OpenShift Container Platform 4.16

机器管理

添加和维护集群机器

Last Updated: 2024-07-02

OpenShift Container Platform 4.16 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 机器管理概述

您可以使用机器管理来灵活地处理底层基础架构，如 Amazon Web Services (AWS)、Microsoft Azure、Google Cloud Platform (GCP)、Red Hat OpenStack Platform (RHOSP)和 VMware vSphere 来管理 OpenShift Container Platform 集群。您可以控制集群并执行自动扩展，如根据特定工作负载策略扩展和缩减集群。

务必要让集群适应不断变化的工作负载。当负载增加或减少时，OpenShift Container Platform 集群可以水平扩展和缩减。

机器管理作为一个自定义资源定义 (CRD) 实施。CRD 对象在集群中定义了一个新的唯一对象 **Kind**，并允许 Kubernetes API 服务器处理对象的整个生命周期。

Machine API Operator 置备以下资源：

- **MachineSet**
- **机器**
- **ClusterAutoscaler**
- **MachineAutoscaler**
- **MachineHealthCheck**

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 4.16 集群，在集群安装完成后，Machine API 会执行所有节点主机置备管理操作。因此，OpenShift Container Platform 4.16 在公有或私有云基础架构之上提供了一种弹性动态置备的方法。

两种主要资源分别是：

Machines

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，计算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 资源是计算机器组。计算机器集适用于计算机器，因为副本集是针对 pod。如果需要更多计算机器或必须缩减规模，您可以更改 **MachineSet** 资源的 **replicas** 字段来满足您的计算需求。

警告

control plane 机器不能由计算机器集管理。

control plane 机器集为 control plane 机器提供管理功能，与为计算机器提供的计算机器集类似。

如需更多信息，请参阅"管理 control plane 机器"。

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计算机器。您可以为指定计算机器集中的节点设置最小和最大扩展界限，机器自动扩展则维护该范围内的节点。
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，它通过扩展计算机器设置 API 来与 Machine API 集成。您可以使用以下方法使用集群自动扩展来管理集群：

- 为内核、节点、内存和 GPU 等资源设置集群范围的扩展限制
- 设置优先级，以便集群对 pod 和新节点进行优先排序，而在不太重要的 pod 时不会上线
- 设置扩展策略，以便您可以扩展节点，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。每个计算机器集限定在一个区，因此安装程序可以代表您的可用区向计算机器集发送。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。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。自动扩展器在集群生命周期内尽可能提供平衡。

其他资源

- 机器阶段和生命周期

1.2. 管理计算机器

作为集群管理员，您可以执行以下操作：

- 为以下云供应商创建计算机器集：
  - Alibaba Cloud

    - AWS
    - Azure
    - Azure Stack Hub
    - GCP
    - IBM Cloud
    - IBM Power Virtual Server
    - Nutanix
    - RHOSP
    - vSphere
- 为裸机部署创建机器集：在裸机上创建计算机器集
- 通过从计算机器集中添加或删除机器，手动扩展计算机器。
- 通过 **MachineSet** YAML 配置文件修改计算机器设置。
- 删除机器。
- 创建基础架构计算机器集。
- 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。

1.3. 管理 CONTROL PLANE 机器

作为集群管理员，您可以执行以下操作：

- 使用以下云供应商的 control plane 机器集更新 control plane 配置 ：
    - Amazon Web Services
    - Google Cloud Platform
    - Microsoft Azure
    - Nutanix
    - Red Hat OpenStack Platform(RHOSP)
    - VMware vSphere
- 配置和部署 机器健康检查，以自动恢复不健康的 control plane 机器。

1.4. 将自动扩展应用到 OPENSHIFT CONTAINER PLATFORM 集群

您可以自动扩展 OpenShift Container Platform 集群，以确保更改工作负载的灵活性。要 自动扩展 您的集群，必须首先部署集群自动扩展，然后为每个计算机器集部署机器自动扩展。

- *集群自动扩展*会根据部署需求增加和缩小集群的大小。

- *机器自动扩展*会调整您在 OpenShift Container Platform 集群中部署的计算机器集中的机器数量。

1.5. 在用户置备的基础架构上添加计算机器

用户置备的基础架构是一个环境，您可以在其中部署托管 OpenShift Container Platform 的计算、网络和存储资源等基础架构。您可以在安装过程中或安装后，将计算机器添加到用户置备的基础架构上的集群。

1.6. 在集群中添加 RHEL 计算机器

作为集群管理员，您可以执行以下操作：

- 将 Red Hat Enterprise Linux(RHEL)计算机器（也称为 worker 机器）添加到 用户置备的基础架构集群或安装置备的基础架构集群中。
- 将更多 Red Hat Enterprise Linux(RHEL)计算机器添加到 现有集群中。

第 2 章 使用 MACHINE API 管理计算机器

2.1. 在 ALIBABA CLOUD 上创建计算机器设置

您可以在 Alibaba Cloud 上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.1.1. Alibaba Cloud 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在区域中指定的 Alibaba Cloud 区域中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<zone> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 10
    spec:
      metadata:
        labels:
```

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

1 5 7 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（**oc**）软件包，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 指定要添加的节点标签。

4 6 10 指定基础架构 ID、节点标签和区域。

11 指定要使用的镜像。使用集群的现有默认计算机器集中的镜像。

12 指定用于计算机器集的实例类型。

13 指定要用于计算机器设置的 RAM 角色的名称。使用安装程序在默认计算机器集中填充的值。

14 指定要放置机器的区域。

15 指定集群的资源组和类型。您可以使用安装程序在默认计算机器集中填充的值，或者指定不同的值。

16 18 20 指定要用于计算机器设置的标签。最少，您必须包含本例中显示的标签，以及适当的集群值。您可以包括额外的标签，包括安装程序根据需要在默认的计算机器集中填充的标签。

17 指定根磁盘的类型和大小。使用安装程序在其创建的默认计算机器集中填充的 **category** 值。如果需要，以 GB 为单位指定**大小**的不同值。

19 指定 **openshift-machine-api** 命名空间中的用户数据 YAML 文件中的 secret 名称。使用安装程序在默认计算机器集中填充的值。

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

2.1.1.1. Alibaba Cloud 使用统计的机器集参数

安装程序为 Alibaba Cloud 集群创建的默认计算机器集包括 Alibaba Cloud 用来跟踪用量统计的 nonessential 标签值。这些标签在 **spec.template.spec.providerSpec.value** 列表的 **securityGroups**、**tag** 和 **vSwitch** 参数中填充。

在创建部署额外机器的计算机器集时，必须包含所需的 Kubernetes 标签。使用统计标签默认应用，即使它们没有在您创建的计算机器集中指定。您还可以根据需要包含其他标签。

以下 YAML 片段表示默认计算机器集中哪些标签是可选的，且需要哪些标签。

spec.template.spec.providerSpec.value.securityGroups 中的标签

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

1 2 可选：即使计算机器集中没有指定，也会应用该标签。

3 必需。

其中：

- **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。
- **<role>** 是要添加的节点标签。

spec.template.spec.providerSpec.value.tag 中的标签

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

2 3 可选：即使计算机器集中没有指定，也会应用该标签。

1 必需。

其中 **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。

spec.template.spec.providerSpec.value.vSwitch中的标签

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

1 2 3 可选：即使计算机器集中没有指定，也会应用该标签。

4 必需。

其中：

- **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。
- **<zone>** 是要放置机器的区域里的区。

2.1.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

输**出示例**

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输**出示例**

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

```
    machine.openshift.io/cluster-api-machine-role: <role>
    machine.openshift.io/cluster-api-machine-type: <role>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
spec:
  providerSpec: 3
    ...
```

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输**出示例**

```
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
```

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

2.2. 在 AWS 上创建计算机器集

您可以在 Amazon Web Services (AWS)上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.2.1. AWS 上计算机器设置自定义资源的 YAML 示例

YAML 示例定义了一个在 **us-east-1a** Amazon Web Services (AWS) Local Zone 中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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
  name: <infrastructure_id>-<role>-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <role> 6
        machine.openshift.io/cluster-api-machine-type: <role> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone> 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 9
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 10
          apiVersion: machine.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
```

```
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile 11
      instanceType: m6i.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: <zone> 12
        region: <region> 13
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg 14
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-<zone> 15
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> 16
          value: owned
        - name: <custom_tag_name> 17
          value: <custom_tag_value> 18
      userDataSecret:
        name: worker-user-data
```

1 3 5 11 14 16 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
```

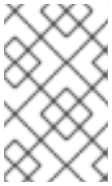
2 4 8 指定基础架构 ID、角色节点标签和区域。

6 7 9 指定要添加的角色节点标签。

10 为 OpenShift Container Platform 节点的 AWS 区域指定有效的 Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI)。如果要使用 AWS Marketplace 镜像，则必须从 AWS Marketplace 完成 OpenShift Container Platform 订阅来获取您所在地区的 AMI ID。

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

17 18 可选：为集群指定自定义标签数据。例如，您可以通过指定 **Email:admin-email@example.com** 的 **name:value** 对来添加管理员的电子邮件地址。



注意

也可以在 **install-config.yml** 文件中在安装过程中指定自定义标签。如果 **install-config.yml** 文件和机器集包含具有相同 **name** 数据的标签，则机器集的标签值优先于 **install-config.yml** 文件中的标签值。

12 指定区域，如 **us-east-1a**。

13 指定区域，如 **us-east-1**。

15 指定基础架构 ID 和区域。

2.2.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

输**出示例**

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输**出示例**

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

```
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
        ...
```

1 集群基础架构 ID。

2 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

4. 如果需要其他可用区中的计算机器集，请重复此过程来创建更多计算机器集。

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

-

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

2.2.3. 使用机器集，为 Elastic Fabric Adapter 实例分配机器到放置组

您可以配置机器集，在现有 AWS PG 中的 Elastic Fabric Adapter (EFA) 实例上部署机器。

EFA 实例不需要放置组，您可以使用放置组作为配置 EFA 以外的目的。本例使用这两者来演示能够提高指定放置组内机器的网络性能的配置。

先决条件

- 您在 AWS 控制台中创建了放置组。

注意

确保您创建的放置组类型的规则和限制与预期的用例兼容。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 编辑 **providerSpec** 字段中的以下行：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          instanceType: <supported_instance_type> 1
          networkInterfaceType: EFA 2
          placement:
            availabilityZone: <zone> 3
            region: <region> 4
          placementGroupName: <placement_group> 5
# ...
```

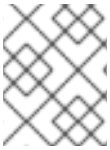
1. 指定支持 EFA 的实例类型。
2. 指定 **EFA** 网络接口类型。
3. 指定区域，如 **us-east-1a**。
4. 指定区域，如 **us-east-1**。
5. 指定要部署机器的现有 AWS PG 的名称。

验证

- 在 AWS 控制台中，找到机器集创建的机器，并在机器属性中验证以下内容：
  - placement group 字段具有您为机器集中的 **placementGroupName** 参数指定的值。
  - 接口类型字段表示它使用 EFA。

2.2.4. Amazon EC2 实例元数据服务的机器集选项

您可以使用机器集创建使用 Amazon EC2 实例元数据服务 (IMDS) 的特定版本的机器。机器集可以创建允许使用 IMDSv1 和 IMDSv2 的机器或需要使用 IMDSv2 的机器。



注意

只有在 OpenShift Container Platform 版本 4.7 或更高版本中创建的 AWS 集群上才支持使用 IMDSv2。

要使用您首选的 IMDS 配置部署新计算机器，请使用适当的值创建计算机器设置 YAML 文件。在扩展机器集时，您还可以编辑现有的机器集，以使用首选 IMDS 配置创建新机器。

重要

在配置机器集来创建需要 IMDSv2 的机器前，请确保与 AWS 元数据服务交互的工作负载都支持 IMDSv2。

2.2.4.1. 使用机器集配置 IMDS

您可以通过在机器集 YAML 文件中添加或编辑 **metadataServiceOptions.authentication**，来指定是否需要使用 IMDSv2。

先决条件

- 要使用 IMDSv2，您的 AWS 集群必须使用 OpenShift Container Platform 版本 4.7 或更高版本创建。

流程

- 在 **providerSpec** 字段中添加或编辑以下行：

```
providerSpec:
  value:
    metadataServiceOptions:
      authentication: Required 1
```

1 为了要求 IMDSv2，请将参数值设置为 **Required**。要允许使用 IMDSv1 和 IMDSv2，请将参数值设置为 **Optional**。如果没有指定值，则允许 IMDSv1 和 IMDSv2。

2.2.5. 将机器部署为 Dedicated 实例的机器集

您可以创建在 AWS 上运行的机器集，该机器将机器部署为 Dedicated 实例。专用实例在专用于单一客户的硬件上运行虚拟私有云（VPC）。这些 Amazon EC2 实例在主机硬件级别被物理隔离。Dedicated 实例的隔离也会存在，即使实例属于链接到一个 Forer 帐户的不同 AWS 帐户。但是，其他未专用实例如果属于同一 AWS 帐户，则可以与 Dedicated 实例共享硬件。

Machine API 支持具有公共或专用租期的实例。具有公共租期的实例在共享硬件上运行。公共租期是默认租期。具有专用租期的实例在单租户硬件上运行。

2.2.5.1. 使用机器集创建 Dedicated 实例

您可以使用 Machine API 集成来运行由 Dedicated 实例支持的机器。设置机器设置 YAML 文件中的 **tenancy** 字段，以便在 AWS 上启动 Dedicated 实例。

流程

- 在 **providerSpec** 字段中指定专用租户：

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

2.2.6. 将机器部署为 Spot 实例的机器集

您可以通过创建一个在 AWS 上运行的计算机器集来节约成本，该机器集将机器部署为非保障的 Spot 实例。Spo 实例使用未使用的 AWS EC2 容量，且比按需（On-Demand）实例的成本要低。您可以将 Spot 实例用于可容许中断的工作负载，如批处理或无状态工作负载、横向可扩展工作负载。

AWS EC2 可随时终止 Spot 实例。当发生中断时，AWS 会向用户发出两分钟警告信息。当 AWS 发出终止警告时，OpenShift Container Platform 开始从受影响的实例中删除工作负载。

使用 Spot 实例时可能会因为以下原因造成中断：

- 实例价格超过您的最大价格
- Spot 实例的需求增加
- Spot 实例的提供减少

当 AWS 终止实例时，Spot 实例节点上运行的终止处理器会删除机器资源。为了满足计算机器设置 **副本数量**，计算机器会创建一个请求 Spot 实例的机器。

2.2.6.1. 使用计算机器集创建 Spot 实例

您可以通过在计算机器设置 YAML 文件中添加 **SpotMarketOptions**，在 AWS 上启动 Spot 实例。

流程

- 在 **providerSpec** 字段中添加以下行：

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

  您可以选择设置 **spotMarketOptions.maxPrice** 字段来限制 Spot 实例的成本。例如，您可以设置 **maxPrice: '2.50'**。

