略：

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

无论删除策略是什么，都可通过在相关机器上添加 **machine.openshift.io/cluster-api-delete-machine** 注解来指定机器删除的优先级。

重要

默认情况下，OpenShift Container Platform 路由器 Pod 部署在 worker 上。由于路由器需要访问某些集群资源（包括 Web 控制台），除非先重新放置了路由器 Pod，否则请不要将 worker 机器集扩展为 **0**。

注意

当用户需要特定的服务必须运行在特定节点，在 worker 机器集进行缩减时需要忽略这些服务时，可以使用自定义机器集。这可防止服务被中断。

第 3 章 修改机器集

您可以对机器集进行更改，例如添加标签、更改实例类型或更改块存储。

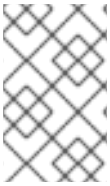
注意

如果您需要在不进行其他更改的情况下扩展机器集，请参阅手动扩展机器集。

3.1. 修改机器集

要更改机器集，编辑 **MachineSet** YAML。然后，通过删除每台机器或将机器设置为 **0** 个副本来删除与机器设置关联的所有机器。然后，将副本数量调回所需的数量。您对机器集所做的更改不会影响现有的机器。

如果您需要在不进行其他更改的情况下扩展机器集，则不需要删除机器。



注意

默认情况下，OpenShift Container Platform 路由器 Pod 部署在 worker 上。由于路由器需要访问某些集群资源（包括 Web 控制台），除非先重新放置了路由器 Pod，否则请不要将 worker 机器集扩展为 **0**。

先决条件

- 安装 OpenShift Container Platform 集群和 **oc** 命令行。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 编辑机器集：

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

2. 将机器缩减为 **0**:

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

或者：

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

等待机器被删除。

3. 根据需要扩展机器设置：

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

或者：

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

等待机器启动。新机器包含您对机器集所做的更改。

第 4 章 删除机器

您可以删除特定的机器。

4.1. 删除一个特定的机器

您可以删除特定的机器。

先决条件

- 安装 OpenShift Container Platform 集群：
- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

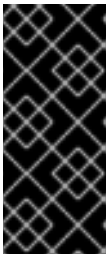
1. 查看集群中的机器，找到要删除的机器：

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

命令输出包含 **<clusterid>-worker-<cloud_region>** 格式的机器列表。

2. 删除机器：

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



重要

默认情况下，机器控制器会尝试排空在机器上运行的节点，直到成功为止。在某些情况下，如错误配置了 pod 的中断预算，节点排空操作可能无法成功完成，从而导致机器无法被删除。您可以在特定机器上使用 "machine.openshift.io/exclude-node-draining" 注解来跳过排空节点的过程。如果要删除的机器属于机器集，则会立即创建一个新机器来满足指定的副本数要求。

第 5 章 将自动扩展应用到 OPENSHIFT CONTAINER PLATFORM 集群

将自动扩展应用到 OpenShift Container Platform 集群涉及部署集群自动扩展，然后为集群中的每种 Machine 类型部署机器自动扩展。

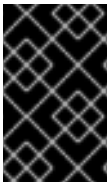
重要

您只能在机器 API 正常工作的集群中配置集群自动扩展。

5.1. 关于集群自动扩展

集群自动扩展会调整 OpenShift Container Platform 集群的大小，以满足其当前的部署需求。它使用 Kubernetes 样式的声明性参数来提供基础架构管理，而且这种管理不依赖于特定云提供商的对象。集群自动控制会在集群范围内有效，不与特定的命名空间相关联。

当由于资源不足而无法在任何当前节点上调度 Pod 时，或者在需要另一个节点来满足部署需求时，集群自动扩展会增加集群的大小。集群自动扩展不会将集群资源增加到超过您指定的限制。



重要

确保您所创建的 **ClusterAutoscaler** 资源定义中的 **maxNodesTotal** 值足够大，足以满足计算集群中可能的机器总数。此值必须包含 control plane 机器的数量以及可扩展至的机器数量。

如果相当长的一段时间内都不需要某些节点，例如集群资源使用率较低并且所有重要 Pod 都可以安置在其他节点上时，集群自动扩展会减小集群的大小。

如果节点上存在以下类型的 pod，集群自动扩展不会删除该节点：

- 具有限制性 pod 中断预算（PDB）的 Pod。
- 默认不在节点上运行的 Kube 系统 Pod。
- 没有 PDB 或 PDB 限制性太强的 Kube 系统 pod。
- 不受控制器对象支持的 Pod,如部署、副本集或有状态集。
- 具有本地存储的 Pod。
- 因为缺乏资源、节点选择器或关联性不兼容或有匹配的反关联性等原因而无法移至其他位置的 Pod。
- 具有 **"cluster-autoscaler.kubernetes.io/safe-to-evict": "false"** 注解的 Pod，除非同时也具有 **"cluster-autoscaler.kubernetes.io/safe-to-evict": "true"** 注解。

如果配置集群自动扩展，则需要额外的使用限制：

- 不要直接修改位于自动扩展节点组中的节点。同一节点组中的所有节点具有相同的容量和标签，并且运行相同的系统 Pod。
- 指定适合您的 Pod 的请求。
- 如果需要防止 Pod 被过快删除，请配置适当的 PDB。

- 确认您的云提供商配额足够大，能够支持您配置的最大节点池。
- 不要运行其他节点组自动扩展器，特别是云提供商提供的自动扩展器。

pod 横向自动扩展（HPA）和集群自动扩展以不同的方式修改集群资源。HPA 根据当前的 CPU 负载更改部署或副本集的副本数。如果负载增加，HPA 会创建新的副本，不论集群可用的资源量如何。如果没有足够的资源，集群自动扩展会添加资源，以便 HPA 创建的 pod 可以运行。如果负载减少，HPA 会停止一些副本。如果此操作导致某些节点利用率低下或完全为空，集群自动扩展会删除不必要的节点。

集群自动扩展会考虑 pod 优先级。如果集群没有足够的资源，则"Pod 优先级和抢占"功能可根据优先级调度 Pod，但集群自动扩展会确保集群具有运行所有 Pod 需要的资源。为满足这两个功能，集群自动扩展包含一个优先级截止函数。您可以使用此截止函数来调度"尽力而为"的 Pod，它们不会使集群自动扩展增加资源，而是仅在有可用备用资源时运行。

优先级低于截止值的 Pod 不会导致集群扩展或阻止集群缩减。系统不会添加新节点来运行 Pod，并且可能会删除运行这些 Pod 的节点来释放资源。

5.2. 关于机器自动扩展

机器自动扩展会调整您在 OpenShift Container Platform 集群中部署的机器集中的 Machine 数量。您可以扩展默认 **worker** 机器集，以及您创建的其他机器集。当集群没有足够资源来支持更多部署时，机器自动扩展会增加 Machine。对 **MachineAutoscaler** 资源中的值（如最小或最大实例数量）的任何更改都会立即应用到目标机器设置中。

重要

您必须部署机器自动扩展才能使用集群自动扩展功能来扩展机器。集群自动扩展使用机器自动扩展集上的注解来确定可扩展的资源。如果您在没有定义机器自动扩展的情况下定义集群自动扩展，集群自动扩展永远不会扩展集群。