  如果设置了 **maxPrice**，则将此值用作每小时最大即时价格。如果没有设置，则默认使用最大价格收费，以达到按需处理的实例价格。

注意

强烈建议您使用默认的 On-Demand 价格作为 **maxPrice** 值，不要为 Spot 实例设置最大价格。

2.2.7. 将 GPU 节点添加到现有 OpenShift Container Platform 集群中

您可以复制并修改默认计算机器集配置，以便为 AWS EC2 云供应商创建启用了 GPU 的机器集和机器。

有关支持的实例类型的更多信息，请参阅以下 NVIDIA 文档：

- NVIDIA GPU Operator 社区支持列表
- NVIDIA AI Enterprise 支持列表

流程

1. 运行以下命令，查看现有节点、机器和机器集。请注意，每个节点都是带有特定 AWS 区域和 OpenShift Container Platform 角色的机器定义实例。

```
$ oc get nodes
```

输出示例

```
NAME                                        STATUS   ROLES                  AGE     VERSION
ip-10-0-52-50.us-east-2.compute.internal    Ready    worker                 3d17h   v1.29.4
ip-10-0-58-24.us-east-2.compute.internal    Ready    control-plane,master   3d17h   v1.29.4
ip-10-0-68-148.us-east-2.compute.internal   Ready    worker                 3d17h   v1.29.4
ip-10-0-68-68.us-east-2.compute.internal    Ready    control-plane,master   3d17h   v1.29.4
ip-10-0-72-170.us-east-2.compute.internal   Ready    control-plane,master   3d17h   v1.29.4
ip-10-0-74-50.us-east-2.compute.internal    Ready    worker                 3d17h   v1.29.4
```

2. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器和机器集。每个计算机器集都与 AWS 区域的不同可用区关联。安装程序会在可用区之间自动负载平衡计算机器。

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

输出示例

```
NAME                                  DESIRED   CURRENT   READY   AVAILABLE   AGE
preserve-dsoc12r4-ktjfc-worker-us-east-2a   1         1         1       1           3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b   2         2         2       2           3d11h
```

3. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器。目前，每个机器集只有一个计算机器，但可以扩展计算机器集，以便在特定地区和区域中添加节点。

```
$ oc get machines -n openshift-machine-api | grep worker
```

输出示例

```
preserve-dsoc12r4-ktjfc-worker-us-east-2a-dts8r     Running   m5.xlarge   us-east-2   us-east-2a   3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b-dkv7w     Running   m5.xlarge   us-east-2   us-
```

```
east-2b   3d11h
preserve-dsoc12r4-ktjfc-worker-us-east-2b-k58cw     Running   m5.xlarge   us-east-2   us-
east-2b   3d11h
```

4. 运行以下命令，复制现有计算 **MachineSet** 定义并将结果输出到 JSON 文件。这将是启用了 GPU 的计算机器集定义的基础。

```
$ oc get machineset preserve-dsoc12r4-ktjfc-worker-us-east-2a -n openshift-machine-api -o
json > <output_file.json>
```

5. 编辑 JSON 文件，并对新 **MachineSet** 定义进行以下更改：

   - 将 **worker** 替换为 **gpu**。这将是新计算机集的名称。
   - 将新 **MachineSet** 定义的实例类型更改为 **g4dn**，其中包括 NVIDIA Tesla T4 GPU。要了解更多有关 AWS **g4dn** 实例类型的信息，请参阅加速计算。

     ```
     $ jq .spec.template.spec.providerSpec.value.instanceType preserve-dsoc12r4-ktjfc-
     worker-gpu-us-east-2a.json

     "g4dn.xlarge"
     ```

     <output_file.json> 文件被保持为 **preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json**。

6. 更新 **preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json** 中的以下字段：

   - 将 **.metadata.name** 替换为一个包括 **gpu** 的名称。
   - **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]** 匹配新的 **.metadata.name**。
   - **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]** 以匹配新的 **.metadata.name**。
   - 将 **.spec.template.spec.providerSpec.value.instanceType** 替换为 **g4dn.xlarge**。

7. 要验证您的更改，请运行以下命令对原始计算定义和新的 GPU 节点定义执行 **diff** ：

```
$ oc -n openshift-machine-api get preserve-dsoc12r4-ktjfc-worker-us-east-2a -o json | diff
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json -
```

输出示例

```
10c10

< "name": "preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a",
---
> "name": "preserve-dsoc12r4-ktjfc-worker-us-east-2a",

21c21

< "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-gpu-us-
east-2a"
---
> "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-us-east-2a"
```

```
31c31

< "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a"
---
> "machine.openshift.io/cluster-api-machineset": "preserve-dsoc12r4-ktjfc-worker-us-east-2a"

60c60

< "instanceType": "g4dn.xlarge",
---
> "instanceType": "m5.xlarge",
```

8. 运行以下命令，从定义创建启用了 GPU 的计算机器集：

```
$ oc create -f preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a.json
```

输**出示例**

```
machineset.machine.openshift.io/preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a created
```

验证

1. 运行以下命令，查看您创建的机器集：

```
$ oc -n openshift-machine-api get machinesets | grep gpu
```

MachineSet 副本数被设置为 **1**，以便自动创建新的 **Machine** 对象。

输**出示例**

```
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a   1         1         1       1           4m21s
```

2. 运行以下命令，查看创建机器集的 **Machine** 对象：

```
$ oc -n openshift-machine-api get machines | grep gpu
```

输**出示例**

```
preserve-dsoc12r4-ktjfc-worker-gpu-us-east-2a   running   g4dn.xlarge   us-east-2   us-east-2a   4m36s
```

请注意，不需要为节点指定命名空间。节点定义是在集群范围之内。

2.2.8. 部署 Node Feature Discovery Operator

创建启用了 GPU 的节点后，您需要发现启用了 GPU 的节点，以便调度它。为此，请安装 Node Feature Discovery (NFD) Operator。NFD Operator 识别节点中的硬件设备功能。它解决了在基础架构节点中识别和目录硬件资源的一般问题，以便 OpenShift Container Platform 可以使用它们。

流程

1. 在 OpenShift Container Platform 控制台中，从 **OperatorHub** 安装 Node Feature Discovery Operator。
2. 将 NFD Operator 安装到 **OperatorHub** 后，从已安装的 Operator 列表中选择 **Node Feature Discovery**，然后选择 **Create instance**。这会在 **openshift-nfd** 命名空间中安装 **nfd-master** 和 **nfd-worker** pod，每个计算节点一个 **nfd-worker** pod。
3. 运行以下命令验证 Operator 是否已安装并正在运行：

```
$ oc get pods -n openshift-nfd
```

输**出示例**

```
NAME                                  READY   STATUS    RESTARTS   AGE

nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running 7  (8h ago)   1d
```

4. 浏览到控制台中的已安装的 Oerator，再选择 **Create Node Feature Discovery**。
5. 选择 **Create** 以构建 NFD 自定义资源。这会在 **openshift-nfd** 命名空间中创建 NFD pod，为硬件资源和目录轮询 OpenShift Container Platform 节点。

验证

1. 构建成功后，运行以下命令来验证 NFD pod 是否在每个节点上运行：

```
$ oc get pods -n openshift-nfd
```

输**出示例**

```
NAME                                  READY   STATUS      RESTARTS       AGE
nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running     7 (8h ago)     12d
nfd-master-769656c4cb-w9vrv               1/1     Running     0              12d
nfd-worker-qjxb2                      1/1     Running     3 (3d14h ago)  12d
nfd-worker-xtz9b                      1/1     Running     5 (3d14h ago)  12d
```

NFD Operator 使用厂商 PCI ID 来识别节点的硬件。NVIDIA 使用 PCI ID **10de**。

2. 运行以下命令，查看 NFD Operator 发现的 NVIDIA GPU：

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

输**出示例**

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

10de 会出现在启用了 GPU 的节点的节点功能列表中。这意味着 NFD Operator 可以正确地识别启用了 GPU 的 MachineSet 的节点。

2.3. 在 AZURE 上创建计算机器设置

您可以在 Microsoft Azure 上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.3.1. Azure 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在区域(region)的 **1** Microsoft Azure 区域(zone)中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role>
  name: <infrastructure_id>-<role>-<region> 3
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region>
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region>
    spec:
      metadata:
        creationTimestamp: null
```

```
      labels:
        machine.openshift.io/cluster-api-machineset: <machineset_name>
        node-role.kubernetes.io/<role>: ""
    providerSpec:
      value:
        apiVersion: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image: 4
          offer: ""
          publisher: ""
          resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/galleries/gallery_<infrastructure_id>/images/<infrastructure_id>-
gen2/versions/latest 5
          sku: ""
          version: ""
        internalLoadBalancer: ""
        kind: AzureMachineProviderSpec
        location: <region> 6
        managedIdentity: <infrastructure_id>-identity
        metadata:
          creationTimestamp: null
        natRule: null
        networkResourceGroup: ""
        osDisk:
          diskSizeGB: 128
          managedDisk:
            storageAccountType: Premium_LRS
          osType: Linux
        publicIP: false
        publicLoadBalancer: ""
        resourceGroup: <infrastructure_id>-rg
        sshPrivateKey: ""
        sshPublicKey: ""
        tags:
          - name: <custom_tag_name> 7
            value: <custom_tag_value>
        subnet: <infrastructure_id>-<role>-subnet
        userDataSecret:
          name: worker-user-data
        vmSize: Standard_D4s_v3
        vnet: <infrastructure_id>-vnet
        zone: "1" 8
```

1 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
```

您可以运行以下命令来获取子网：

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

您可以运行以下命令来获取 vnet：

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

2 指定要添加的节点标签。

3 指定基础架构 ID、节点标签和地区。

4 指定计算机器设置的镜像详情。如果要使用 Azure Marketplace 镜像，请参阅"选择 Azure Marketplace 镜像"。

5 指定与实例类型兼容的镜像。安装程序创建的 Hyper-V 生成 V2 镜像具有 **-gen2** 后缀，而 V1 镜像则与没有后缀的名称相同。

6 指定要放置机器的区域。

7 可选：在机器集中指定自定义标签。在 **<custom_tag_name>** 字段中提供标签名称，并在 **<custom_tag_value>** 字段中提供对应的标签值。

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

2.3.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

输**出示例**

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输出示例

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

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输**出示例**

```
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
```

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

2.3.3. 使用 Azure Marketplace 产品

您可以创建在 Azure 上运行的机器集，以部署使用 Azure Marketplace 产品的机器。要使用此产品，您必须首先获取 Azure Marketplace 镜像。在获取您的镜像时，请考虑以下事项：

- 虽然镜像相同，但 Azure Marketplace publisher 根据您的区域。如果您位于北美，请将 **redhat** 指定为发布者。如果您位于 EMEA，请将 **redhat-limited** 指定为发布者。
- 此项优惠包括 **rh-ocp-worker** SKU 和 **rh-ocp-worker-gen1** SKU。**rh-ocp-worker** SKU 代表 Hyper-V 生成版本 2 虚拟机镜像。OpenShift Container Platform 中使用的默认实例类型与版本 2 兼容。如果您计划使用与版本 1 兼容的实例类型，请使用与 **rh-ocp-worker-gen1** SKU 关联的镜像。**rh-ocp-worker-gen1** SKU 代表 Hyper-V 版本 1 虚拟机镜像。

重要

在使用 64 位 ARM 实例的集群上不支持使用 Azure marketplace 安装镜像。

先决条件

- 已安装 Azure CLI 客户端 **(az)**。
- 您的 Azure 帐户为产品授权，您使用 Azure CLI 客户端登录到此帐户。

流程

1. 运行以下命令之一，显示所有可用的 OpenShift Container Platform 镜像：
   - 北美：

```
$  az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
```

输**出示例**

```
Offer          Publisher     Sku          Urn                                                  Version
```

```
------------  -------------  -----------------  ------------------------------------------------------------
----  -----------------
rh-ocp-worker  RedHat         rh-ocp-worker       RedHat:rh-ocp-worker:rh-ocp-
worker:413.92.2023101700          413.92.2023101700
rh-ocp-worker  RedHat         rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
gen1:413.92.2023101700       413.92.2023101700
```

- 欧洲、中东和非洲地区：

```
$  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
```

输**出示例**

```
Offer          Publisher       Sku                 Urn
Version
-------------  --------------  ------------------  ------------------------------------------------------------
----          -----------------
rh-ocp-worker  redhat-limited  rh-ocp-worker       redhat-limited:rh-ocp-worker:rh-ocp-
worker:413.92.2023101700          413.92.2023101700
rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
worker-gen1:413.92.2023101700       413.92.2023101700
```

注意

使用可用于 compute 和 control plane 节点的最新镜像。如果需要，您的虚拟机会在安装过程中自动升级。

2. 运行以下命令之一检查您的所提供的镜像：

- 北美：

```
$ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- 欧洲、中东和非洲地区：

```
$ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

3. 运行以下命令之一查看提供的术语：

- 北美：

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- 欧洲、中东和非洲地区：

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 运行以下命令之一接受产品条款：

- 北美：

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- 欧洲、中东和非洲地区：

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. 记录您所提供的镜像详情，特别是 **publisher**, **offer**, **sku**, 和 **version** 的值。
6. 使用您提供的镜像详情，在机器集 YAML 文件的 **providerSpec** 部分添加以下参数：

Azure Marketplace 机器的 providerSpec 镜像值示例

```
providerSpec:
  value:
    image:
      offer: rh-ocp-worker
      publisher: redhat
      resourceID: ""
      sku: rh-ocp-worker
      type: MarketplaceWithPlan
      version: 413.92.2023101700
```

2.3.4. 启用 Azure 引导诊断

您可以在机器集创建的 Azure 机器上启用引导诊断。

先决条件

- 已有 Microsoft Azure 集群。

流程

- 将适用于您的存储类型的 **diagnostics** 配置添加到机器集 YAML 文件中的 **providerSpec** 字段中：
  - 对于 Azure Managed 存储帐户：

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged 1
```

1 指定 Azure Managed 存储帐户。

  - 对于 Azure Unmanaged 存储帐户：

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: CustomerManaged 1
      customerManaged:
        storageAccountURI: https://<storage-account>.blob.core.windows.net 2
```

1 指定 Azure Unmanaged 存储帐户。

2 将 **<storage-account>** 替换为存储帐户的名称。

注意

仅支持 Azure Blob Storage 数据服务。

验证

- 在 Microsoft Azure 门户上，查看机器集部署的机器的 **Boot diagnostics** 页面，并验证您可以看到机器的串行日志。

2.3.5. 将机器部署为 Spot 虚拟机的机器

您可以通过创建一个在 Azure 上运行的计算机器集来节约成本，该机器集将机器部署为非保障的 Spot 虚拟机。Spot VM 使用未使用的 Azure 容量，且比标准虚拟机的成本要低。您可以将 Spot 虚拟机用于可容许中断的工作负载，如批处理或无状态工作负载、横向可扩展工作负载。