5.3. 配置集群自动扩展

首先，部署集群自动扩展来管理 OpenShift Container Platform 集群中的资源自动扩展。

注意

由于集群自动扩展的范围仅限于整个集群，因此只能为集群创建一个集群自动扩展。

5.3.1. ClusterAutoscaler 资源定义

此 **ClusterAutoscaler** 资源定义显示了集群自动扩展的参数和示例值。

```
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. 指定 Pod 必须超过哪一优先级才能让机器自动扩展部署更多节点。输入一个 32 位整数值。**podPriorityThreshold** 值将与您分配给每个 Pod 的 **PriorityClass** 值进行比较。
2. 指定要部署的最大节点数。这个值是集群中部署的机器总数，而不仅仅是自动扩展器控制的机器。确保这个值足够大，足以满足所有 control plane 和计算机器以及您在 **MachineAutoscaler** 资源中指定的副本总数。
3. 指定要部署的最小内核数。
4. 指定要部署的最大内核数。
5. 指定每个节点的最小内存量，以 GiB 为单位。
6. 指定每个节点的最大内存量，以 GiB 为单位。
7. 10 （可选）指定要部署的 GPU 节点的类型。只有 **nvidia.com/gpu** 和 **amd.com/gpu** 是有效的类型。
8. 11 指定要部署的最小 GPU 数。
9. 12 指定要部署的最大 GPU 数。
13. 在此部分中，您可以指定每个操作要等待的时长，可以使用任何有效的 ParseDuration 间隔，包括 **ns**、**us**、**ms**、**s**、**m** 和 **h**。
14. 指定集群自动扩展是否可以删除不必要的节点。
15. （可选）指定在最近*添加*节点之后要等待多久才能删除节点。如果不指定值，则使用默认值 **10m**。
16. 指定在最近*删除*节点之后要等待多久才能删除节点。如果不指定值，则使用默认值 **10s**。
17. 指定在发生缩减失败之后要等待多久才能删除节点。如果不指定值，则使用默认值 **3m**。
18. 指定要经过多长时间之后，不需要的节点才符合删除条件。如果不指定值，则使用默认值 **10m**。

5.3.2. 部署集群自动扩展

要部署集群自动扩展，请创建一个 **ClusterAutoscaler** 资源实例。

流程

1. 为 **ClusterAutoscaler** 资源创建一个 YAML 文件，其中包含自定义的资源定义。
2. 在集群中创建资源：

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

1 **<filename>** 是您自定义的资源文件的名称。

5.4. 后续步骤

- 配置集群自动扩展后，必须至少配置一台机器自动扩展。

5.5. 配置机器自动扩展

部署集群自动扩展后，部署 **MachineAutoscaler** 资源来引用用于扩展集群的机器集。

重要

部署 **ClusterAutoscaler** 资源后，必须至少部署一个 **MachineAutoscaler** 资源。

注意

您必须为每个机器集配置单独的资源。请记住，每个地区中的机器集都不同，因此请考虑是否要在多个地区中启用机器扩展。扩展的机器集必须至少有一台机器。

5.5.1. MachineAutoscaler 资源定义

此 **MachineAutoscaler** 资源定义显示了机器自动扩展器的参数和示例值。

```
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 指定机器自动扩展名称。为了更容易识别此机器自动扩展会扩展哪些机器集，请指定或注明要扩展的机器集的名称。机器集名称的格式如下：**<clusterid>-<machineset>-<aws-region-az>**

2 指定在机器自动扩展启动集群扩展后必须保留在指定区域中的指定类型的最小机器数量。不要将此值设置为 **0**。

3 指定集群自动扩展初始化集群扩展后可在指定 AWS 区域中部署的指定类型的最大机器数量。确保 **ClusterAutoscaler** 资源定义的 **maxNodesTotal** 值足够大，以便机器自动扩展器可以部署这个数量

的机器。

4 在本小节中，提供用于描述要扩展的现有机器集的值。

5 **kind** 参数值始终为 **MachineSet**。

6 **name** 值必须与现有机器集的名称匹配，如 **metadata.name** 参数值所示。

5.5.2. 部署机器自动扩展

要部署机器自动扩展，请创建一个 **MachineAutoscaler** 资源实例。

流程

1. 为 **MachineAutoscaler** 资源创建一个 YAML 文件，其中包含自定义的资源定义。
2. 在集群中创建资源：

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

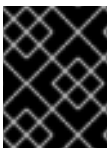
1 **<filename>** 是您自定义的资源文件的名称。

5.6. 其他资源

- 如需有关 Pod 优先级的更多信息，请参阅在 OpenShift Container Platform 的 Pod 调度决策中纳入 Pod 优先级。

第 6 章 创建基础架构机器集

您可以创建一个机器集来仅托管基础架构组件。将特定的 Kubernetes 标签应用于这些机器，然后将基础架构组件更新为仅在那些机器上运行。这些基础架构节点不计入运行环境所需的订阅总数中。



重要

与早期版本的 OpenShift Container Platform 不同，您无法将基础架构组件移到 master 机器。要移动组件，必须创建新的机器集。

6.1. OPENSHIFT CONTAINER PLATFORM 基础架构组件

以下 OpenShift Container Platform 组件是基础架构组件：

- 在主控机上运行的 Kubernetes 和 OpenShift Container Platform control plane 服务
- 默认路由器
- 容器镜像 registry
- 集群指标收集或监控服务
- 集群聚合日志
- 服务代理

运行任何其他容器、Pod 或组件的所有节点都需要是您的订阅可涵盖的 worker 节点。

6.2. 为生产环境创建基础架构机器集

在生产环境部署中，至少部署三个机器集来容纳基础架构组件。日志记录聚合解决方案和服务网格都部署 Elasticsearch，而且 Elasticsearch 需要三个安装到不同节点上的实例。为获得高可用性，将这些节点部署到不同的可用区。由于每个可用区需要不同的机器集，因此至少创建三台机器集。

6.2.1. 为不同云创建机器集

使用云的示例机器集。

6.2.1.1. AWS 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 **us-east-1a** Amazon Web Services (AWS) 区域中运行的机器集，并创建通过 **node-role.kubernetes.io/<role>: ""** 标记的节点。

在本例中，**<infrastructureID>** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **<role>** 则是要添加的节点标签。

```
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 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

2 4 8 指定基础架构 ID、节点标签和区域。

6 7 9 指定要添加的节点标签。

10 为您的 OpenShift Container Platform 节点的 AWS 区域指定有效的 Red Hat Enterprise Linux CoreOS (RHCOS) AMI。

6.2.1.2. Azure 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 **centralus** 地区（region）的 **1** Microsoft Azure 区域（zone）中运行的机器集，并创建了通过 **node-role.kubernetes.io/<role>: ""** 标记的节点。

在本例中，**<infrastructureID>** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **<role>** 则是要添加的节点标签。

```
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) 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

(2) (3) (8) (9) (11) (15) (16) 指定要添加的节点标签。

(4) (6) (10) 指定基础架构 ID、节点标签和地区。

(18) 指定您所在地区（region）内要放置 Machine 的区域 (zone)。确保您的地区支持您指定的区域。

6.2.1.3. GCP 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 Google Cloud Platform (GCP) 中运行的机器集，并创建通过 **node-role.kubernetes.io/<role>: ""** 标记的节点。

在本例中，**<infrastructureID>** 是基础架构 ID 标签，该标签基于您在置备集群时设定的集群 ID，而 **<role>** 则是要添加的节点标签。

```
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 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

12 16 指定基础架构 ID 和节点标签。

6 7 9 指定要添加的节点标签。

13 15 指定用于集群的 GCP 项目的名称。

6.2.2. 创建机器集

除了安装程序创建的机器集之外，还可创建自己的机器集来动态管理您选择的特定工作负载的机器计算资源。

先决条件

- 部署一个 OpenShift Container Platform 集群。
- 安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录 **oc**。

流程

1. 创建一个包含机器集自定义资源（CR）示例的新 YAML 文件，并将其命名为 **<file_name>.yaml**。
   确保设置 **<clusterID>** 和 **<role>** 参数值。

   a. 如果不确定要为特定字段设置哪个值，您可以从集群中检查现有的机器集。

```
$ 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. 检查特定机器集的值：

```
$ 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. 集群 ID。
2. 默认节点标签。

2. 创建新的 **MachineSet** CR:

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

3. 查看机器集列表：

```
$ 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
```

当新机器集可用时，**DESIRED** 和 **CURRENT** 的值会匹配。如果机器集不可用，请等待几分钟，然后再次运行命令。

4. 新机器设置可用后，检查机器及其引用的节点的状态：

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

例如：