Azure 可随时终止 Spot 虚拟机。Azure 在发生中断时向用户发出 30 秒警告。当 Azure 发出终止警告时，OpenShift Container Platform 开始从受影响的实例中删除工作负载。

使用 Spot 虚拟机时可能会因为以下原因造成中断：

- 实例价格超过您的最大价格
- Spot 虚拟机的提供减少
- Azure 需要容量退回

当 Azure 终止实例时，在 Spot VM 节点上运行的终止处理器会删除机器资源。为了满足计算机器设置**副本数量**，计算机器会创建一个请求 Spot 虚拟机的机器。

2.3.5.1. 使用计算机器集创建 Spot 虚拟机

您可以通过在计算机器设置 YAML 文件中添加 **spotVMOptions**，在 Azure 上启动 Spot 虚拟机。

流程

- 在 **providerSpec** 字段中添加以下行：

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

您可以选择设置 **spotVMOptions.maxPrice** 字段来限制 Spot 虚拟机的成本。例如，您可以设置 **maxPrice: '0.98765'**。如果设置了 **maxPrice**，则将此值用作每小时最大即时价格。如果没有设置，则最大价格默认为 **-1** 且不超过标准虚拟机价格。

Azure 封顶 Spot VM 价格以标准价格为基础。如果实例使用默认的 **maxPrice** 设置，Azure 不会因为定价而驱除实例。但是，一个实例仍然可能会因为容量限制而被驱除。

注意

强烈建议您使用默认标准 VM 价格作为 **maxPrice** 值，而不为 Spot 虚拟机设置最大价格。

2.3.6. 在临时操作系统磁盘中部署机器的机器集

您可以创建在 Azure 上运行的计算机器，用于在 Ephemeral OS 磁盘上部署机器。临时 OS 磁盘使用本地虚拟机容量，而不是远程 Azure 存储。因此，此配置不会产生额外费用，并提供了较低的读、写和重新处理延迟。

其他资源

- 如需更多信息，请参阅 Microsoft Azure 文档中有关 Azure 虚拟机的 Ephemeral OS 磁盘 的内容。

2.3.6.1. 使用计算机器在临时磁盘上创建机器

您可以通过编辑计算机器设置 YAML 文件在 Azure 上的 Ephemeral OS 磁盘上启动机器。

先决条件

- 已有 Microsoft Azure 集群。

流程

1. 运行以下命令来编辑自定义资源(CR)：

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

其中 **<machine-set-name>** 是您希望在 Ephemeral OS 磁盘上置备机器的计算机器集。

2. 在 **providerSpec** 字段中添加以下内容：

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

1 2 3 这些行允许使用 Ephemeral OS 磁盘。

4 临时磁盘仅支持虚拟机或扩展使用标准 LRS 存储帐户类型的实例。

重要

OpenShift Container Platform 中支持 Ephemeral OS 磁盘实现只支持 **CacheDisk** 放置类型。不要更改 **placement** 配置设置。

3. 使用更新的配置创建计算机器集：

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

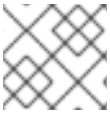
验证

- 在 Microsoft Azure 门户上，查看由计算机器设置部署的机器的 **Overview** 页面，并验证 **Ephemeral OS 磁盘** 字段是否已设置为 **OS 缓存放置**。

2.3.7. 使用计算磁盘部署机器的机器集作为数据磁盘

您可以创建在 Azure 上运行的机器集，该机器集用来部署带有巨型磁盘的机器。ultra 磁盘是高性能存储，用于要求最苛刻的数据工作负载。

您还可以创建一个持久性卷声明(PVC)来动态绑定到 Azureultra 磁盘支持的存储类，并将它们挂载到 pod。



注意

数据磁盘不支持指定磁盘吞吐量或磁盘 IOPS。您可以使用 PVC 配置这些属性。

其他资源

- Microsoft Azure ultra 磁盘文档
- 使用 CSI PVC 在强制磁盘上部署机器的机器集
- 使用树内 PVC 部署机器的机器集

2.3.7.1. 使用机器集创建带有巨型磁盘的机器

您可以通过编辑机器集 YAML 文件在 Azure 上部署带有巨型磁盘的机器。

先决条件

- 已有 Microsoft Azure 集群。

流程

1. 运行以下命令，使用 **worker** 数据 secret 在 **openshift-machine-api** 命名空间中创建自定义 secret：

```
$ oc -n openshift-machine-api \
get secret <role>-user-data \ 1
--template='{{index .data.userData | base64decode}}' | jq > userData.txt 2
```

1. 将 **<role>** 替换为 **worker**。
2. 指定 **userData.txt** 作为新自定义 secret 的名称。

2. 在文本编辑器中，打开 **userData.txt** 文件，并在文件中找到最后的 **}** 字符。
   a. 在紧接下来的行中，添加一个 **,**
   b. 在 **,** 之后创建一个新行并添加以下配置详情：

```
"storage": {
  "disks": [ (1)
    {
      "device": "/dev/disk/azure/scsi1/lun0", (2)
      "partitions": [ (3)
        {
          "label": "lun0p1", (4)
          "sizeMiB": 1024, (5)
          "startMiB": 0
        }
      ]
    }
  ],
  "filesystems": [ (6)
    {
      "device": "/dev/disk/by-partlabel/lun0p1",
      "format": "xfs",
      "path": "/var/lib/lun0p1"
    }
  ]
},
"systemd": {
  "units": [ (7)
    {
      "contents": "[Unit]\nBefore=local-
fs.target\n[Mount]\nWhere=/var/lib/lun0p1\nWhat=/dev/disk/by-
partlabel/lun0p1\nOptions=defaults,pquota\n[Install]\nWantedBy=local-fs.target\n", (8)
      "enabled": true,
      "name": "var-lib-lun0p1.mount"
    }
  ]
}
```

(1) 您要作为 ultra 磁盘附加到节点的磁盘的配置详情。

(2) 指定您使用的机器集的 **dataDisks** 小节中定义的 **lun** 值。例如，如果机器集包含 **lun: 0**，请指定 **lun0**。您可以通过在这个配置文件中指定多个 **"disks"** 条目来初始化多个数据磁盘。如果您指定多个 **"disks"** 条目，请确保每个条目的 **lun** 值与机器集中的值匹配。

(3) 磁盘上新分区的配置详情。

(4) 为分区指定标签。使用分层的名称可能会有帮助，如 **lun0p1** 代表 **lun0** 的第一个分区。

(5) 指定分区的总大小（以 MiB 为单位）。

(6) 指定在格式化分区时要使用的文件系统。使用分区标签来指定分区。

(7) 指定一个 **systemd** 单元来在引导时挂载分区。使用分区标签来指定分区。您可以通过在这个配置文件中指定多个 **"partitions"** 条目来创建多个分区。如果指定多个 **"partitions"** 条目，则必须为每个条目指定一个 **systemd** 单元。

(8) 对于 **where**，指定 **storage.filesystems.path** 的值。对于 **What**，指定 **storage.filesystems.device** 的值。

3. 运行以下命令，将禁用模板值提取到名为 **disableTemplating.txt** 的文件：

```
$ oc -n openshift-machine-api get secret <role>-user-data \ 1
--template='{{index .data.disableTemplating | base64decode}}' | jq > disableTemplating.txt
```

1 将 **<role>** 替换为 **worker**。

4. 运行以下命令组合 **userData.txt** 文件和 **disableTemplating.txt** 文件来创建数据 secret 文件：

```
$ oc -n openshift-machine-api create secret generic <role>-user-data-x5 \ 1
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt
```

1 对于 **<role>-user-data-x5**，请指定 secret 的名称。将 **<role>** 替换为 **worker**。

5. 运行以下命令，复制现有的 Azure **MachineSet** 自定义资源(CR)并编辑它：

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

其中 **<machine-set-name>** 是您要使用巨型磁盘置备机器的机器集。

6. 在指示的位置中添加以下行：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd 1
      providerSpec:
        value:
          ultraSSDCapability: Enabled 2
          dataDisks: 3
          - nameSuffix: ultrassd
            lun: 0
            diskSizeGB: 4
            deletionPolicy: Delete
            cachingType: None
            managedDisk:
              storageAccountType: UltraSSD_LRS
          userDataSecret:
            name: <role>-user-data-x5 4
```

1 指定标签，用于选择此机器集创建的节点。此流程使用 **disk.ulssd** 用于这个值。

2 3 这些行支持使用 ultra 磁盘。对于 **dataDisks**，请包括整个小节。

4 指定之前创建的用户数据 secret。将 **<role>** 替换为 **worker**。

7. 运行以下命令，使用更新的配置创建机器集：

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

验证

1. 运行以下命令验证机器是否已创建：

   ```
   $ oc get machines
   ```

   机器应处于 **Running** 状态。

2. 对于正在运行并附加节点的机器，请运行以下命令验证分区：

   ```
   $ oc debug node/<node-name> -- chroot /host lsblk
   ```

   在这个命令中，**oc debug node/<node-name>** 会在节点 **<node-name>** 上启动一个 debugging shell，并传递一个带有 **--** 的命令。传递的命令 **chroot /host** 提供对底层主机操作系统二进制文件的访问，**lsblk** 显示连接至主机操作系统计算机的块设备。

后续步骤

- 要在 pod 中使用大量磁盘，请创建使用挂载点的工作负载。创建一个类似以下示例的 YAML 文件：

```
apiVersion: v1
kind: Pod
metadata:
  name: ssd-benchmark1
spec:
  containers:
  - name: ssd-benchmark1
    image: nginx
    ports:
      - containerPort: 80
        name: "http-server"
    volumeMounts:
    - name: lun0p1
      mountPath: "/tmp"
  volumes:
    - name: lun0p1
      hostPath:
        path: /var/lib/lun0p1
        type: DirectoryOrCreate
  nodeSelector:
    disktype: ultrassd
```

2.3.7.2. 启用 ultra 磁盘的机器集的故障排除资源

使用本节中的信息从您可能会遇到的问题了解和恢复。

2.3.7.2.1. 不正确的 ultra 磁盘配置

如果在机器集中指定 **ultraSSDCapability** 参数的配置不正确，则机器置备会失败。

例如，如果 **ultraSSDCapability** 参数设置为 **Disabled**，但在 **dataDisks** 参数中指定了 ultra 磁盘，则会出现以下出错信息：

> StorageAccountType UltraSSD_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.

- 要解决这个问题，请验证机器集配置是否正确。

2.3.7.2.2. 不支持的磁盘参数

如果在机器集中指定与 ultra 磁盘不兼容的区域、可用性区域或实例大小，则机器置备会失败。检查日志中的以下出错信息：

> failed to create vm <machine_name>: failure sending request for machine <machine_name>: cannot create vm: compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request: StatusCode=400 -- Original Error: Code="BadRequest" Message="Storage Account type 'UltraSSD_LRS' is not supported <more_information_about_why>."

- 要解决这个问题，请验证您是否在受支持的环境中使用此功能，以及机器设置配置是否正确。

2.3.7.2.3. 无法删除磁盘

如果因为数据磁盘无法按预期工作，则会删除大量磁盘，则机器会被删除，数据磁盘会孤立。如果需要，您必须手动删除孤立的磁盘。

2.3.8. 为机器集启用客户管理的加密密钥

您可以为 Azure 提供加密密钥，以便加密受管磁盘上的数据。您可以使用 Machine API 使用客户管理的密钥启用服务器端加密。

使用客户管理的密钥需要 Azure Key Vault、磁盘加密集和加密密钥。磁盘加密集必须在 Cloud Credential Operator（CCO）授予权限的资源组中。如果没有，则需要在磁盘加密集中授予额外的 reader 角色。

先决条件

- 创建 Azure Key Vault 实例。
- 创建磁盘加密集的实例。
- 授予磁盘加密集对密钥 vault 的访问权限。

流程

- 在机器集 YAML 文件中的 **providerSpec** 字段中配置磁盘加密集。例如：

```
providerSpec:
  value:
    osDisk:
      diskSizeGB: 128
      managedDisk:
        diskEncryptionSet:
          id:
```

```
/subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.Compute/diskEncryptionSets/<disk_encryption_set_name>
        storageAccountType: Premium_LRS
```

其他资源

- Azure 文档中有关客户管理的密钥

2.3.9. 使用机器集为 Azure 虚拟机配置可信启动

重要

为 Azure 虚拟机使用可信启动只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

OpenShift Container Platform 4.16 支持 Azure 虚拟机 (VM) 的可信启动。通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的可信启动选项。例如，您可以将这些机器配置为使用 UEFI 安全功能，如安全引导或专用虚拟信任平台模块 (vTPM) 实例。

注意

有些功能组合会导致配置无效。

表 2.1. UEFI 功能组合兼容性

安全引导 [1]	vTPM[2]	有效配置
Enabled	Enabled	是
Enabled	Disabled	是
Enabled	省略	是
Disabled	Enabled	是
省略	Enabled	是
Disabled	Disabled	否
省略	Disabled	否
省略	省略	否

1. 使用 **secureBoot** 字段。
2. 使用 **virtualizedTrustedPlatformModule** 字段。

有关相关特性和功能的更多信息，请参阅 Microsoft Azure 文档中有关 Azure 虚拟机的受信任的启动 文档。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 编辑 **providerSpec** 字段下的以下部分以提供有效的配置：

   启用 UEFI 安全引导和 vTPM 的有效配置示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          securityProfile:
            settings:
              securityType: TrustedLaunch (1)
              trustedLaunch:
                uefiSettings: (2)
                  secureBoot: Enabled (3)
                  virtualizedTrustedPlatformModule: Enabled (4)
# ...
```

(1) 为 Azure 虚拟机启用可信启动。所有有效配置都需要这个值。

(2) 指定要使用的 UEFI 安全功能。所有有效配置都需要这个部分。

(3) 启用 UEFI 安全引导。

(4) 启用使用 vTPM。

验证

- 在 Azure 门户中，查看机器集部署的机器的详情，并验证可信启动选项是否与您配置的值匹配。

2.3.10. 使用机器集配置 Azure 机密虚拟机

重要

使用 Azure 机密虚拟机只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

OpenShift Container Platform 4.16 支持 Azure 机密虚拟机 (VM)。

注意

64 位 ARM 架构目前不支持机密虚拟机。

通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的机密虚拟机选项。例如，您可以将这些机器配置为使用 UEFI 安全功能，如安全引导或专用虚拟信任平台模块 (vTPM) 实例。

有关相关特性和功能的更多信息，请参阅 Microsoft Azure 文档中有关机密虚拟机的信息。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 在 **providerSpec** 字段中编辑以下部分：

配置示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          osDisk:
            # ...
            managedDisk:
              securityProfile: 1
                securityEncryptionType: VMGuestStateOnly 2
            # ...
          securityProfile: 3
            settings:
              securityType: ConfidentialVM 4
              confidentialVM:
                uefiSettings: 5
                  secureBoot: Disabled 6
                  virtualizedTrustedPlatformModule: Enabled 7
          vmSize: Standard_DC16ads_v5 8
# ...
```

1. 在使用机密虚拟机时，指定受管磁盘的安全配置集设置。
2. 启用 Azure VM Guest State (VMGS) blob 加密。此设置需要使用 vTPM。
3. 指定机密虚拟机的安全配置集设置。
4. 启用使用机密虚拟机。所有有效配置都需要这个值。
5. 指定要使用的 UEFI 安全功能。所有有效配置都需要这个部分。
6. 禁用 UEFI 安全引导。
7. 启用使用 vTPM。

8 指定支持机密虚拟机的实例类型。

验证

- 在 Azure 门户中，查看机器集部署的机器的详情，并验证机密虚拟机选项是否与您配置的值匹配。

2.3.11. Microsoft Azure 虚拟机的加速网络

加速网络使用单一根 I/O 虚拟化(SR-IOV)为 Microsoft Azure 虚拟机提供更直接的路径到交换机。这提高了网络性能。此功能可在安装过程中或安装后启用。

2.3.11.1. 限制

在决定是否使用加速网络时，请考虑以下限制：

- 只有在 Machine API 操作的集群中支持加速网络。
- 虽然 Azure worker 节点的最低要求是两个 vCPU，但 Accelerated Networking 需要包含至少四个 vCPU 的 Azure 虚拟机大小。为了满足此要求，您可以在机器集中更改 **vmSize** 的值。有关 Azure VM 大小的信息，请参阅 Microsoft Azure 文档。
- 当在现有 Azure 集群上启用这个功能时，只有新置备的节点会受到影响。当前运行的节点不会被协调。要在所有节点上启用这个功能，必须替换每个现有机器。这可以为每个机器单独完成，或者将副本缩减为零，然后备份到所需的副本数。

2.3.12. 将 GPU 节点添加到现有 OpenShift Container Platform 集群中

您可以复制并修改默认计算机器集配置，以便为 Azure 云供应商创建启用了 GPU 的机器集和机器。

下表列出了经过验证的实例类型：

vmSize	NVIDIA GPU 加速器	最大 GPU 数	架构
Standard_NC24s_v3	V100	4	x86
Standard_NC4as_T4_v3	T4	1	x86
ND A100 v4	A100	8	x86

注意

默认情况下，Azure 订阅没有 GPU 的 Azure 实例类型的配额。客户必须为上面列出的 Azure 实例系列请求配额增加。

流程

1. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器和机器集。每个计算机器集都与 Azure 区域的不同可用区关联。安装程序会在可用区之间自动负载平衡计算机器。

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

输**出示例**

```
NAME                          DESIRED   CURRENT   READY   AVAILABLE   AGE
myclustername-worker-centralus1   1         1         1       1           6h9m
myclustername-worker-centralus2   1         1         1       1           6h9m
myclustername-worker-centralus3   1         1         1       1           6h9m
```

2. 运行以下命令，复制现有计算 **MachineSet** 定义并将结果输出到 YAML 文件。这将是启用了 GPU 的计算机器集定义的基础。

```
$ oc get machineset -n openshift-machine-api myclustername-worker-centralus1 -o yaml > machineset-azure.yaml
```

3. 查看 machineset 的内容：

```
$ cat machineset-azure.yaml
```

machineset-azure.yaml 文件示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  annotations:
    machine.openshift.io/GPU: "0"
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
  creationTimestamp: "2023-02-06T14:08:19Z"
  generation: 1
  labels:
    machine.openshift.io/cluster-api-cluster: myclustername
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: myclustername-worker-centralus1
  namespace: openshift-machine-api
  resourceVersion: "23601"
  uid: acd56e0c-7612-473a-ae37-8704f34b80de
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: myclustername
      machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: myclustername
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
    spec:
      lifecycleHooks: {}
      metadata: {}
```

```
      providerSpec:
        value:
          acceleratedNetworking: true
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          diagnostics: {}
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/myclustername-
rg/providers/Microsoft.Compute/galleries/gallery_myclustername_n6n4r/images/myclustername
-gen2/versions/latest
            sku: ""
            version: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: myclustername-identity
          metadata:
            creationTimestamp: null
          networkResourceGroup: myclustername-rg
          osDisk:
            diskSettings: {}
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: myclustername
          resourceGroup: myclustername-rg
          spotVMOptions: {}
          subnet: myclustername-worker-subnet
          userDataSecret:
            name: worker-user-data
          vmSize: Standard_D4s_v3
          vnet: myclustername-vnet
          zone: "1"
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
```

4. 运行以下命令，生成 **machineset-azure.yaml** 文件的副本：

```
$ cp machineset-azure.yaml machineset-azure-gpu.yaml
```

5. 更新 **machineset-azure-gpu.yaml** 中的以下字段：

   - 将 **.metadata.name** 更改为包含 **gpu** 的名称。
   - 更改 **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]**，以匹配新的 .metadata.name。

- 更改 **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]**，以匹配新的 **.metadata.name**。
- 将 **.spec.template.spec.providerSpec.value.vmSize** 更改为 **Standard_NC4as_T4_v3**。

machineset-azure-gpu.yaml 文件示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  annotations:
    machine.openshift.io/GPU: "1"
    machine.openshift.io/memoryMb: "28672"
    machine.openshift.io/vCPU: "4"
  creationTimestamp: "2023-02-06T20:27:12Z"
  generation: 1
  labels:
    machine.openshift.io/cluster-api-cluster: myclustername
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: myclustername-nc4ast4-gpu-worker-centralus1
  namespace: openshift-machine-api
  resourceVersion: "166285"
  uid: 4eedce7f-6a57-4abe-b529-031140f02ffa
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: myclustername
      machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: myclustername
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
    spec:
      lifecycleHooks: {}
      metadata: {}
      providerSpec:
        value:
          acceleratedNetworking: true
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          diagnostics: {}
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/myclustername-
rg/providers/Microsoft.Compute/galleries/gallery_myclustername_n6n4r/images/myclustern
ame-gen2/versions/latest
```

```
            sku: ""
            version: ""
          kind: AzureMachineProviderSpec
          location: centralus
          managedIdentity: myclustername-identity
          metadata:
            creationTimestamp: null
          networkResourceGroup: myclustername-rg
          osDisk:
            diskSettings: {}
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: myclustername
          resourceGroup: myclustername-rg
          spotVMOptions: {}
          subnet: myclustername-worker-subnet
          userDataSecret:
            name: worker-user-data
          vmSize: Standard_NC4as_T4_v3
          vnet: myclustername-vnet
          zone: "1"
      status:
        availableReplicas: 1
        fullyLabeledReplicas: 1
        observedGeneration: 1
        readyReplicas: 1
        replicas: 1
```

6. 要验证您的更改，请运行以下命令对原始计算定义和新的 GPU 节点定义执行 **diff** ：

```
$ diff machineset-azure.yaml machineset-azure-gpu.yaml
```

输**出示例**

```
14c14
<   name: myclustername-worker-centralus1
---
>   name: myclustername-nc4ast4-gpu-worker-centralus1
23c23
<       machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
---
>       machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
30c30
<         machine.openshift.io/cluster-api-machineset: myclustername-worker-centralus1
---
>         machine.openshift.io/cluster-api-machineset: myclustername-nc4ast4-gpu-worker-
centralus1
67c67
<           vmSize: Standard_D4s_v3
---
>           vmSize: Standard_NC4as_T4_v3
```

7. 运行以下命令，从定义文件创建启用了 GPU 的计算机器集：

```
$ oc create -f machineset-azure-gpu.yaml
```

输**出示例**

```
machineset.machine.openshift.io/myclustername-nc4ast4-gpu-worker-centralus1 created
```

8. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器和机器集。每个计算机器集都与 Azure 区域的不同可用区关联。安装程序会在可用区之间自动负载平衡计算机器。

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

输**出示例**

```
NAME                                  DESIRED   CURRENT   READY   AVAILABLE   AGE
clustername-n6n4r-nc4ast4-gpu-worker-centralus1   1         1         1       1           122m
clustername-n6n4r-worker-centralus1               1         1         1       1           8h
clustername-n6n4r-worker-centralus2               1         1         1       1           8h
clustername-n6n4r-worker-centralus3               1         1         1       1           8h
```

9. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器。您只能为每个集合配置一个计算机器，但您可以扩展计算机器集，以便在特定地区和区中添加节点。

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

输**出示例**

```
NAME                                        PHASE     TYPE              REGION      ZONE   AGE
myclustername-master-0                      Running   Standard_D8s_v3   centralus   2      6h40m
myclustername-master-1                      Running   Standard_D8s_v3   centralus   1      6h40m
myclustername-master-2                      Running   Standard_D8s_v3   centralus   3      6h40m
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn   Running   centralus   1   21m
myclustername-worker-centralus1-rbh6b       Running   Standard_D4s_v3   centralus   1      6h38m
myclustername-worker-centralus2-dbz7w       Running   Standard_D4s_v3   centralus   2      6h38m
myclustername-worker-centralus3-p9b8c       Running   Standard_D4s_v3   centralus   3      6h38m
```

10. 运行以下命令，查看现有节点、机器和机器集。请注意，每个节点都是带有特定 Azure 区域和 OpenShift Container Platform 角色的机器定义实例。

```
$ oc get nodes
```

输**出示例**

```
NAME                                STATUS   ROLES                  AGE     VERSION
myclustername-master-0              Ready    control-plane,master   6h39m   v1.29.4
```

```
myclustername-master-1                          Ready    control-plane,master   6h41m   v1.29.4
myclustername-master-2                          Ready    control-plane,master   6h39m   v1.29.4
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn   Ready    worker                 14m
v1.29.4
myclustername-worker-centralus1-rbh6b           Ready    worker                 6h29m   v1.29.4
myclustername-worker-centralus2-dbz7w           Ready    worker                 6h29m   v1.29.4
myclustername-worker-centralus3-p9b8c           Ready    worker                 6h31m   v1.29.4
```

11. 查看计算机器集的列表：

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

输**出示例**

```
NAME                                 DESIRED   CURRENT   READY   AVAILABLE   AGE
myclustername-worker-centralus1      1         1         1       1           8h
myclustername-worker-centralus2      1         1         1       1           8h
myclustername-worker-centralus3      1         1         1       1           8h
```

12. 运行以下命令，从定义文件创建启用了 GPU 的计算机器集：

```
$ oc create -f machineset-azure-gpu.yaml
```

13. 查看计算机器集的列表：

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

输**出示例**

```
NAME                                 DESIRED   CURRENT   READY   AVAILABLE   AGE
myclustername-nc4ast4-gpu-worker-centralus1   1         1         1       1           121m
myclustername-worker-centralus1      1         1         1       1           8h
myclustername-worker-centralus2      1         1         1       1           8h
myclustername-worker-centralus3      1         1         1       1           8h
```

验证

1. 运行以下命令，查看您创建的机器集：

```
$ oc get machineset -n openshift-machine-api | grep gpu
```

MachineSet 副本数被设置为 **1**，以便自动创建新的 **Machine** 对象。

输**出示例**

```
myclustername-nc4ast4-gpu-worker-centralus1   1         1         1       1           121m
```

2. 运行以下命令，查看创建机器集的 **Machine** 对象：

```
$ oc -n openshift-machine-api get machines | grep gpu
```

输**出示例**

```
myclustername-nc4ast4-gpu-worker-centralus1-w9bqn   Running   Standard_NC4as_T4_v3
centralus   1      21m
```

注意

不需要为节点指定命名空间。节点定义是在集群范围之内。

2.3.13. 部署 Node Feature Discovery Operator

创建启用了 GPU 的节点后，您需要发现启用了 GPU 的节点，以便调度它。为此，请安装 Node Feature Discovery (NFD) Operator。NFD Operator 识别节点中的硬件设备功能。它解决了在基础架构节点中识别和目录硬件资源的一般问题，以便 OpenShift Container Platform 可以使用它们。

流程

1. 在 OpenShift Container Platform 控制台中，从 **OperatorHub** 安装 Node Feature Discovery Operator。
2. 将 NFD Operator 安装到 **OperatorHub** 后，从已安装的 Operator 列表中选择 **Node Feature Discovery**，然后选择 **Create instance**。这会在 **openshift-nfd** 命名空间中安装 **nfd-master** 和 **nfd-worker** pod，每个计算节点一个 **nfd-worker** pod。
3. 运行以下命令验证 Operator 是否已安装并正在运行：

```
$ oc get pods -n openshift-nfd
```

输**出示例**

```
NAME                                  READY   STATUS    RESTARTS   AGE

nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running 7  (8h ago)   1d
```

4. 浏览到控制台中的已安装的 Oerator，再选择 **Create Node Feature Discovery**。
5. 选择 **Create** 以构建 NFD 自定义资源。这会在 **openshift-nfd** 命名空间中创建 NFD pod，为硬件资源和目录轮询 OpenShift Container Platform 节点。

验证

1. 构建成功后，运行以下命令来验证 NFD pod 是否在每个节点上运行：

```
$ oc get pods -n openshift-nfd
```

输**出示例**

```
NAME                                  READY   STATUS      RESTARTS       AGE
nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running     7 (8h ago)     12d
nfd-master-769656c4cb-w9vrv               1/1     Running     0              12d
nfd-worker-qjxb2                      1/1     Running     3 (3d14h ago)   12d
nfd-worker-xtz9b                      1/1     Running     5 (3d14h ago)   12d
```

NFD Operator 使用厂商 PCI ID 来识别节点的硬件。NVIDIA 使用 PCI ID **10de**。

2. 运行以下命令，查看 NFD Operator 发现的 NVIDIA GPU：

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

输出示例

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

10de 会出现在启用了 GPU 的节点的节点功能列表中。这意味着 NFD Operator 可以正确地识别启用了 GPU 的 MachineSet 的节点。

其他资源

- 在安装过程中启用加速网络

2.3.13.1. 在现有 Microsoft Azure 集群上启用加速网络

您可以通过在机器集 YAML 文件中添加 **acceleratedNetworking**，在 Azure 上启用加速网络。

先决条件

- 有一个现有的 Microsoft Azure 集群，其中的 Machine API 正常运行。

流程

- 在 **providerSpec** 字段中添加以下内容：

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

(1) 此行启用加速网络。

(2) 指定包含至少四个 vCPU 的 Azure VM 大小。有关 VM 大小的信息，请参阅 Microsoft Azure 文档。

后续步骤

- 要在当前运行的节点上启用这个功能，必须替换每个现有机器。这可以为每个机器单独完成，或者将副本缩减为零，然后备份到所需的副本数。

验证

- 在 Microsoft Azure 门户上，查看机器集调配的机器的 **Networking** 设置页面，并验证 **Accelerated networking** 字段设置为 **Enabled**。

其他资源

- 手动扩展计算机器集

2.4. 在 AZURE STACK HUB 上创建计算机器集

您可以在 Microsoft Azure Stack Hub 上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.4.1. Azure Stack Hub 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在区域(region)的 **1** Microsoft Azure 区域(zone)中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
```

```
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: "" 11
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          availabilitySet: <availability_set> 12
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/images/<infrastructure_id> 13
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 14
          managedIdentity: <infrastructure_id>-identity 15
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: ""
          resourceGroup: <infrastructure_id>-rg 16
          sshPrivateKey: ""
          sshPublicKey: ""
          subnet: <infrastructure_id>-<role>-subnet 17 18
          userDataSecret:
            name: worker-user-data 19
          vmSize: Standard_DS4_v2
          vnet: <infrastructure_id>-vnet 20
          zone: "1" 21
```