```
$ 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. 查看新节点，并确认新节点具有您指定的标签：

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

查看命令输出，并确认 **node-role.kubernetes.io/<your_label>** 列在 **LABELS** 列表中。

注意

对机器集的任何更改都不会应用到机器集拥有的现有机器。例如，对现有机器集编辑或添加的标签不会传播到与该机器集关联的现有机器和节点。

后续步骤

如果需要其他可用区中的机器集，请重复此过程来创建更多 MachineSet。

6.3. 为基础架构节点分配机器设置资源

在创建了基础架构机器集后，**worker** 和 **infra** 角色将应用到新的 infra 节点。应用 **infra** 角色的节点不会被计算为运行环境所需的订阅总数，即使也应用了 **worker** 角色。

但是，如果将 infra 节点分配为 worker，则用户工作负载可能会意外地分配给 infra 节点。要避免这种情况，您必须将污点应用到 infra 节点，并为您要控制的 pod 应用容限。

6.3.1. 使用污点和容限绑定基础架构节点工作负载

如果您有一个分配了 **infra** 和 **worker** 角色的 infra 节点，您必须配置该节点，以便不为其分配用户工作负载。

先决条件

- 在 OpenShift Container Platform 集群中配置额外的 **MachineSet** 对象。

流程

1. 使用以下命令，将污点添加到 infra 节点，以防止在其上调度用户工作负载：

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

例如：

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

本例在 **node1** 上放置一个键为 **node-role.kubernetes.io/infra** 的污点，污点是 **NoSchedule**。具有 **NoSchedule effect** 的节点仅调度容许该污点的 pod，但允许现有 pod 继续调度到该节点上。

注意

如果使用 descheduler，则违反了节点污点的 pod 可能会从集群驱除。

2. 为要在 infra 节点上调度的 pod 配置添加容限，如路由器、registry 和监控工作负载。在 **Pod** 对象规格中添加以下代码：

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

(1) 指定添加到节点的效果。

(2) 指定添加到节点的键。

(3) 指定 **Exists** Operator，以要求节点上存在一个带有键为 **node-role.kubernetes.io/infra** 的污点。

此容忍度与 **oc adm taint** 命令创建的污点匹配。具有此容忍度的 pod 可以调度到 infra 节点上。

注意

并不总是能够将通过 OLM 安装的 Operator 的 Pod 移到 infra 节点。移动 Operator Pod 的能力取决于每个 Operator 的配置。

3. 使用调度程序将 pod 调度到 infra 节点。详情请参阅*控制节点上的 pod 放置*的文档。

其他资源

- 如需了解有关将 pod 调度到节点的信息，请参阅使用调度程序控制 pod 放置。
- 如需有关将 pod 调度到 infra 节点的说明，请参阅将资源移动到基础架构机器集。

6.4. 将资源移到基础架构机器集

默认情况下，您的集群中已部署了某些基础架构资源。您可将它们移至您创建的基础架构机器集。

6.4.1. 移动路由器

您可以将路由器 pod 部署到不同的机器集中。默认情况下，pod 部署到 worker 节点。

先决条件

- 在 OpenShift Container Platform 集群中配置额外的机器集。

流程

1. 查看路由器 Operator 的 **IngressController** 自定义资源：

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

命令输出类似于以下文本：

```
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. 编辑 **ingresscontroller** 资源，并更改 **nodeSelector** 以使用**infra** 标签：

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

在 **spec** 中添加使用 **infra** 标签的 **nodeSelector** 的部分，如下所示：

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

3. 确认路由器 Pod 在 **infra** 节点上运行。

   a. 查看路由器 Pod 列表，并记下正在运行的 Pod 的节点名称：

```
$ 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>
```

在本例中，正在运行的 Pod 位于 **ip-10-0-217-226.ec2.internal** 节点上。

b. 查看正在运行的 Pod 的节点状态：

```
$ oc get node <node_name> 1

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

1 指定从 Pod 列表获得的 **<node_name>**。

由于角色列表包含 **infra**，因此 Pod 在正确的节点上运行。

6.4.2. 移动默认 registry

您需要配置 registry Operator，以便将其 Pod 部署到其他节点。

先决条件

- 在 OpenShift Container Platform 集群中配置额外的机器集。

流程

1. 查看 **config/instance** 对象：

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

输出类似于以下文本：

```
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. 编辑 **config/instance** 对象：

```
$ oc edit config/cluster
```

3. 在对象的 **spec** 部分添加以下文本行：

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

4. 验证 registry pod 已移至基础架构节点。

   a. 运行以下命令，以识别 registry pod 所在的节点：

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

   b. 确认节点具有您指定的标签：

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

   查看命令输出，并确认 **node-role.kubernetes.io/infra** 列在 **LABELS** 列表中。

6.4.3. 移动监控解决方案

默认情况下，部署包含 Prometheus、Grafana 和 AlertManager 的 Prometheus Cluster Monitoring 堆栈来提供集群监控功能。它由 Cluster Monitoring Operator 进行管理。若要将其组件移到其他机器上，需要创建并应用自定义配置映射。

流程

1. 将以下 **ConfigMap** 定义保存为 **cluster-monitoring-configmap.yaml** 文件：

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

```
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  k8sPrometheusAdapter:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  kubeStateMetrics:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  telemeterClient:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  openshiftStateMetrics:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
  thanosQuerier:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
```

运行此配置映射会强制将监控堆栈的组件重新部署到基础架构节点。

2. 应用新的配置映射：

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

3. 观察监控 pod 移至新机器：

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

4. 如果组件没有移到 **infra** 节点，请删除带有这个组件的 pod:

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

已删除 pod 的组件在 **infra** 节点上重新创建。

其他资源

- 如需了解有关移动 OpenShift Container Platform 组件的信息，请参阅监控文档。

6.4.4. 移动集群日志资源

您可以配置 Cluster Logging Operator，以将任何或所有 Cluster Logging 组件、Elasticsearch、Kibana 和 Curator 的 Pod 部署到不同的节点上。您无法将 Cluster Logging Operator Pod 从其安装位置移走。

例如，您可以因为 CPU、内存和磁盘要求较高而将 Elasticsearch Pod 移到一个单独的节点上。

注意

您应该将机器设置为至少使用 6 个副本。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。默认情况下没有安装这些功能。

流程

1. 编辑 **openshift-logging** 项目中的 **ClusterLogging** 自定义资源（CR）：

```
$ 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 添加 **nodeSelector** 参数，并设为适用于您想要移动的组件的值。您可以根据为节点指定的值，按所示格式使用 **nodeSelector** 或使用 **<key>: <value>** 对。

验证步骤

要验证组件是否已移动，您可以使用 **oc get pod -o wide** 命令。

例如：

- 您需要移动来自 **ip-10-0-147-79.us-east-2.compute.internal** 节点上的 Kibana pod：

```
$ 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>
```

- 您需要将 Kibana Pod 移到 **ip-10-0-139-48.us-east-2.compute.internal** 节点，该节点是一个专用的基础架构节点：

```
$ oc get nodes
NAME                                         STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.17.1
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.17.1
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.17.1
```

请注意，该节点具有 **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: ''
....
```

- 要移动 Kibana pod，编辑 **ClusterLogging** CR 以添加节点选择器：

```
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 添加节点选择器以匹配节点规格中的 label。

- 保存 CR 后，当前 Kibana Pod 将被终止，新的 Pod 会被部署：

```
$ 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
```

- 新 pod 位于 **ip-10-0-139-48.us-east-2.compute.internal** 节点上：

```
$ 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>
```

- 片刻后，原始 Kibana Pod 将被删除。

```
$ 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
```