1 5 7 13 15 16 17 20 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
```

您可以运行以下命令来获取子网：

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

您可以运行以下命令来获取 vnet：

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

2 3 8 9 11 18 19 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。

14 指定要放置机器的区域。

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

12 指定集群的可用性集。

2.4.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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**。
- 创建一个可用性集，在其中部署 Azure Stack Hub 计算机器。

流程

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

2. 可选：如果您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计算机器集：

   a. 要列出集群中的计算机器集，请运行以下命令：

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

   输出示例

   ```
   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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输出示例

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

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.4.3. 启用 Azure 引导诊断

您可以在机器集创建的 Azure 机器上启用引导诊断。

先决条件

- 有一个现有的 Microsoft Azure Stack Hub 集群。

流程

- 将适用于您的存储类型的 **diagnostics** 配置添加到机器集 YAML 文件中的 **providerSpec** 字段中：
  - 对于 Azure Managed 存储帐户：

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged 1
```

1 指定 Azure Managed 存储帐户。

  - 对于 Azure Unmanaged 存储帐户：

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: CustomerManaged 1
      customerManaged:
        storageAccountURI: https://<storage-account>.blob.core.windows.net 2
```

1 指定 Azure Unmanaged 存储帐户。

2 将 **<storage-account>** 替换为存储帐户的名称。

注意

仅支持 Azure Blob Storage 数据服务。

验证

- 在 Microsoft Azure 门户上，查看机器集部署的机器的 **Boot diagnostics** 页面，并验证您可以看到机器的串行日志。

2.4.4. 为机器集启用客户管理的加密密钥

您可以为 Azure 提供加密密钥，以便加密受管磁盘上的数据。您可以使用 Machine API 使用客户管理的密钥启用服务器端加密。

使用客户管理的密钥需要 Azure Key Vault、磁盘加密集和加密密钥。磁盘加密集必须在 Cloud Credential Operator（CCO）授予权限的资源组中。如果没有，则需要在磁盘加密集中授予额外的 reader 角色。

先决条件

- 创建 Azure Key Vault 实例。
- 创建磁盘加密集的实例。
- 授予磁盘加密集对密钥 vault 的访问权限。

流程

- 在机器集 YAML 文件中的 **providerSpec** 字段中配置磁盘加密集。例如：

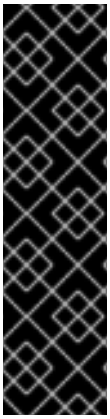
```
providerSpec:
  value:
    osDisk:
      diskSizeGB: 128
      managedDisk:
        diskEncryptionSet:
          id:
/subscriptions/<subscription_id>/resourceGroups/<resource_group_name>/providers/Microsoft.
Compute/diskEncryptionSets/<disk_encryption_set_name>
        storageAccountType: Premium_LRS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

其他资源

- Azure 文档中有关客户管理的密钥

2.5. 在 GCP 上创建计算机器设置

您可以在 Google Cloud Platform (GCP)上的 OpenShift Container Platform 集群中创建不同的计算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。



重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.5.1. GCP 上计算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)中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<role>: ""** 标记的节点，其中 **<role>** 是要添加的节点标签。

使用 OpenShift CLI 获取的值
在以下示例中，您可以使用 OpenShift CLI 获取集群的一些值。

基础架构 ID

<infrastructure_id> 字符串是基础架构 ID，它基于您在置备集群时设定的集群 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
```

镜像路径

<path_to_image> 字符串是用于创建磁盘的镜像的路径。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取镜像的路径：

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}{"\n"}' \
  get machineset/<infrastructure_id>-worker-a
```

GCP MachineSet 值示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
  template:
    metadata:
      creationTimestamp: null
```

```
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <role> 2
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: ""
    providerSpec:
      value:
        apiVersion: gcpprovider.openshift.io/v1beta1
        canIPForward: false
        credentialsSecret:
          name: gcp-cloud-credentials
        deletionProtection: false
        disks:
        - autoDelete: true
          boot: true
          image: <path_to_image> 3
          labels: null
          sizeGb: 128
          type: pd-ssd
        gcpMetadata: 4
        - key: <custom_metadata_key>
          value: <custom_metadata_value>
        kind: GCPMachineProviderSpec
        machineType: n1-standard-4
        metadata:
          creationTimestamp: null
        networkInterfaces:
        - network: <infrastructure_id>-network
          subnetwork: <infrastructure_id>-worker-subnet
        projectID: <project_name> 5
        region: us-central1
        serviceAccounts:
        - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com
          scopes:
          - https://www.googleapis.com/auth/cloud-platform
        tags:
          - <infrastructure_id>-worker
        userDataSecret:
          name: worker-user-data
        zone: us-central1-a
```

1 其中 **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。

2 对于 **<node>**，指定要添加的节点标签。

3 指定当前计算机器集中使用的镜像的路径。

要使用 GCP Marketplace 镜像，请指定要使用的功能：

- OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**

- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

4 可选：以 **key:value** 对的形式指定自定义元数据。有关用例，请参阅 GCP 文档，以查看设置自定义元数据。

5 对于 **<project_name>**，请指定用于集群的 GCP 项目的名称。

2.5.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

   输出示例

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

   输出示例

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

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.5.3. 使用机器集配置持久性磁盘类型

您可以通过编辑机器集 YAML 文件，配置机器集在其中部署机器的持久性磁盘类型。

有关持久性磁盘类型、兼容性、区域可用性和限制的更多信息，请参阅 GCP Compute Engine 文档有关持久性磁盘的信息。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 编辑 **providerSpec** 字段中的以下行：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            type: <pd-disk-type> 1
```

1 指定持久性磁盘类型。有效值为 **pd-ssd**、**pd-standard** 和 **pd-balanced**。默认值为 **pd-standard**。

验证

- 在 Google Cloud 控制台中，查看机器集部署的机器的详情，并验证 **Type** 字段是否与配置的磁盘类型匹配。

2.5.4. 使用机器集配置机密虚拟机

通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的机密虚拟机选项。

有关机密虚拟机功能、功能和兼容性的更多信息，请参阅 GCP Compute Engine 文档有关 机密虚拟机 的信息。

注意

64 位 ARM 架构目前不支持机密虚拟机。

重要

OpenShift Container Platform 4.16 不支持一些机密计算功能，如使用 AMD Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP) 的机密虚拟机。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 在 **providerSpec** 字段中编辑以下部分：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          confidentialCompute: Enabled (1)
          onHostMaintenance: Terminate (2)
          machineType: n2d-standard-8 (3)
...
```

(1) 指定是否启用机密虚拟机。有效值为 **Disabled** 或 **Enabled**。

(2) 指定主机维护事件期间虚拟机的行为，如硬件或软件更新。对于使用机密虚拟机的机器，此值必须设置为 **Terminate**，这会停止虚拟机。机密虚拟机不支持实时迁移。

(3) 指定支持机密虚拟机的机器类型。机密虚拟机支持 N2D 和 C2D 系列机器类型。

验证

- 在 Google Cloud 控制台中，查看机器集部署的机器的详情，并验证 Confidential VM 选项是否与您配置的值匹配。

2.5.5. 将机器部署为可抢占虚拟机实例的机器集

您可以通过创建一个在 GCP 上运行的计算机器集来节约成本，该 MachineSet 将机器部署为非保障的虚拟机实例。抢占虚拟机实例使用了超额的 Compute Engine 容量，且比一般实例的成本要低。您可以将抢占虚拟机实例用于可容许中断的工作负载，如批处理或无状态工作负载、横向可扩展工作负载。

GCP Compute Engine 可随时终止可抢占的虚拟机实例。Compute Engine 向用户发送抢占通知，表示会在 30 秒内发生中断。当 Compute Engine 发出抢占通知时，OpenShift Container Platform 开始从受影响的实例中删除工作负载。如果实例没有停止，则 ACPI G3 Mechanical Off 信号会在 30 秒后发送到操作系统。然后，抢占虚拟机实例由 Compute Engine 转换为 **TERMINATED** 状态。

使用抢占虚拟机实例时可能会出现中断，理由如下：

- 有系统或维护事件
- 提供的抢占虚拟机实例减少
- 该实例为抢占虚拟机实例到达分配的 24 小时期限的结束

当 GCP 终止一个实例时，在可抢占虚拟机实例节点上运行的终止处理器会删除机器资源。为了满足计算机器集**副本数**量，计算机器会创建一个请求抢占虚拟机实例的机器。

2.5.5.1. 使用计算机器集创建抢占虚拟机实例

您可以通过在计算机器设置 YAML 文件中添加 **preemptible**，在 GCP 上启动抢占虚拟机实例。

流程

- 在 **providerSpec** 字段中添加以下行：

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

如果 **preemptible** 被设置为 **true**，则在实例启动后，机器将被标记为 **interruptable-instance**。

2.5.6. 使用机器集配置 Shielded VM 选项

通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的 Shielded VM 选项。

有关 Shielded VM 特性和功能的更多信息，请参阅有关 Shielded VM 的 GCP Compute Engine 文档。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 在 **providerSpec** 字段中编辑以下部分：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          shieldedInstanceConfig: 1
            integrityMonitoring: Enabled 2
            secureBoot: Disabled 3
            virtualizedTrustedPlatformModule: Enabled 4
# ...
```

1. 在本节中，指定您想要的 Shielded VM 选项。
2. 指定是否启用了完整性监控。有效值为 **Disabled** 或 **Enabled**。

注意

启用完整性监控后，不得禁用虚拟可信平台模块 (vTPM)。

3. 指定是否启用了 UEFI 安全引导。有效值为 **Disabled** 或 **Enabled**。

4 指定是否启用了 vTPM。有效值为 **Disabled** 或 **Enabled**。

验证

- 使用 Google Cloud 控制台，查看机器集部署的机器的详情，并验证 Shielded VM 选项是否与您配置的值匹配。

其他资源

- 什么是 Shielded VM?
  - 安全引导
  - 虚拟受信任的平台模块 (vTPM)
  - 完整性监控

2.5.7. 为机器集启用客户管理的加密密钥

Google Cloud Platform（GCP）Compute Engine 允许用户提供加密密钥来加密磁盘上的数据。密钥用于对数据加密密钥进行加密，而不是加密客户的数据。默认情况下，Compute Engine 使用 Compute Engine 密钥加密这些数据。

您可以使用 Machine API 的集群中使用客户管理的密钥启用加密。您必须首先创建 KMS 密钥并为服务帐户分配正确的权限。需要 KMS 密钥名称、密钥环名称和位置来允许服务帐户使用您的密钥。

注意

如果您不想将专用服务帐户用于 KMS 加密，则使用 Compute Engine 默认服务帐户。如果没有使用专用服务帐户，则必须授予默认服务帐户权限来访问密钥。Compute Engine 默认服务帐户名称遵循 **service-<project_number>@compute-system.iam.gserviceaccount.com** 模式。

流程

1. 要允许特定服务帐户使用 KMS 密钥，并为服务帐户授予正确的 IAM 角色，请使用您的 KMS 密钥名称、密钥环名称和位置运行以下命令：

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

2. 在机器集 YAML 文件中的 **providerSpec** 字段中配置加密密钥。例如：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
spec:
  template:
    spec:
      providerSpec:
```

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

1. 用于磁盘加密的客户管理的加密密钥名称。
2. KMS 密钥所属的 KMS 密钥环的名称。
3. KMS 密钥环存在的 GCP 位置。
4. 可选：存在 KMS 密钥环的项目 ID。如果没有设置项目 ID，则会使用创建机器设置的机器设置 **projectID**。
5. 可选：用于给定 KMS 密钥加密请求的服务帐户。如果没有设置服务帐户，则使用 Compute Engine 默认服务帐户。

当使用更新的 **providerSpec** 对象配置创建新机器后，磁盘加密密钥就会使用 KMS 密钥加密。

2.5.8. 为计算机器设置启用 GPU 支持

Google Cloud Platform(GCP)Compute Engine 允许用户在虚拟机实例中添加 GPU。得益于访问 GPU 资源的工作负载可在启用了此功能的计算机器上更好地执行。GCP 上的 OpenShift Container Platform 支持 A2 和 N1 机器集中的 NVIDIA GPU 模型。

表 2.2. 支持的 GPU 配置

型号名称	GPU 类型	机器类型 [1]
NVIDIA A100	**nvidia-tesla-a100**	• **a2-highgpu-1g** • **a2-highgpu-2g** • **a2-highgpu-4g** • **a2-highgpu-8g** • **a2-megagpu-16g**

型号名称	GPU 类型	机器类型 [1]
NVIDIA K80	**nvidia-tesla-k80**	• **n1-standard-1** • **n1-standard-2** • **n1-standard-4** • **n1-standard-8** • **n1-standard-16** • **n1-standard-32** • **n1-standard-64** • **n1-standard-96** • **n1-highmem-2** • **n1-highmem-4** • **n1-highmem-8** • **n1-highmem-16** • **n1-highmem-32** • **n1-highmem-64** • **n1-highmem-96** • **n1-highcpu-2** • **n1-highcpu-4** • **n1-highcpu-8** • **n1-highcpu-16** • **n1-highcpu-32** • **n1-highcpu-64** • **n1-highcpu-96**
NVIDIA P100	**nvidia-tesla-p100**	
NVIDIA P4	**nvidia-tesla-p4**	
NVIDIA T4	**nvidia-tesla-t4**	
NVIDIA V100	**nvidia-tesla-v100**	

1. 有关机器类型的更多信息，包括规格、兼容性、区域可用性和限制，请参阅 GCP Compute Engine 文档，有关 N1 机器集、A2 计算机系列和 GPU 区域以及区域可用性。

您可以使用 Machine API 定义要用于实例的 GPU。

您可以在 N1 计算机序列中配置机器，以使用其中一个支持的 GPU 类型进行部署。A2 计算机系列中的机器附带关联的 GPU，无法使用客户机加速器。

注意

不支持用于图形工作负载的 GPU。

流程

1. 在文本编辑器中，打开现有计算机器集的 YAML 文件，或创建新的计算机器。
2. 在计算机器设置 YAML 文件中的 **providerSpec** 字段中指定 GPU 配置。请参见以下有效配置示例：

A2 机器集配置示例：

```
providerSpec:
  value:
    machineType: a2-highgpu-1g (1)
    onHostMaintenance: Terminate (2)
    restartPolicy: Always (3)
```

(1) 指定机器类型。确保在 A2 机器集中包含机器类型。

(2) 在使用 GPU 支持时，您必须将 **HostMaintenance** 设置为 **终止**。

(3) 为计算机器设置部署的机器指定重启策略。允许的值是 **Always** 或 **Never**。

N1 机器集配置示例：

```
providerSpec:
  value:
    gpus:
    - count: 1 (1)
      type: nvidia-tesla-p100 (2)
    machineType: n1-standard-1 (3)
    onHostMaintenance: Terminate (4)
    restartPolicy: Always (5)
```

(1) 指定要附加到机器的 GPU 数。

(2) 指定要附加到机器的 GPU 类型。确保机器类型和 GPU 类型兼容。

(3) 指定机器类型。确保机器类型和 GPU 类型兼容。

(4) 在使用 GPU 支持时，您必须将 **HostMaintenance** 设置为 **终止**。

(5) 为计算机器设置部署的机器指定重启策略。允许的值是 **Always** 或 **Never**。

2.5.9. 将 GPU 节点添加到现有 OpenShift Container Platform 集群中

您可以复制并修改默认计算机器集配置，以便为 GPU 云供应商创建启用了 GPU 的机器集和机器。

下表列出了经过验证的实例类型：

实例类型	NVIDIA GPU 加速器	最大 GPU 数	架构
a2-highgpu-1g	A100	1	x86

实例类型	NVIDIA GPU 加速器	最大 GPU 数	架构
n1-standard-4	T4	1	x86

流程

1. 制作现有 **MachineSet** 的副本。
2. 在新副本中，在 **metadata.name** 中更改机器集**名称**，并在 **machine.openshift.io/cluster-api-machineset** 的两个实例中更改机器集名称。
3. 更改实例类型，将以下两行添加到新复制的 **MachineSet** ：

```
machineType: a2-highgpu-1g
onHostMaintenance: Terminate
```

a2-highgpu-1g.json 文件示例

```
{
    "apiVersion": "machine.openshift.io/v1beta1",
    "kind": "MachineSet",
    "metadata": {
        "annotations": {
            "machine.openshift.io/GPU": "0",
            "machine.openshift.io/memoryMb": "16384",
            "machine.openshift.io/vCPU": "4"
        },
        "creationTimestamp": "2023-01-13T17:11:02Z",
        "generation": 1,
        "labels": {
            "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p"
        },
        "name": "myclustername-2pt9p-worker-gpu-a",
        "namespace": "openshift-machine-api",
        "resourceVersion": "20185",
        "uid": "2daf4712-733e-4399-b4b4-d43cb1ed32bd"
    },
    "spec": {
        "replicas": 1,
        "selector": {
            "matchLabels": {
                "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p",
                "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-gpu-
a"
            }
        },
        "template": {
            "metadata": {
                "labels": {
                    "machine.openshift.io/cluster-api-cluster": "myclustername-2pt9p",
                    "machine.openshift.io/cluster-api-machine-role": "worker",
                    "machine.openshift.io/cluster-api-machine-type": "worker",
                    "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
```

```
gpu-a"
            }
          },
          "spec": {
            "lifecycleHooks": {},
            "metadata": {},
            "providerSpec": {
              "value": {
                "apiVersion": "machine.openshift.io/v1beta1",
                "canIPForward": false,
                "credentialsSecret": {
                  "name": "gcp-cloud-credentials"
                },
                "deletionProtection": false,
                "disks": [
                  {
                    "autoDelete": true,
                    "boot": true,
                    "image": "projects/rhcos-cloud/global/images/rhcos-412-86-
202212081411-0-gcp-x86-64",
                    "labels": null,
                    "sizeGb": 128,
                    "type": "pd-ssd"
                  }
                ],
                "kind": "GCPMachineProviderSpec",
                "machineType": "a2-highgpu-1g",
                "onHostMaintenance": "Terminate",
                "metadata": {
                  "creationTimestamp": null
                },
                "networkInterfaces": [
                  {
                    "network": "myclustername-2pt9p-network",
                    "subnetwork": "myclustername-2pt9p-worker-subnet"
                  }
                ],
                "preemptible": true,
                "projectID": "myteam",
                "region": "us-central1",
                "serviceAccounts": [
                  {
                    "email": "myclustername-2pt9p-w@myteam.iam.gserviceaccount.com",
                    "scopes": [
                      "https://www.googleapis.com/auth/cloud-platform"
                    ]
                  }
                ],
                "tags": [
                  "myclustername-2pt9p-worker"
                ],
                "userDataSecret": {
                  "name": "worker-user-data"
                },
                "zone": "us-central1-a"
              }
```

```
            }
          }
        }
    },
    "status": {
        "availableReplicas": 1,
        "fullyLabeledReplicas": 1,
        "observedGeneration": 1,
        "readyReplicas": 1,
        "replicas": 1
    }
}
```

4. 运行以下命令，查看现有节点、机器和机器集。请注意，每个节点都是带有特定 GCP 区域和 OpenShift Container Platform 角色的机器定义实例。

```
$ oc get nodes
```

输**出示例**

```
NAME                                                STATUS   ROLES                  AGE   VERSION
myclustername-2pt9p-master-0.c.openshift-qe.internal        Ready    control-plane,master   8h    v1.29.4
myclustername-2pt9p-master-1.c.openshift-qe.internal        Ready    control-plane,master   8h    v1.29.4
myclustername-2pt9p-master-2.c.openshift-qe.internal        Ready    control-plane,master   8h    v1.29.4
myclustername-2pt9p-worker-a-mxtnz.c.openshift-qe.internal   Ready    worker                 8h    v1.29.4
myclustername-2pt9p-worker-b-9pzzn.c.openshift-qe.internal   Ready    worker                 8h    v1.29.4
myclustername-2pt9p-worker-c-6pbg6.c.openshift-qe.internal   Ready    worker                 8h    v1.29.4
myclustername-2pt9p-worker-gpu-a-wxcr6.c.openshift-qe.internal   Ready    worker         4h35m   v1.29.4
```

5. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器和机器集。每个计算机器集都与 GCP 区域的不同可用区关联。安装程序会在可用区之间自动负载平衡计算机器。

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

输**出示例**

```
NAME                           DESIRED   CURRENT   READY   AVAILABLE   AGE
myclustername-2pt9p-worker-a   1         1         1       1           8h
myclustername-2pt9p-worker-b   1         1         1       1           8h
myclustername-2pt9p-worker-c   1         1                 8h
myclustername-2pt9p-worker-f   0         0                 8h
```

6. 运行以下命令，查看 **openshift-machine-api** 命名空间中存在的机器。您只能为每个集合配置一个计算机器，但您可以扩展计算机器集，以便在特定地区和区中添加节点。

```
$ oc get machines -n openshift-machine-api | grep worker
```

输**出示例**

```
myclustername-2pt9p-worker-a-mxtnz      Running   n2-standard-4   us-central1   us-
central1-a   8h
myclustername-2pt9p-worker-b-9pzzn      Running   n2-standard-4   us-central1   us-
central1-b   8h
myclustername-2pt9p-worker-c-6pbg6      Running   n2-standard-4   us-central1   us-
central1-c   8h
```

7. 运行以下命令，复制现有计算 **MachineSet** 定义并将结果输出到 JSON 文件。这将是启用了 GPU 的计算机器集定义的基础。

```
$ oc get machineset myclustername-2pt9p-worker-a -n openshift-machine-api -o json  >
<output_file.json>
```

8. 编辑 JSON 文件，对新 **MachineSet** 定义进行以下更改：

   - 通过在 **metadata.name** 中以及在两个 **machine.openshift.io/cluster-api-machineset** 实例中插入子字符串 **gpu** 来重新命名集群集**名称**。
   - 将新 **MachineSet** 定义的 **machineType** 更改为 **a2-highgpu-1g**，其中包括 NVIDIA A100 GPU。

     ```
     jq .spec.template.spec.providerSpec.value.machineType ocp_4.16_machineset-a2-
     highgpu-1g.json

     "a2-highgpu-1g"
     ```

     <output_file.json> 文件被保持为 **ocp_4.16_machineset-a2-highgpu-1g.json**。

9. 更新 **ocp_4.16_machineset-a2-highgpu-1g.json** 中的以下字段：

   - 将 **.metadata.name** 更改为包含 **gpu** 的名称。
   - 更改 **.spec.selector.matchLabels["machine.openshift.io/cluster-api-machineset"]**，以匹配新的 **.metadata.name**。
   - 更改 **.spec.template.metadata.labels["machine.openshift.io/cluster-api-machineset"]**，以匹配新的 **.metadata.name**。
   - 将 **.spec.template.spec.providerSpec.value.MachineType** 更改为 **a2-highgpu-1g**。
   - 在 **machineType** 下添加以下行：`"onHostMaintenance": "Terminate".例如：

     ```
     "machineType": "a2-highgpu-1g",
     "onHostMaintenance": "Terminate",
     ```

10. 要验证您的更改，请运行以下命令对原始计算定义和新的 GPU 节点定义执行 **diff** ：

```
$ oc get machineset/myclustername-2pt9p-worker-a -n openshift-machine-api -o json | diff
ocp_4.16_machineset-a2-highgpu-1g.json -
```

输**出示例**

```
15c15
<         "name": "myclustername-2pt9p-worker-gpu-a",
---
>         "name": "myclustername-2pt9p-worker-a",
25c25
<                 "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
---
>                 "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-a"
34c34
<                   "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
gpu-a"
---
>                   "machine.openshift.io/cluster-api-machineset": "myclustername-2pt9p-worker-
a"
59,60c59
<                       "machineType": "a2-highgpu-1g",
<                       "onHostMaintenance": "Terminate",
---
>                       "machineType": "n2-standard-4",
```

11. 运行以下命令，从定义文件创建启用了 GPU 的计算机器集：

```
$ oc create -f ocp_4.16_machineset-a2-highgpu-1g.json
```

输**出示例**

```
machineset.machine.openshift.io/myclustername-2pt9p-worker-gpu-a created
```

验证

1. 运行以下命令，查看您创建的机器集：

```
$ oc -n openshift-machine-api get machinesets | grep gpu
```

MachineSet 副本数被设置为 **1**，以便自动创建新的 **Machine** 对象。

输**出示例**

```
myclustername-2pt9p-worker-gpu-a   1         1         1       1           5h24m
```

2. 运行以下命令，查看创建机器集的 **Machine** 对象：

```
$ oc -n openshift-machine-api get machines | grep gpu
```

输**出示例**

```
myclustername-2pt9p-worker-gpu-a-wxcr6   Running   a2-highgpu-1g   us-central1   us-
central1-a   5h25m
```

注意

请注意，不需要为节点指定命名空间。节点定义是在集群范围之内。

2.5.10. 部署 Node Feature Discovery Operator

创建启用了 GPU 的节点后，您需要发现启用了 GPU 的节点，以便调度它。为此，请安装 Node Feature Discovery (NFD) Operator。NFD Operator 识别节点中的硬件设备功能。它解决了在基础架构节点中识别和目录硬件资源的一般问题，以便 OpenShift Container Platform 可以使用它们。

流程

1. 在 OpenShift Container Platform 控制台中，从 **OperatorHub** 安装 Node Feature Discovery Operator。
2. 将 NFD Operator 安装到 **OperatorHub** 后，从已安装的 Operator 列表中选择 **Node Feature Discovery**，然后选择 **Create instance**。这会在 **openshift-nfd** 命名空间中安装 **nfd-master** 和 **nfd-worker** pod，每个计算节点一个 **nfd-worker** pod。
3. 运行以下命令验证 Operator 是否已安装并正在运行：

```
$ oc get pods -n openshift-nfd
```

输**出示例**

```
NAME                                  READY   STATUS    RESTARTS   AGE

nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running 7  (8h ago)   1d
```

4. 浏览到控制台中的已安装的 Oerator，再选择 **Create Node Feature Discovery**。
5. 选择 **Create** 以构建 NFD 自定义资源。这会在 **openshift-nfd** 命名空间中创建 NFD pod，为硬件资源和目录轮询 OpenShift Container Platform 节点。

验证

1. 构建成功后，运行以下命令来验证 NFD pod 是否在每个节点上运行：

```
$ oc get pods -n openshift-nfd
```

输**出示例**

```
NAME                                  READY   STATUS      RESTARTS       AGE
nfd-controller-manager-8646fcbb65-x5qgk   2/2     Running     7 (8h ago)     12d
nfd-master-769656c4cb-w9vrv               1/1     Running     0              12d
nfd-worker-qjxb2                      1/1     Running     3 (3d14h ago)   12d
nfd-worker-xtz9b                      1/1     Running     5 (3d14h ago)   12d
```

NFD Operator 使用厂商 PCI ID 来识别节点的硬件。NVIDIA 使用 PCI ID **10de**。

2. 运行以下命令，查看 NFD Operator 发现的 NVIDIA GPU：

```
$ oc describe node ip-10-0-132-138.us-east-2.compute.internal | egrep 'Roles|pci'
```

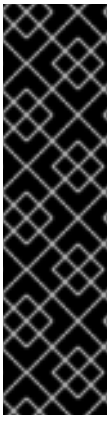
输出示例

```
Roles: worker

feature.node.kubernetes.io/pci-1013.present=true

feature.node.kubernetes.io/pci-10de.present=true

feature.node.kubernetes.io/pci-1d0f.present=true
```

10de 会出现在启用了 GPU 的节点的节点功能列表中。这意味着 NFD Operator 可以正确地识别启用了 GPU 的 MachineSet 的节点。

2.6. 在 IBM CLOUD 上创建计算机器集

您可以在 IBM Cloud® 上的 OpenShift Container Platform 集群中创建不同的计算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。



重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.6.1. IBM Cloud 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在地区中指定的 IBM Cloud® 区域中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
```

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

1 5 7 基于您在置备集群时设定的集群 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 16 要添加的节点标签。

4 6 10 基础架构 ID、节点标签和地区。

11 用于集群安装的自定义 Red Hat Enterprise Linux CoreOS(RHCOS)镜像。

12 您区域内的基础架构 ID 和区域来放置机器。确保您的地区支持您指定的区域。

13 指定 IBM Cloud® 实例配置集。

14 指定要放置机器的区域。

15 在其中放置机器资源的资源组。这是安装时指定的现有资源组，或根据基础架构 ID 命名的安装程序创建资源组。

17 VPC 名称。

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

2.6.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

   输**出示例**

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

   输**出示例**

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
```

```
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  spec:
    providerSpec: 3
      ...
```

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.7. 在 IBM POWER VIRTUAL SERVER 上创建计算机器集

您可以在 IBM Power® Virtual Server 上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.7.1. IBM Power Virtual Server 上计算机器设置自定义资源的 YAML 示例

此 YAML 文件示例定义了一个在区域中指定的 IBM Power® Virtual Server 区域中运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<region> 10
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: ""
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1
```