第 7 章 用户置备的基础架构

7.1. 在 OPENSHIFT CONTAINER PLATFORM 集群中添加 RHEL 计算机器

在 OpenShift Container Platform 中，您可以将 Red Hat Enterprise Linux (RHEL) 计算或 worker 添加到用户置备的基础架构集群中。计算机器上只能使用 RHEL 作为操作系统。

7.1.1. 关于在集群中添加 RHEL 计算节点

在 OpenShift Container Platform 4.4 中，如果使用用户置备的基础架构安装，您可以选择将 Red Hat Enterprise Linux (RHEL) 机器用作集群中的计算机器（也称为 worker）。集群中的 control plane 或主控机器（master）必须使用 Red Hat Enterprise Linux CoreOS (RHCOS)。

与所有使用用户置备的基础架构的安装一样，如果选择在集群中使用 RHEL 计算机器，您将需要负责所有操作系统生命周期的管理和维护任务，包括执行系统更新、应用补丁以及完成所有其他必要的任务。

重要

由于从集群中的机器上删除 OpenShift Container Platform 需要破坏操作系统，因此您必须对添加到集群中的所有 RHEL 机器使用专用的硬件。

重要

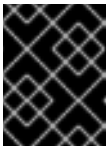
添加到 OpenShift Container Platform 集群的所有 RHEL 机器上都禁用内存交换功能。您无法在这些机器上启用交换内存。

您必须在初始化 control plane 之后将所有 RHEL 计算机器添加到集群。

7.1.2. RHEL 计算节点的系统要求

OpenShift Container Platform 环境中的 Red Hat Enterprise Linux (RHEL) 计算机器（也称为 worker）主机必须满足以下最低硬件规格和系统级别要求。

- 您的红帽帐户必须具有有效的 OpenShift Container Platform 订阅。如果没有，请与您的销售代表联系以了解更多信息。
- 生产环境必须提供能支持您的预期工作负载的计算机器。作为集群管理员，您必须计算预期的工作负载，再加上大约 10% 的开销。对于生产环境，请分配足够的资源，以防止节点主机故障影响您的最大容量。
- 所有系统都必须满足以下硬件要求：
  - 物理或虚拟系统，或在公有或私有 IaaS 上运行的实例。
  - 基础操作系统：使用 "Minimal" 安装选项的 RHEL 7.6-7.8。

    重要

    OpenShift Container Platform 4.4 仅支持 RHEL 7.6-7.8。您不能将计算机器升级到 RHEL 8。

  - 如果以 FIPS 模式部署 OpenShift Container Platform，则需要在 RHEL 机器上启用 FIPS，然后才能引导它。请参阅 RHEL 7 文档中的启用 FIPS 模式 。

- NetworkManager 1.0 或更高版本。
- 1 个 vCPU。
- 最小 8 GB RAM。
- 最小15 GB 硬盘空间，用于包含 **/var/** 的文件系统。
- 最小1 GB 硬盘空间，用于包含 **/usr/local/bin/** 的文件系统。
- 最小1 GB 硬盘空间，用于包含系统临时目录的文件系统。系统的临时目录根据 Python 标准库中 tempfile 模块中定义的规则确定。

- 每个系统都必须满足您的系统提供商的任何其他要求。例如，如果在 VMware vSphere 上安装了集群，必须根据其存储准则配置磁盘，而且必须设置 **disk.enableUUID=true** 属性。
- 每个系统都必须能够使用可解析的主机名访问集群的 API 端点。任何现有的网络安全访问控制都必须允许系统访问集群的 API 服务端点。

7.1.2.1. 证书签名请求管理

在使用您置备的基础架构时，集群只能有限地访问自动机器管理，因此您必须提供一种在安装后批准集群证书签名请求 (CSR) 的机制。**kube-controller-manager** 只能批准 kubelet 客户端 CSR。 **machine-approver** 无法保证使用 kubelet 凭证请求的提供证书的有效性，因为它不能确认是正确的机器发出了该请求。您必须决定并实施一种方法，以验证 kubelet 提供证书请求的有效性并进行批准。

7.1.3. 准备机器以运行 Playbook

在将使用 Red Hat Enterprise Linux 作为操作系统的计算机器添加到 OpenShift Container Platform 4.4 集群之前，必须准备一台机器来运行 playbook。这台机器不是集群的一部分，但必须能够访问集群。

先决条件

- 在运行 playbook 的机器上安装 OpenShift CLI（**oc**）。
- 以具有 **cluster-admin** 权限的用户身份登录。

流程

1. 确保机器上具有集群的 **kubeconfig** 文件，以及用于安装集群的安装程序。若要实现这一目标，一种方法是使用安装集群时所用的同一台机器。
2. 配置机器，以访问您计划用作计算机器的所有 RHEL 主机。您可以使用公司允许的任何方法，包括使用 SSH 代理或 VPN 的堡垒主机。
3. 在运行 playbook 的机器上配置一个用户，该用户对所有 RHEL 主机具有 SSH 访问权限。

重要

如果使用基于 SSH 密钥的身份验证，您必须使用 SSH 代理来管理密钥。

4. 如果还没有这样做，请使用 RHSM 注册机器，并为它附加一个带有 **OpenShift** 订阅的池：
   a. 使用 RHSM 注册机器：

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

b. 从 RHSM 获取最新的订阅数据：

```
# subscription-manager refresh
```

c. 列出可用的订阅：

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

d. 在上一命令的输出中，找到 OpenShift Container Platform 订阅的池 ID 并附加该池：

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

5. 启用 OpenShift Container Platform 4.4 所需的存储库：

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms" \
    --enable="rhel-7-server-ose-4.4-rpms"
```

6. 安装所需的软件包，包括 **openshift-ansible**:

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

openshift-ansible 软件包提供了安装实用程序，并且会拉取将 RHEL 计算节点添加到集群所需要的其他软件包，如 Ansible、playbook 和相关的配置文件。**openshift-clients** 提供 **oc** CLI，**jq** 软件包则可改善命令行中 JSON 输出的显示。

7.1.4. 准备 RHEL 计算节点

在将 Red Hat Enterprise Linux (RHEL) 机器添加到 OpenShift Container Platform 集群之前，您必须将每台主机注册到 Red Hat Subscription Manager (RHSM)，为其附加有效的 OpenShift Container Platform 订阅，并且启用所需的存储库。

1. 在每一主机上进行 RHSM 注册：

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

2. 从 RHSM 获取最新的订阅数据：

```
# subscription-manager refresh
```

3. 列出可用的订阅：

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

4. 在上一命令的输出中，找到 OpenShift Container Platform 订阅的池 ID 并附加该池：

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

5. 禁用所有 yum 存储库：

   a. 禁用所有已启用的 RHSM 存储库：

   ```
   # subscription-manager repos --disable="*"
   ```

   b. 列出剩余的 yum 存储库，并记录它们在 **repo id** 下的名称（若有）：

   ```
   # yum repolist
   ```

   c. 使用 **yum-config-manager** 禁用剩余的 yum 存储库：

   ```
   # yum-config-manager --disable <repo_id>
   ```

   或者，禁用所有存储库：

   ```
   yum-config-manager --disable \*
   ```

   请注意，有大量可用存储库时可能需要花费几分钟

6. 仅启用 OpenShift Container Platform 4.4 需要的存储库：

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

7. 停止并禁用主机上的防火墙：

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

注意

请不要在以后启用防火墙。如果这样做，则无法访问 worker 上的 OpenShift Container Platform 日志。

7.1.5. 在集群中添加 RHEL 计算机器

您可以将使用 Red Hat Enterprise Linux 作为操作系统的计算机器添加到 OpenShift Container Platform 4.4 集群中。

先决条件

- 运行 playbook 的机器上已安装必需的软件包并且执行了必要的配置。
- RHEL 主机已做好安装准备。

流程

在为运行 playbook 而准备的机器上执行以下步骤：

1. 创建一个名为 **/<path>/inventory/hosts** 的 Ansible 清单文件，以定义您的计算机器主机和必要的变量：

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

openshift_kubeconfig_path="~/.kube/config" 3

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

1. 指定要在远程计算机器上运行 Ansible 任务的用户名。
2. 如果不将 **root** 指定为 **ansible_user**，您必须将 **ansible_become** 设置为 **True**，并为该用户分配 sudo 权限。
3. 指定集群的 **kubeconfig** 文件的路径和文件名。
4. 列出要添加到集群中的每台 RHEL 机器。必须为每个主机提供完全限定域名。此名称是集群用来访问机器的主机名，因此请设置用于访问该机器的正确公用或私有名称。

2. 运行 playbook：

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

1. 对于**<path>**，指定您创建的Ansible库存文件的路径。

7.1.6. 批准机器的证书签名请求

将机器添加到集群时，会为您添加的每台机器生成两个待处理证书签名请求（CSR）。您必须确认这些 CSR 已获得批准，或根据需要自行批准。客户端请求必须首先被批准，然后是服务器请求。

先决条件

- 您已将机器添加到集群中。

流程

1. 确认集群可以识别这些机器：