```
      credentialsSecret:
        name: powervs-credentials
      image:
        name: rhcos-<infrastructure_id> 11
        type: Name
      keyPairName: <infrastructure_id>-key
      kind: PowerVSMachineProviderConfig
      memoryGiB: 32
      network:
        regex: ^DHCPSERVER[0-9a-z]{32}_Private$
        type: RegEx
      processorType: Shared
      processors: "0.5"
      serviceInstance:
        id: <ibm_power_vs_service_instance_id>
        type: ID 12
      systemType: s922
      userDataSecret:
        name: <role>-user-data
```

1 5 7 基于您在置备集群时设定的集群 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 要添加的节点标签。

4 6 10 基础架构 ID、节点标签和地区。

11 用于集群安装的自定义 Red Hat Enterprise Linux CoreOS(RHCOS)镜像。

12 要放置机器的区域内的基础架构 ID。

2.7.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：

a. 要列出集群中的计算机器集，请运行以下命令：

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

输出示例

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输出示例

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

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.8. 在 NUTANIX 上创建计算机器集

您可以在 Nutanix 的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.8.1. Nutanix 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个 Nutanix 计算机器集，它创建标记为 **node-role.kubernetes.io/<role>: ""** 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 **<role>** 则是要添加的节点标签。

使用 OpenShift CLI 获取的值
在以下示例中，您可以使用 OpenShift CLI (**oc**) 获取集群的一些值。

基础架构 ID

<infrastructure_id> 字符串是基础架构 ID，它基于您在置备集群时设定的集群 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
```

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role>
  name: <infrastructure_id>-<role>-<zone> 3
  namespace: openshift-machine-api
  annotations: 4
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-<zone>
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: ""
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1
          bootType: "" 5
          categories: 6
          - key: <category_name>
            value: <category_value>
          cluster: 7
            type: uuid
            uuid: <cluster_uuid>
          credentialsSecret:
            name: nutanix-credentials
          image:
            name: <infrastructure_id>-rhcos 8
            type: name
```

```
      kind: NutanixMachineProviderConfig
      memorySize: 16Gi 9
      project: 10
        type: name
        name: <project_name>
      subnets:
      - type: uuid
        uuid: <subnet_uuid>
      systemDiskSize: 120Gi 11
      userDataSecret:
        name: <user_data_secret> 12
      vcpuSockets: 4 13
      vcpusPerSocket: 1 14
```

1. 其中 **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。
2. 指定要添加的节点标签。
3. 指定基础架构 ID、节点标签和区域。
4. 集群自动扩展的注解。
5. 指定计算机器使用的引导类型。有关引导类型的更多信息，请参阅虚拟环境中的了解 UEFI、安全引导和 TPM。有效值为 **Legacy**、**SecureBoot** 或 **UEFI**。默认值为 **Legacy**。

注意

您必须在 OpenShift Container Platform 4.16 中使用 **Legacy** 引导类型。

6. 指定一个或多个 Nutanix Prism 类别以应用到计算机器。此小节需要 **key** 和 **value** 参数代表存在于 Prism Central 中的类别的键值对。有关类别的更多信息，请参阅类别管理。
7. 指定 Nutanix Prism Element 集群配置。在本例中，集群类型是 **uuid**，因此有一个 **uuid** 小节。
8. 指定要使用的镜像。使用集群的现有默认计算机器集中的镜像。
9. 指定集群的内存量（以 Gi 为单位）。
10. 指定用于集群的 Nutanix 项目。在本例中，项目类型是 **name**，因此有一个 **name** 小节。
11. 指定系统磁盘大小（以 Gi 为单位）。
12. 指定 **openshift-machine-api** 命名空间中的用户数据 YAML 文件中的 secret 名称。安装程序在默认计算机器集中填充时使用的值。
13. 指定 vCPU 套接字数量。
14. 指定每个插槽的 vCPU 数量。

2.8.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

输**出示例**

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输**出示例**

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

```
    machine.openshift.io/cluster-api-machine-role: <role>
    machine.openshift.io/cluster-api-machine-type: <role>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
spec:
  providerSpec: 3
    ...
```

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.8.3. Nutanix 集群的故障域

要在 Nutanix 集群中添加或更新故障域配置，您必须对几个资源进行协调更改。需要以下操作：

1. 修改集群基础架构自定义资源 (CR)。
2. 修改集群 control plane 机器集 CR。
3. 修改或替换计算机器设置 CR。

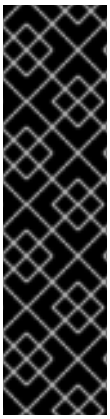
如需更多信息，请参阅 *安装后配置* 内容中的"将故障域添加到现有 Nutanix 集群"。

其他资源

- 在现有的 Nutanix 集群中添加故障域

2.9. 在 OPENSTACK 上创建计算机器集

您可以在 Red Hat OpenStack Platform (RHOSP) 上的 OpenShift Container Platform 集群中创建不同的计算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。



重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.9.1. RHOSP 上计算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) 上运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<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: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <role> 8
        machine.openshift.io/cluster-api-machine-type: <role> 9
```

```
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 10
spec:
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1alpha1
      cloudName: openstack
      cloudsSecret:
        name: openstack-cloud-credentials
        namespace: openshift-machine-api
      flavor: <nova_flavor>
      image: <glance_image_name_or_location>
      serverGroupID: <optional_UUID_of_server_group> 11
      kind: OpenstackProviderSpec
      networks: 12
      - filter: {}
        subnets:
        - filter:
            name: <subnet_name>
            tags: openshiftClusterID=<infrastructure_id> 13
      primarySubnet: <rhosp_subnet_UUID> 14
      securityGroups:
      - filter: {}
        name: <infrastructure_id>-worker 15
      serverMetadata:
        Name: <infrastructure_id>-worker 16
        openshiftClusterID: <infrastructure_id> 17
      tags:
      - openshiftClusterID=<infrastructure_id> 18
      trunk: true
      userDataSecret:
        name: worker-user-data 19
      availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 13 15 16 17 18 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 19 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。

11 要为 MachineSet 设置服务器组策略，请输入从创建服务器组返回的值。对于大多数部署，推荐使用 **anti-affinity** 或 **soft-anti-affinity** 策略。

12 部署到多个网络需要。要指定多个网络，请在网络数组中添加另一个条目。此外，您必须包含用作 **primarySubnet** 值的网络。

14 指定您要发布节点端点的 RHOSP 子网。通常，这与 **install-config.yaml** 文件中的 **machineSubnet** 值相同。

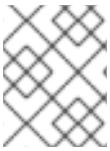
2.9.2. 在 RHOSP 上使用 SR-IOV 的计算机器设置自定义资源的 YAML 示例

如果您为单根 I/O 虚拟化(SR-IOV)配置了集群，您可以创建使用该技术的计算机器集。

此 YAML 示例定义了一个使用 SR-IOV 网络的计算机器集。它创建的节点标记为 **node-role.openshift.io/<node_role>: ""**

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。

示例假定两个名为"radio"和"uplink"的 SR-IOV 网络。网络在 **spec.template.spec.providerSpec.value.ports** 列表中的端口定义中使用。



注意

本例中仅描述特定于 SR-IOV 部署的参数。要查看更常规的示例，请参阅 "Sample YAML for a compute machine set custom resource on RHOSP"。

使用 SR-IOV 网络的计算机器集示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
    spec:
      metadata:
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: <optional_UUID_of_server_group>
          kind: OpenstackProviderSpec
          networks:
            - subnets:
```

```
    - UUID: <machines_subnet_UUID>
  ports:
   - networkID: <radio_network_UUID> 1
     nameSuffix: radio
     fixedIPs:
      - subnetID: <radio_subnet_UUID> 2
     tags:
      - sriov
      - radio
     vnicType: direct 3
     portSecurity: false 4
   - networkID: <uplink_network_UUID> 5
     nameSuffix: uplink
     fixedIPs:
      - subnetID: <uplink_subnet_UUID> 6
     tags:
      - sriov
      - uplink
     vnicType: direct 7
     portSecurity: false 8
  primarySubnet: <machines_subnet_UUID>
  securityGroups:
  - filter: {}
    name: <infrastructure_id>-<node_role>
  serverMetadata:
    Name: <infrastructure_id>-<node_role>
    openshiftClusterID: <infrastructure_id>
  tags:
  - openshiftClusterID=<infrastructure_id>
  trunk: true
  userDataSecret:
    name: <node_role>-user-data
  availabilityZone: <optional_openstack_availability_zone>
```

1 5 输入每个端口的网络 UUID。

2 6 输入每个端口的子网 UUID。

3 7 对于每个端口，**vnicType** 参数的值必须为 **direct**。

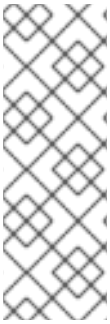
4 8 每个端口的 **portSecurity** 参数的值必须是 **false**。

禁用端口安全性时，您无法为端口设置安全组和允许的地址对。在实例上设置安全组会将组应用到连接的所有端口。

重要

部署支持 SR-IOV 的计算机器后，您必须标记它们，如：例如，在命令行中输入：

```
$ oc label node <NODE_NAME> feature.node.kubernetes.io/network-
sriov.capable="true"
```



注意

对于由网络和子网列表中的条目创建的端口，启用中继（Trunking）。从这些列表中创建的端口名称遵循 **<machine_name>-<nameSuffix>** 模式。端口定义中需要 **nameSuffix** 字段。

您可以为每个端口启用中继。

另外，您还可以在端口中添加标签作为其标签（**tags**）列表的一部分。

其他资源

- 准备在 OpenStack 上安装使用 SR-IOV 或 OVS-DPDK 的集群

2.9.3. 禁用端口安全性的 SR-IOV 部署的 YAML 示例

要在禁用端口安全性的网络上创建单根 I/O 虚拟化 (SR-IOV) 端口，请定义一个计算机器集，将端口作为 **spec.template.spec.providerSpec.value.ports** 列表中的项目包含。与标准 SR-IOV 计算机器集的区别在于，自动安全组以及使用网络和子网接口创建的端口允许的地址对配置。

您为机器子网定义的端口需要：

- API 和入口虚拟 IP 端口允许的地址对
- 计算安全组
- 附加到机器网络和子网

注意

本例中仅描述特定于禁用端口安全性的 SR-IOV 部署的参数。要查看更常规的示例，请参阅 RHOSP 上使用 SR-IOV 的计算机器设置自定义资源的 Sample YAML"。

使用 SR-IOV 网络并禁用端口安全性的计算机器集示例

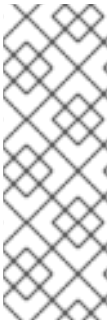
```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-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

```
    machine.openshift.io/cluster-api-machine-type: <node_role>
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1alpha1
      cloudName: openstack
      cloudsSecret:
        name: openstack-cloud-credentials
        namespace: openshift-machine-api
      flavor: <nova_flavor>
      image: <glance_image_name_or_location>
      kind: OpenstackProviderSpec
      ports:
        - allowedAddressPairs: (1)
          - ipAddress: <API_VIP_port_IP>
          - ipAddress: <ingress_VIP_port_IP>
          fixedIPs:
            - subnetID: <machines_subnet_UUID> (2)
          nameSuffix: nodes
          networkID: <machines_network_UUID> (3)
          securityGroups:
              - <compute_security_group_UUID> (4)
        - networkID: <SRIOV_network_UUID>
          nameSuffix: sriov
          fixedIPs:
            - subnetID: <SRIOV_subnet_UUID>
          tags:
            - sriov
          vnicType: direct
          portSecurity: False
      primarySubnet: <machines_subnet_UUID>
      serverMetadata:
        Name: <infrastructure_ID>-<node_role>
        openshiftClusterID: <infrastructure_id>
      tags:
      - openshiftClusterID=<infrastructure_id>
      trunk: false
      userDataSecret:
        name: worker-user-data
```

(1) 为 API 和入口端口指定允许的地址对。

(2) (3) 指定机器网络和子网。

(4) 指定计算机器安全组。



注意

对于由网络和子网列表中的条目创建的端口，启用中继（Trunking）。从这些列表中创建的端口名称遵循 **<machine_name>-<nameSuffix>** 模式。端口定义中需要 **nameSuffix** 字段。

您可以为每个端口启用中继。

另外，您还可以在端口中添加标签作为其标签（**tags**）列表的一部分。

2.9.4. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

   输**出示例**

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

   输**出示例**

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

```
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
        ...
```

1. 集群基础架构 ID。
2. 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3. 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.10. 在 VSPHERE 上创建计算机器设置

您可以在 VMware vSphere 上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.10.1. vSphere 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 VMware vSphere 上运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<role>** 则是要添加的节点标签。

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

```
    machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
spec:
  metadata:
    creationTimestamp: null
    labels:
      node-role.kubernetes.io/<role>: "" 9
  providerSpec:
    value:
      apiVersion: vsphereprovider.openshift.io/v1beta1
      credentialsSecret:
        name: vsphere-cloud-credentials
      diskGiB: 120
      kind: VSphereMachineProviderSpec
      memoryMiB: 8192
      metadata:
        creationTimestamp: null
      network:
        devices:
        - networkName: "<vm_network_name>" 10
      numCPUs: 4
      numCoresPerSocket: 1
      snapshot: ""
      template: <vm_template_name> 11
      userDataSecret:
        name: worker-user-data
      workspace:
        datacenter: <vcenter_datacenter_name> 12
        datastore: <vcenter_datastore_name> 13
        folder: <vcenter_vm_folder_path> 14
        resourcepool: <vsphere_resource_pool> 15
        server: <vcenter_server_ip> 16
```

1 3 5 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（**oc**）软件包，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 和节点标签。

6 7 9 指定要添加的节点标签。

10 指定要将计算机器设置为的 vSphere VM 网络。此虚拟机网络必须是集群中其他计算机器所处的位置。

11 指定要使用的 vSphere 虚拟机模板，如 **user-5ddjd-rhcos**。

12 指定要将计算机器设置为的 vCenter Datacenter。

13 指定要部署计算机器的 vCenter Datastore。

14 指定 vCenter 中 vSphere 虚拟机文件夹的路径，如 **/dc1/vm/user-inst-5ddjd**。

15 指定虚拟机的 vSphere 资源池。

16 指定 vCenter 服务器 IP 或完全限定域名。

2.10.2. 计算机器设置管理的最低 vCenter 权限

要在 vCenter 上的 OpenShift Container Platform 集群中管理计算机器，您必须使用一个具有特权的帐户来读取、创建和删除所需资源。使用具有全局管理特权的帐户是访问所有所需权限的最简单方法。

如果无法使用具有全局管理权限的帐户，您必须创建一个角色来授予最低所需的权限。下表列出了创建、扩展和删除 OpenShift Container Platform 集群中的机器所需的最低 vCenter 角色和特权。

例 2.1. 计算机器设置管理所需的最低 vCenter 角色和权限

适用于角色的 vSphere 对象	必要时	所需的权限
vSphere vCenter	Always	**InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.Update**[1] **StorageProfile.View**[1]
vSphere vCenter 集群	Always	**Resource.AssignVMToPool**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse**
vSphere 端口组	Always	**Network.Assign**

适用于角色的 vSphere 对象	必要时	所需的权限
虚拟机文件夹	Always	**VirtualMachine.Config.AddRemoveDevice** **VirtualMachine.Config.AdvancedConfig** **VirtualMachine.Config.Annotation** **VirtualMachine.Config.CPUCount** **VirtualMachine.Config.DiskExtend** **VirtualMachine.Config.Memory** **VirtualMachine.Config.Settings** **VirtualMachine.Interact.PowerOff** **VirtualMachine.Interact.PowerOn** **VirtualMachine.Inventory.CreateFromExisting** **VirtualMachine.Inventory.Delete** **VirtualMachine.Provisioning.Clone**
vSphere vCenter Datacenter	如果安装程序创建虚拟机文件夹	**Resource.AssignVMToPool** **VirtualMachine.Provisioning.DeployTemplate**

[1] **StorageProfile.Update** 和 **StorageProfile.View** 权限只适用于使用 Container Storage Interface(CSI) 的存储后端。

下表详细介绍了计算机器设置管理所需的权限和传播设置。

例 2.2. 所需的权限和传播设置

vSphere object	文件夹类型	传播到子对象	所需的权限
vSphere vCenter	Always	不是必需的	列出所需的权限
vSphere vCenter Datacenter	现有文件夹	不是必需的	**只读** 权限
	安装程序创建文件夹	必填	列出所需的权限
vSphere vCenter 集群	Always	必填	列出所需的权限

vSphere object	文件夹类型	传播到子对象	所需的权限
vSphere vCenter 数据存储	Always	不是必需的	列出所需的权限
vSphere Switch	Always	不是必需的	**只读** 权限
vSphere 端口组	Always	不是必需的	列出所需的权限
vSphere vCenter 虚拟机文件夹	现有文件夹	必填	列出所需的权限

有关只使用所需权限创建帐户的更多信息，请参阅 vSphere 文档中的 vSphere 权限和用户管理任务。

2.10.3. 具有用户置备基础架构的集群的要求以使用计算机器集

要在具有用户置备的基础架构的集群中使用计算机器集，您必须确保集群配置支持使用 Machine API。

获取基础架构 ID
要创建计算机器集，您必须能够为集群提供基础架构 ID。

流程

- 要获取集群的基础架构 ID，请运行以下命令：

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

满足 vSphere 凭证要求
要使用计算机器集，Machine API 必须能够与 vCenter 交互。授权 Machine API 组件与 vCenter 交互的凭证必须存在于 **openshift-machine-api** 命名空间中的 secret 中。

流程

1. 要确定所需的凭证是否存在，请运行以下命令：