```
# oc get nodes

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

输出将列出您创建的所有机器。

2. 检查待处理的 CSR，并确保可以看到添加到集群中的每台机器都有 **Pending** 或 **Approved** 状态的客户端请求：

```
$ oc get csr

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

在本例中，两台机器加入了集群。您可能会在列表中看到更多已批准的 CSR。

3. 如果 CSR 没有获得批准，请在所添加机器的所有待处理 CSR 都处于 **Pending** 状态后，为您的集群机器批准这些 CSR：

注意

由于 CSR 会自动轮转，因此请在将机器添加到集群后一小时内批准您的 CSR。如果没有在一小时内批准，证书将会轮转，每个节点将会存在多个证书。您必须批准所有这些证书。批准初始 CSR 后，集群的 **kube-controller-manager** 会自动批准后续的节点客户端 CSR。您必须实施一个方法来自动批准 kubelet 提供的证书请求。

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

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

4. 现在，您的客户端请求已被批准，您必须查看添加到集群中的每台机器的服务器请求：

```
$ oc get csr
```

输**出示例**

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

5. 如果剩余的 CSR 没有被批准，且处于 **Pending** 状态，请批准集群机器的 CSR：

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

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

6. 批准所有客户端和服务器 CSR 后，器将处于 **Ready** 状态。运行以下命令验证：

```
$ oc get nodes
```

输**出示例**

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

注意

批准服务器 CSR 后可能需要几分钟时间让机器转换为 **Ready** 状态。

其他信息

- 如需有关 CSR 的更多信息，请参阅证书签名请求。

7.1.7. Ansible hosts 文件的必要参数

在将 Red Hat Enterprise Linux (RHEL) 计算机器添加到集群之前，必须在 Ansible hosts 文件中定义以下参数。

参数	描述	值
ansible_user	能够以免密码方式进行 SSH 身份验证的 SSH 用户。如果使用基于 SSH 密钥的身份验证，则必须使用 SSH 代理来管理密钥。	系统上的用户名。默认值为 **root**。
ansible_become	如果 **ansible_user** 的值不是 root，您必须将 **ansible_become** 设置为 **True**，并且您指定为 **ansible_user** 的用户必须配置有免密码 sudo 访问权限。	**True**。如果值不是 **True**，请不要指定和定义此参数。
openshift_kubeconfig_path	指定包含集群的 **kubeconfig** 文件的本地目录的路径和文件名。	配置文件的路径和名称。

7.1.7.1. 可选：从集群中删除 RHCOS 计算机器

将 Red Hat Enterprise Linux（RHEL）计算机器添加到集群后，您可以选择性地删除 Red Hat Enterprise Linux CoreOS（RHCOS）计算机器来释放资源。

先决条件

- 您已将 RHEL 计算机器添加到集群中。

流程

1. 查看机器列表并记录 RHCOS 计算机器的节点名称：

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

2. 对于每一台 RHCOS 计算机器，删除其节点：
   a. 通过运行 **oc adm cordon** 命令，将节点标记为不可调度：

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

   1 指定其中一台 RHCOS 计算机器的节点名称。

   b. 清空节点中的所有 Pod：

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

   1 指定您隔离的 RHCOS 计算机器的节点名称。

   c. 删除节点：

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

   1 指定您清空的 RHCOS 计算机器的节点名称。

3. 查看计算机器的列表，以确保仅保留 RHEL 节点：

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

4. 从集群的计算机器的负载均衡器中删除 RHCOS 机器。您可以删除虚拟机或重新制作 RHCOS 计算机器物理硬件的镜像。

7.2. 在 OPENSHIFT CONTAINER PLATFORM 集群中添加更多 RHEL 计算机器

如果 OpenShift Container Platform 集群中已经包含 Red Hat Enterprise Linux (RHEL) 计算机器（也称为 worker），您可以向其中添加更多 RHEL 计算机器。

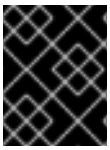
7.2.1. 关于在集群中添加 RHEL 计算节点

在 OpenShift Container Platform 4.4 中，如果使用用户置备的基础架构安装，您可以选择将 Red Hat Enterprise Linux (RHEL) 机器用作集群中的计算机器（也称为 worker）。集群中的 control plane 或主控机器（master）必须使用 Red Hat Enterprise Linux CoreOS (RHCOS)。

与所有使用用户置备的基础架构的安装一样，如果选择在集群中使用 RHEL 计算机器，您将需要负责所有操作系统生命周期的管理和维护任务，包括执行系统更新、应用补丁以及完成所有其他必要的任务。

重要

由于从集群中的机器上删除 OpenShift Container Platform 需要破坏操作系统，因此您必须对添加到集群中的所有 RHEL 机器使用专用的硬件。



重要

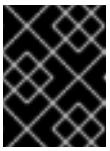
添加到 OpenShift Container Platform 集群的所有 RHEL 机器上都禁用内存交换功能。您无法在这些机器上启用交换内存。

您必须在初始化 control plane 之后将所有 RHEL 计算机器添加到集群。

7.2.2. RHEL 计算节点的系统要求

OpenShift Container Platform 环境中的 Red Hat Enterprise Linux (RHEL) 计算机器（也称为 worker）主机必须满足以下最低硬件规格和系统级别要求。

- 您的红帽帐户必须具有有效的 OpenShift Container Platform 订阅。如果没有，请与您的销售代表联系以了解更多信息。
- 生产环境必须提供能支持您的预期工作负载的计算机器。作为集群管理员，您必须计算预期的工作负载，再加上大约 10% 的开销。对于生产环境，请分配足够的资源，以防止节点主机故障影响您的最大容量。
- 所有系统都必须满足以下硬件要求：
  - 物理或虚拟系统，或在公有或私有 IaaS 上运行的实例。
  - 基础操作系统：使用 "Minimal" 安装选项的 RHEL 7.6-7.8。

    重要

    OpenShift Container Platform 4.4 仅支持 RHEL 7.6-7.8。您不能将计算机器升级到 RHEL 8。

  - 如果以 FIPS 模式部署 OpenShift Container Platform，则需要在 RHEL 机器上启用 FIPS，然后才能引导它。请参阅 RHEL 7 文档中的启用 FIPS 模式 。
  - NetworkManager 1.0 或更高版本。
  - 1 个 vCPU。
  - 最小 8 GB RAM。
  - 最小15 GB 硬盘空间，用于包含 **/var/** 的文件系统。
  - 最小1 GB 硬盘空间，用于包含 **/usr/local/bin/** 的文件系统。

- 最小1 GB 硬盘空间，用于包含系统临时目录的文件系统。系统的临时目录根据 Python 标准库中 tempfile 模块中定义的规则确定。

- 每个系统都必须满足您的系统提供商的任何其他要求。例如，如果在 VMware vSphere 上安装了集群，必须根据其存储准则配置磁盘，而且必须设置 **disk.enableUUID=true** 属性。

- 每个系统都必须能够使用可解析的主机名访问集群的 API 端点。任何现有的网络安全访问控制都必须允许系统访问集群的 API 服务端点。

7.2.2.1. 证书签名请求管理

在使用您置备的基础架构时，集群只能有限地访问自动机器管理，因此您必须提供一种在安装后批准集群证书签名请求 (CSR) 的机制。**kube-controller-manager** 只能批准 kubelet 客户端 CSR。 **machine-approver** 无法保证使用 kubelet 凭证请求的提供证书的有效性，因为它不能确认是正确的机器发出了该请求。您必须决定并实施一种方法，以验证 kubelet 提供证书请求的有效性并进行批准。

7.2.3. 准备 RHEL 计算节点

在将 Red Hat Enterprise Linux (RHEL) 机器添加到 OpenShift Container Platform 集群之前，您必须将每台主机注册到 Red Hat Subscription Manager (RHSM)，为其附加有效的 OpenShift Container Platform 订阅，并且启用所需的存储库。

1. 在每一主机上进行 RHSM 注册：

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

2. 从 RHSM 获取最新的订阅数据：

```
# subscription-manager refresh
```

3. 列出可用的订阅：

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

4. 在上一命令的输出中，找到 OpenShift Container Platform 订阅的池 ID 并附加该池：

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

5. 禁用所有 yum 存储库：

   a. 禁用所有已启用的 RHSM 存储库：

   ```
   # subscription-manager repos --disable="*"
   ```

   b. 列出剩余的 yum 存储库，并记录它们在 **repo id** 下的名称（若有）：

   ```
   # yum repolist
   ```

   c. 使用 **yum-config-manager** 禁用剩余的 yum 存储库：

   ```
   # yum-config-manager --disable <repo_id>
   ```

   或者，禁用所有存储库：

```
    yum-config-manager --disable \*
```

请注意，有大量可用存储库时可能需要花费几分钟

6. 仅启用 OpenShift Container Platform 4.4 需要的存储库：

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

7. 停止并禁用主机上的防火墙：

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

注意

请不要在以后启用防火墙。如果这样做，则无法访问 worker 上的 OpenShift Container Platform 日志。

7.2.4. 在集群中添加更多 RHEL 计算机器

您可以将使用 Red Hat Enterprise Linux（RHEL）作为操作系统的更多计算机器添加到 OpenShift Container Platform 4.4 集群中。

先决条件

- 您的 OpenShift Container Platform 集群已包含 RHEL 计算节点。
- 用于运行 playbook 的机器上具有您将第一台 RHEL 计算机器添加到集群时使用的 **hosts** 文件。
- 运行 playbook 的机器必须能够访问所有 RHEL 主机。您可以使用公司允许的任何方法，包括使用 SSH 代理或 VPN 的堡垒主机。
- 运行 playbook 的机器上具有集群的 **kubeconfig** 文件，以及用于安装集群的安装程序。
- 您必须对 RHEL 主机进行安装准备。
- 在运行 playbook 的机器上配置一个用户，该用户对所有 RHEL 主机具有 SSH 访问权限。
- 如果使用基于 SSH 密钥的身份验证，您必须使用 SSH 代理来管理密钥。
- 在运行 playbook 的机器上安装 OpenShift CLI（**oc**）。

流程

1. 打开位于 **/<path>/inventory/hosts** 的 Ansible 清单文件，该文件定义您的计算机器主机和必要的变量。
2. 将文件的 **[new_workers]** 部分重命名为 **[workers]**。
3. 在文件中添加 **[new_workers]** 部分，并且定义每个新主机的完全限定域名。该文件类似于以下示例：

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

openshift_kubeconfig_path="~/.kube/config"

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

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

在本示例中，**mycluster-rhel7-0.example.com** 和 **mycluster-rhel7-1.example.com** 机器已在集群中嗯，您需要添加 **mycluster-rhel7-2.example.com** 和 **mycluster-rhel7-3.example.com** 机器。

4. 运行扩展 playbook：

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

1 对于**<path>**，指定您创建的Ansible库存文件的路径。

7.2.5. 批准机器的证书签名请求

将机器添加到集群时，会为您添加的每台机器生成两个待处理证书签名请求（CSR）。您必须确认这些 CSR 已获得批准，或根据需要自行批准。客户端请求必须首先被批准，然后是服务器请求。

先决条件

- 您已将机器添加到集群中。

流程

1. 确认集群可以识别这些机器：

```
# oc get nodes

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

输出将列出您创建的所有机器。

2. 检查待处理的 CSR，并确保可以看到添加到集群中的每台机器都有 **Pending** 或 **Approved** 状态的客户端请求：

```
$ oc get csr
```

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

在本例中，两台机器加入了集群。您可能会在列表中看到更多已批准的 CSR。

3. 如果 CSR 没有获得批准，请在所添加机器的所有待处理 CSR 都处于 **Pending** 状态后，为您的集群机器批准这些 CSR：

注意

由于 CSR 会自动轮转，因此请在将机器添加到集群后一小时内批准您的 CSR。如果没有在一小时内批准，证书将会轮转，每个节点将会存在多个证书。您必须批准所有这些证书。批准初始 CSR 后，集群的 **kube-controller-manager** 会自动批准后续的节点客户端 CSR。您必须实施一个方法来自动批准 kubelet 提供的证书请求。

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

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

4. 现在，您的客户端请求已被批准，您必须查看添加到集群中的每台机器的服务器请求：

```
$ oc get csr
```

输**出示例**

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

5. 如果剩余的 CSR 没有被批准，且处于 **Pending** 状态，请批准集群机器的 CSR：

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

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

6. 批准所有客户端和服务器 CSR 后，器将处于 **Ready** 状态。运行以下命令验证：

```
$ oc get nodes
```

输**出示例**

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

注意

批准服务器 CSR 后可能需要几分钟时间让机器转换为 **Ready** 状态。

其他信息

- 如需有关 CSR 的更多信息，请参阅证书签名请求。

7.2.6. Ansible hosts 文件的必要参数

在将 Red Hat Enterprise Linux (RHEL) 计算机器添加到集群之前，必须在 Ansible hosts 文件中定义以下参数。

参数	描述	值
ansible_user	能够以免密码方式进行 SSH 身份验证的 SSH 用户。如果使用基于 SSH 密钥的身份验证，则必须使用 SSH 代理来管理密钥。	系统上的用户名。默认值为 **root**。
ansible_become	如果 **ansible_user** 的值不是 root，您必须将 **ansible_become** 设置为 **True**，并且您指定为 **ansible_user** 的用户必须配置有免密码 sudo 访问权限。	**True**。如果值不是 **True**，请不要指定和定义此参数。
openshift_kubeconfig_path	指定包含集群的 **kubeconfig** 文件的本地目录的路径和文件名。	配置文件的路径和名称。

7.3. 将计算机器添加到 VSPHERE

您可以将更多计算机器添加到 VMware vSphere 上的 OpenShift Container Platform 集群中。

7.3.1. 先决条件

- 您在 vSphere 上安装了集群。

7.3.2. 在 vSphere 中创建更多 Red Hat Enterprise Linux CoreOS（RHCOS）机器

您可以为集群创建更多计算机器，在 VMware vSphere 上使用用户置备的基础架构。

先决条件

- 获取计算机器的 Base64 编码 Ignition 文件。
- 您可以访问您为集群创建的 vSphere 模板。

流程

1. 部署模板后，为集群中的机器部署虚拟机。
   a. 右键点击模板的名称，再点击 **Clone → Clone to Virtual Machine**。
   b. 在 **Select a name and folder**选项卡中，指定虚拟机的名称。您可以在名称中包含机器类型，如 **compute-1**。
   c. 在 **Select a name and folder**选项卡中，选择您为集群创建的文件夹名称。
   d. 在 **Select a compute resource** 选项卡中，选择数据中心中的主机名称。
   e. 可选：在 **Select storage** 选项卡中，自定义存储选项。
   f. 在 **Select clone options** 中，选择 **Customize this virtual machine's hardware**。
   g. 在 **Customize hardware** 选项卡中，点击 **VM Options → Advanced**。
      - 从 **Latency Sensitivity** 列表中选择 **High**。
      - 点击 **Edit Configuration**，然后在 **Configuration Parameters** 窗口中点击 **Add Configuration Params**。定义以下参数名称和值：
        - **guestinfo.ignition.config.data**：粘贴此机器类型的 Base64 编码计算 Ignition 配置文件的内容。
        - **guestinfo.ignition.config.data.encoding**：指定 **base64**。
        - **disk.EnableUUID**：指定 **TRUE**。
   h. 在 **Customize hardware** 选项卡的 **Virtual Hardware** 面板中，根据需要修改指定的值。确保 RAM、CPU 和磁盘存储的数量满足机器类型的最低要求。另外，如果有多个可用的网络，请确定在 **Add network adapter 中** 选择正确的网络。
   i. 完成配置并打开虚拟机电源。
2. 继续为集群创建更多计算机器。

7.3.3. 批准机器的证书签名请求

将机器添加到集群时，会为您添加的每台机器生成两个待处理证书签名请求（CSR）。您必须确认这些 CSR 已获得批准，或根据需要自行批准。客户端请求必须首先被批准，然后是服务器请求。

先决条件

- 您已将机器添加到集群中。

流程

1. 确认集群可以识别这些机器：

```
# oc get nodes

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

   输出将列出您创建的所有机器。

2. 检查待处理的 CSR，并确保可以看到添加到集群中的每台机器都有 **Pending** 或 **Approved** 状态的客户端请求：

```
$ oc get csr

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

   在本例中，两台机器加入了集群。您可能会在列表中看到更多已批准的 CSR。

3. 如果 CSR 没有获得批准，请在所添加机器的所有待处理 CSR 都处于 **Pending** 状态后，为您的集群机器批准这些 CSR：

注意

由于 CSR 会自动轮转，因此请在将机器添加到集群后一小时内批准您的 CSR。如果没有在一小时内批准，证书将会轮转，每个节点将会存在多个证书。您必须批准所有这些证书。批准初始 CSR 后，集群的 **kube-controller-manager** 会自动批准后续的节点客户端 CSR。您必须实施一个方法来自动批准 kubelet 提供的证书请求。

   - 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

     1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

   - 要批准所有待处理的 CSR，请运行以下命令：

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

4. 现在，您的客户端请求已被批准，您必须查看添加到集群中的每台机器的服务器请求：

```
$ oc get csr
```

输**出示例**

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

5. 如果剩余的 CSR 没有被批准，且处于 **Pending** 状态，请批准集群机器的 CSR：

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

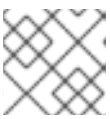
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. 批准所有客户端和服务器 CSR 后，器将处于 **Ready** 状态。运行以下命令验证：

```
$ oc get nodes
```

输**出示例**

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



注意

批准服务器 CSR 后可能需要几分钟时间让机器转换为 **Ready** 状态。

其他信息

- 如需有关 CSR 的更多信息，请参阅证书签名请求。

7.4. 在裸机中添加计算机器

您可以在裸机上的 OpenShift Container Platform 集群中添加更多计算机器。

7.4.1. 先决条件

- 您在裸机上安装了集群。
- 您有用来创建集群的安装介质和 Red Hat Enterprise Linux CoreOS（RHCOS）镜像。如果您没有这些文件，必须按照 安装过程的说明获得这些文件。

7.4.2. 创建 Red Hat Enterprise Linux CoreOS (RHCOS) 机器

在将更多计算机器添加到在裸机基础架构上安装的集群之前，必须先创建 RHCOS 机器供其使用。按照相应的步骤，使用 ISO 镜像或网络 PXE 启动来创建机器。

7.4.2.1. 使用 ISO 镜像创建更多 Red Hat Enterprise Linux CoreOS（RHCOS）机器

您可以使用 ISO 镜像为裸机集群创建更多计算机器，以创建机器。

先决条件

- 获取集群计算机器的 Ignition 配置文件的 URL。在安装过程中将该文件上传到 HTTP 服务器。
- 获取您在集群安装过程中上传到 HTTP 服务器的 BIOS 或 UEFI RHCOS 镜像文件的 URL。

流程

1. 使用 ISO 文件在更多计算机器上安装 RHCOS。在安装集群前，使用创建机器时使用的相同方法：
   - 将 ISO 镜像刻录到磁盘并直接启动。
   - 在 LOM 接口中使用 ISO 重定向。
2. 实例启动后，按 **TAB** 或 **E** 键编辑内核命令行。
3. 将参数添加到内核命令行：

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

(1) 指定要安装到的系统块设备。

(2) 指定您上传到服务器的 UEFI 或 BIOS 镜像的 URL。

(3) 指定计算 Ignition 配置文件的 URL。

4. 按 **Enter** 键完成安装。安装 RHCOS 后，系统会重启。系统重启后，它会应用您指定的 Ignition 配置文件。
5. 继续为集群创建更多计算机器。

7.4.2.2. 通过 PXE 或 iPXE 启动来创建 Red Hat Enterprise Linux CoreOS (RHCOS) 机器

您可以使用 PXE 或 iPXE 引导为裸机集群创建更多计算机器。

先决条件

- 获取集群计算机器的 Ignition 配置文件的 URL。在安装过程中将该文件上传到 HTTP 服务器。
- 获取您在集群安装过程中上传到 HTTP 服务器的 RHCOS ISO 镜像、压缩的裸机 BIOS、**kernel** 和 **initramfs** 文件的 URL。
- 您可以访问在安装过程中为 OpenShift Container Platform 集群创建机器时使用的 PXE 引导基础架构。机器必须在安装 RHCOS 后从本地磁盘启动。
- 如果使用 UEFI，您可以访问在 OpenShift Container Platform 安装过程中修改的 **grub.conf** 文件。

流程

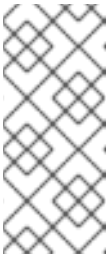
1. 确认 RHCOS 镜像的 PXE 或 iPXE 安装正确。
   - 对于 PXE：

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

(1) 指定上传到 HTTP 服务器的 **kernel** 文件位置。

(2) 如果您使用多个 NIC，请在 **ip** 选项中指定一个接口。例如，要在名为 **eno1** 的 NIC 上使用 DHCP，请设置 **ip=eno1:dhcp**。

(3) 指定上传到 HTTP 服务器的 RHCOS 文件的位置。**initrd** 参数值是 **initramfs** 文件的位置，**coreos.inst.image_url** 参数值是压缩的裸机 RAW 镜像的位置，**coreos.inst.ignition_url** 参数值则是 worker Ignition 配置文件的位置。



注意

这个配置不会在使用图形控制台的机器上启用串口控制台访问。要配置不同的控制台，请在 **APPEND** 行中添加一个或多个 **console=** 参数。例如，添加 **console=tty0 console=ttyS0** 将第一个 PC 串口设置为主控制台，图形控制台作为二级控制台。如需更多信息，请参阅如何在 Red Hat Enterprise Linux 中设置串行终端和（或）控制台？

   - 对于 iPXE：

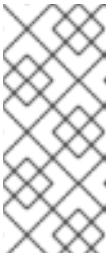
```
kernel http://<HTTP_server>/rhcos-<version>-installer-kernel-<architecture> ip=dhcp
rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-installer-initramfs.
```

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

(1) 指定上传到 HTTP 服务器的 RHCOS 文件的位置。**kernel** 参数值是 **kernel** 文件的位置，**initrd** 参数值是 **initramfs** 文件的位置，**coreos.inst.image_url** 参数值是压缩的裸机 RAW 镜像的位置，**coreos.inst.ignition_url** 参数值则是 worker Ignition 配置文件的位置。

(2) 如果您使用多个 NIC，请在 **ip** 选项中指定一个接口。例如，要在名为 **eno1** 的 NIC 上使用 DHCP，请设置 **ip=eno1:dhcp**。

(3) 指定上传到 HTTP 服务器的 **initramfs** 文件的位置。



注意

这个配置不会在使用图形控制台的机器上启用串口控制台访问。要配置不同的控制台，请在 **kerne** 行中添加一个或多个 **console=** 参数。例如，添加 **console=tty0 console=ttyS0** 将第一个 PC 串口设置为主控制台，图形控制台作为二级控制台。如需更多信息，请参阅如何在 Red Hat Enterprise Linux 中设置串行终端和（或）控制台？

2. 使用 PXE 或 iPXE 基础架构为集群创建所需的计算机器。

7.4.3. 批准机器的证书签名请求

将机器添加到集群时，会为您添加的每台机器生成两个待处理证书签名请求（CSR）。您必须确认这些 CSR 已获得批准，或根据需要自行批准。客户端请求必须首先被批准，然后是服务器请求。

先决条件

- 您已将机器添加到集群中。

流程

1. 确认集群可以识别这些机器：

```
# oc get nodes

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

输出将列出您创建的所有机器。

2. 检查待处理的 CSR，并确保可以看到添加到集群中的每台机器都有 **Pending** 或 **Approved** 状态的客户端请求：

```
$ oc get csr

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

在本例中，两台机器加入了集群。您可能会在列表中看到更多已批准的 CSR。

3. 如果 CSR 没有获得批准，请在所添加机器的所有待处理 CSR 都处于 **Pending** 状态后，为您的集群机器批准这些 CSR：

注意

由于 CSR 会自动轮转，因此请在将机器添加到集群后一小时内批准您的 CSR。如果没有在一小时内批准，证书将会轮转，每个节点将会存在多个证书。您必须批准所有这些证书。批准初始 CSR 后，集群的 **kube-controller-manager** 会自动批准后续的节点客户端 CSR。您必须实施一个方法来自动批准 kubelet 提供的证书请求。

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

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

4. 现在，您的客户端请求已被批准，您必须查看添加到集群中的每台机器的服务器请求：

```
$ oc get csr
```

输出示例

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

5. 如果剩余的 CSR 没有被批准，且处于 **Pending** 状态，请批准集群机器的 CSR：

- 若要单独批准，请对每个有效的 CSR 运行以下命令：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

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

6. 批准所有客户端和服务器 CSR 后，器将处于 **Ready** 状态。运行以下命令验证：

```
$ oc get nodes
```

输**出示例**

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

注意

批准服务器 CSR 后可能需要几分钟时间让机器转换为 **Ready** 状态。

其他信息

- 如需有关 CSR 的更多信息，请参阅证书签名请求。

第 8 章 部署机器健康检查

您可以配置和部署机器健康检查，以自动修复机器池中损坏的机器。

重要

此过程不适用于自己手动置备机器的集群。您只能在使用机器 API 的集群中使用高级机器管理和扩展功能。

8.1. 关于机器健康检查

您可以使用 **MachineHealthCheck** 资源定义集群中的机器被视为不健康的条件。会自动修复满足条件的机器。

要监控机器健康状况，创建一个 **MachineHealthCheck** 自定义资源（CR），其中包含要监控的机器集合的标签以及要检查的条件，如维持 **NotReady** 状态 15 分钟，或在 node-problem-detector 中显示持久性状况。

监控 **MachineHealthCheck** CR 的控制器会检查您定义的条件。如果机器无法进行健康检查，则会自动删除机器并创建新的机器来代替它。删除机器之后，您会看到**机器被删除**事件。

注意

对于具有 master 角色的机器，机器健康检查会报告不健康的节点数量，但不会删除机器。例如：

输**出示例**

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

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

为限制删除机器造成的破坏性影响，控制器一次仅排空并删除一个节点。如果目标机器池中不健康的机器池中不健康的机器数量大于 **maxUnhealthy** 的值，则控制器会停止删除机器，您必须手动进行处理。

要停止检查，请删除自定义资源。

8.1.1. 部署机器健康检查时的限制

部署机器健康检查前需要考虑以下限制：

- 只有机器集拥有的机器才可以由机器健康检查修复。
- 目前不支持 control plane 机器，如果不健康，则不会被修复。
- 如果机器的节点从集群中移除，机器健康检查会认为机器不健康，并立即修复机器。
- 如果机器对应的节点在 **nodeStartupTimeout** 之后没有加入集群，则会修复机器。
- 如果 **Machine** 资源阶段为 **Failed**，则会立即修复机器。

其他资源

- 如需了解更多与短电路相关的信息，请参阅 Short-circuiting 机器健康检查补救

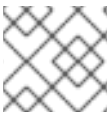
8.2. MACHINEHEALTHCHECK 资源示例

MachineHealthCheck 资源类似以下 YAML 文件：

MachineHealthCheck

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

1. 指定要部署的机器健康检查的名称。
2. 3. 为要检查的机器池指定一个标签。
4. 以 **<cluster_name>-<label>-<zone>** 格式 指定要跟踪的机器集。例如，**prod-node-us-east-1a**。
5. 6. 指定节点条件的超时持续时间。如果在超时时间内满足了条件，则会修复机器。超时时间较长可能会导致不健康的机器上的工作负载长时间停机。
7. 指定目标池中允许的不健康机器的数量。这可设为一个百分比或一个整数。
8. 指定机器健康检查在决定机器不健康前必须等待节点加入集群的超时持续时间。



注意

matchLabels 只是示例; 您必须根据具体需要映射您的机器组。

8.2.1. 短路机器健康检查补救

短路可确保仅在集群健康时机器健康检查修复机器。通过 **MachineHealthCheck** 资源中的 **maxUnhealthy** 字段配置短路。

如果用户在修复任何机器前为 **maxUnhealthy** 字段定义了一个值，**MachineHealthCheck** 会将 **maxUnhealthy** 的值与它决定不健康的目标池中的机器数量进行比较。如果不健康的机器数量超过 **maxUnhealthy** 限制，则不会执行补救。

重要

如果没有设置 **maxUnhealthy**，则默认值为 **100%**，无论集群状态如何，机器都会被修复。

maxUnhealthy 字段可以设置为整数或百分比。根据 **maxUnhealthy** 值，有不同的补救实现。

8.2.1.1. 使用绝对值设置 maxUnhealthy

如果将 **maxUnhealthy** 设为 **2**:

- 如果 2 个或更少节点不健康，则可执行补救
- 如果 3 个或更多节点不健康，则不会执行补救

这些值与机器健康检查要检查的机器数量无关。

8.2.1.2. 使用百分比设置 maxUnhealthy

如果 **maxUnhealthy** 被设置为 **40%**，有 25 个机器被检查：

- 如果有 10 个或更少节点处于不健康状态，则可执行补救
- 如果 11 个或多个节点不健康，则不会执行补救

如果 **maxUnhealthy** 被设置为 **40%**，有 6 个机器被检查：

- 如果 2 个或更少节点不健康，则可执行补救
- 如果 3 个或更多节点不健康，则不会执行补救

注意

当被检查的 **maxUnhealthy** 机器的百分比不是一个整数时，允许的机器数量会被舍入到一个小的整数。

8.3. 创建 MACHINEHEALTHCHECK 资源

您可以为集群中的所有 **MachineSet** 创建 **MachineHealthCheck** 资源。您不应该创建针对 control plane 机器的 **MachineHealthCheck** 资源。

先决条件

- 安装 **oc** 命令行界面。

流程

1. 创建一个 **healthcheck.yml** 文件，其中包含您的机器健康检查的定义。
2. 将 **healthcheck.yml** 文件应用到您的集群：

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