```
$ oc get secret \
  -n openshift-machine-api vsphere-cloud-credentials \
  -o go-template='{{range $k,$v := .data}}{{printf "%s: " $k}}{{if not $v}}{{$v}}{{else}}{{$v | base64decode}}{{end}}{{"\n"}}{{end}}'
```

输出示例

```
<vcenter-server>.password=<openshift-user-password>
<vcenter-server>.username=<openshift-user>
```

其中 **<vcenter-server>** 是 vCenter 服务器的 IP 地址或完全限定域名(FQDN)， **<openshift-user>** 和 **<openshift-user-password>** 是要使用的 OpenShift Container Platform 管理员凭证。

2. 如果 secret 不存在，请运行以下命令来创建它：

```
$ oc create secret generic vsphere-cloud-credentials \
  -n openshift-machine-api \
  --from-literal=<vcenter-server>.username=<openshift-user> --from-literal=<vcenter-server>.password=<openshift-user-password>
```

满足 Ignition 配置要求

置备虚拟机 (VM) 需要有效的 Ignition 配置。Ignition 配置包含 **machine-config-server** 地址和系统信任捆绑包，用于从 Machine Config Operator 获取进一步的 Ignition 配置。

默认情况下，此配置存储在 **machine-api-operator** 命名空间中的 **worker-user-data** secret 中。计算机器集在机器创建过程中引用 secret。

流程

1. 要确定所需的 secret 是否存在，请运行以下命令：

```
$ oc get secret \
  -n openshift-machine-api worker-user-data \
  -o go-template='{{range $k,$v := .data}}{{printf "%s: " $k}}{{if not $v}}{{$v}}{{else}}{{$v | base64decode}}{{end}}{{"\n"}}{{end}}'
```

输**出示例**

```
disableTemplating: false
userData: 1
  {
    "ignition": {
      ...
      },
    ...
  }
```

1 此处省略了完整输出，但应该具有此格式。

2. 如果 secret 不存在，请运行以下命令来创建它：

```
$ oc create secret generic worker-user-data \
  -n openshift-machine-api \
  --from-file=<installation_directory>/worker.ign
```

其中 **<installation_directory>** 是在集群安装过程中存储安装资产的目录。

其他资源

- 了解 Machine Config Operator
- 安装 RHCOS 并启动 OpenShift Container Platform bootstrap 过程

2.10.4. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的来动态管理您选择的特定工作负载的机器计算资源。

注意

使用用户置备的基础架构安装的集群有不同的网络堆栈，与带有安装程序置备的基础架构的集群不同。因此，具有用户置备的基础架构的集群中不支持自动负载均衡器管理。对于这些集群，计算机器集只能创建 **worker** 和 **infra** 类型的机器。

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**。
- 具有在 vCenter 实例中部署虚拟机所需的权限，并对指定的数据存储具有所需的访问权限。
- 如果您的集群使用用户置备的基础架构，则代表已满足该配置的特定 Machine API 要求。

流程

1. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

   输出示例

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

   输出示例

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

```
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
        ...
```

1 集群基础架构 ID。

2 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

c. 如果您要为具有用户置备的基础架构的集群创建计算机器集，请注意以下重要值：

vSphere providerSpec 值示例

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
...
template:
  ...
  spec:
    providerSpec:
      value:
        apiVersion: machine.openshift.io/v1beta1
        credentialsSecret:
          name: vsphere-cloud-credentials 1
        diskGiB: 120
        kind: VSphereMachineProviderSpec
        memoryMiB: 16384
        network:
          devices:
            - networkName: "<vm_network_name>"
```

```
    numCPUs: 4
    numCoresPerSocket: 4
    snapshot: ""
    template: <vm_template_name> 2
    userDataSecret:
      name: worker-user-data 3
    workspace:
      datacenter: <vcenter_datacenter_name>
      datastore: <vcenter_datastore_name>
      folder: <vcenter_vm_folder_path>
      resourcepool: <vsphere_resource_pool>
      server: <vcenter_server_address> 4
```

1. 包含所需 vCenter 凭证的 **openshift-machine-api** 命名空间中的 secret 名称。
2. 安装期间创建的集群的 RHCOS 虚拟机模板名称。
3. **openshift-machine-api** 命名空间中的 secret 名称，其中包含所需的 Ignition 配置凭证。
4. vCenter 服务器的 IP 地址或完全限定域名 (FQDN)。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

2.10.5. 使用机器集向机器添加标签

OpenShift Container Platform 为其创建的每个虚拟机 (VM) 添加特定于集群的标签。安装程序使用这些标签在卸载集群时选择要删除的虚拟机。

除了分配给虚拟机的特定于集群的标签外，您还可以将机器集配置为向它置备的虚拟机中添加最多 10 个额外的 vSphere 标签。

先决条件

- 您可以使用具有 **cluster-admin** 权限的账户访问在 vSphere 上安装的 OpenShift Container Platform 集群。
- 您可以访问与集群关联的 VMware vCenter 控制台。
- 您已在 vCenter 控制台中创建了标签。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 使用 vCenter 控制台查找您要添加到机器的任何标签的标签 ID：
   a. 登录到 vCenter 控制台。
   b. 在 **Home** 菜单中，点 **Tags & Custom Attributes**。
   c. 选择您要添加到机器的标签。
   d. 使用您选择的标签的浏览器 URL 来识别标签 ID。

   标签 **URL 示例**

   > https://vcenter.example.com/ui/app/tags/tag/urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL/permissions

   标签 **ID 示例**

   > urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL
2. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
3. 编辑 **providerSpec** 字段中的以下行：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          tagIDs: 1
          - <tag_id_value> 2
# ...
```

1. 指定要添加到此机器集置备的机器上最多 10 个标签列表。
2. 指定要添加到机器的标签值。例如，**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL**.

2.11. 在裸机上创建计算机器集

您可以在裸机上的 OpenShift Container Platform 集群中创建不同的计算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。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

2.11.1. 裸机上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在裸机上运行的计算机器集，并创建通过 **node-role.kubernetes.io/<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 **<role>** 则是要添加的节点标签。

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

```
      value:
        apiVersion: baremetal.cluster.k8s.io/v1alpha1
        hostSelector: {}
        image:
          checksum: http:/172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2.<md5sum>
(10)
          url: http://172.22.0.3:6181/images/rhcos-<version>.<architecture>.qcow2 (11)
        kind: BareMetalMachineProviderSpec
        metadata:
          creationTimestamp: null
        userData:
          name: worker-user-data
```

(1) (3) (5) 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（**oc**）软件包，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 和节点标签。

(6) (7) (9) 指定要添加的节点标签。

(10) 编辑 **校验和** URL，以使用 API VIP 地址。

(11) 编辑 **URL** 以使用 API VIP 地址。

2.11.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

   输出示例

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输出示例

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

1 集群基础架构 ID。

2 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

第 3 章 手动扩展计算机器集

您可以在计算机器集中添加或删除机器实例。

注意

如果您需要在扩展之外修改计算机器集的各个方面，请参阅修改计算机器集。

3.1. 先决条件

- 如果启用了集群范围代理并扩展未包含在安装配置的 **networking.machineNetwork[].cidr** 中的计算机器，您必须将计算机器添加到 Proxy 对象的 **noProxy** 字段，以防止连接问题。

重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

3.2. 手动扩展计算机器集

要在计算机器集中添加或删除机器实例，您可以手动扩展计算机器集。

这个指南与全自动的、安装程序置备的基础架构安装相关。自定义的、用户置备的基础架构安装没有计算机器集。

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. 运行以下命令，查看集群中的计算机器：

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

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

2. 运行以下命令，查看集群中的计算机器：

   ```
   $ oc get machines.machine.openshift.io -n openshift-machine-api
   ```

3. 运行以下命令，在要删除的计算机器上设置注解：

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

4. 运行以下命令来扩展计算机器集：

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

或者：

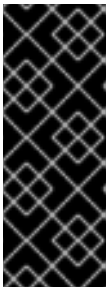
```
$ oc edit machinesets.machine.openshift.io <machineset> -n openshift-machine-api
```

提示

您还可以应用以下 YAML 来扩展计算机器集：

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

您可以扩展或缩减计算机器。需要过几分钟以后新机器才可用。



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 中断预算，排空操作可能无法成功。如果排空操作失败，机器控制器无法继续删除机器。

您可以通过在特定机器上注解 **machine.openshift.io/exclude-node-draining** 来跳过排空节点。

验证

- 运行以下命令，验证删除所需的机器：

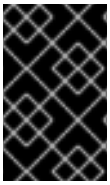
```
$ oc get machines.machine.openshift.io
```

3.3. 计算机器集删除策略

Random、**Newest** 和 **Oldest** 是三个支持的删除选项。默认值为 **Random**，表示在扩展计算机器时随机选择并删除机器。通过修改特定的计算机器集，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/delete-machine=true** 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 计算机器集扩展为 **0**。

注意

对于需要特定节点运行的用例，可以使用自定义计算机器集，在 worker 计算机器集缩减时，控制器会忽略这些服务。这可防止服务被中断。

3.4. 其他资源

- 机器删除阶段的生命周期 hook

第 4 章 修改计算机器集

您可以修改计算机器集，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计算机器设置，请参阅手动扩展计算机器集。

4.1. 使用 CLI 修改计算机器集

当您修改计算机器集时，您的更改只适用于在保存更新的 **MachineSet** 自定义资源 (CR) 后创建的计算机器。更改不会影响现有的机器。您可以通过扩展计算机器集来将现有机器替换为反映更新的配置的新机器。

如果您需要在不进行其他更改的情况下扩展计算机器，则不需要删除机器。

注意

默认情况下，OpenShift Container Platform 路由器 Pod 部署在计算机器上。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 计算机器集扩展为 **0**。

此流程中的输出示例使用 AWS 集群的值。

先决条件

- OpenShift Container Platform 集群使用 Machine API。
- 以管理员身份使用 OpenShift CLI (**oc**) 登录集群。

流程

1. 运行以下命令列出集群中的计算机器集：

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

输**出示例**

```
NAME                        DESIRED   CURRENT   READY   AVAILABLE   AGE
<compute_machine_set_name_1>  1       1         1       1           55m
<compute_machine_set_name_2>  1       1         1       1           55m
```

2. 运行以下命令来编辑计算机器集：

```
$ oc edit machinesets.machine.openshift.io <machine_set_name> \
  -n openshift-machine-api
```

3. 请注意 **spec.replicas** 字段的值，因为在扩展机器集时需要它来应用更改。

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

```
  name: <machine_set_name>
  namespace: openshift-machine-api
spec:
  replicas: 2 (1)
# ...
```

(1) 此流程中的示例显示具有 **replicas** 值 **2** 的计算机器集。

4. 使用您想要的配置选项来更新计算机器设置 CR，并保存您的更改。

5. 运行以下命令，列出由更新的计算机器集管理的机器：

```
$ oc get machines.machine.openshift.io \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

AWS 集群的输出示例

```
NAME                        PHASE    TYPE       REGION     ZONE       AGE
<machine_name_original_1>   Running  m6i.xlarge  us-west-1  us-west-1a  4h
<machine_name_original_2>   Running  m6i.xlarge  us-west-1  us-west-1a  4h
```

6. 对于由更新的计算机器集管理的每台机器，请运行以下命令设置 **delete** 注解：

```
$ oc annotate machine.machine.openshift.io/<machine_name_original_1> \
  -n openshift-machine-api \
  machine.openshift.io/delete-machine="true"
```

7. 要使用新配置创建替换机器，请运行以下命令将计算机器设置为两倍：

```
$ oc scale --replicas=4 \ (1)
  machineset.machine.openshift.io <machine_set_name> \
  -n openshift-machine-api
```

(1) 原始示例值 **2** 加倍到 **4**。

8. 运行以下命令，列出由更新的计算机器集管理的机器：

```
$ oc get machines.machine.openshift.io \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

AWS 集群的输出示例

```
NAME                        PHASE        TYPE       REGION     ZONE       AGE
<machine_name_original_1>   Running      m6i.xlarge  us-west-1  us-west-1a  4h
<machine_name_original_2>   Running      m6i.xlarge  us-west-1  us-west-1a  4h
<machine_name_updated_1>    Provisioned  m6i.xlarge  us-west-1  us-west-1a  55s
<machine_name_updated_2>    Provisioning m6i.xlarge  us-west-1  us-west-1a  55s
```

当新机器处于 **Running** 阶段时，您可以将计算机器设置为原始副本数。

9. 要删除使用旧配置创建的机器，请运行以下命令将计算机器设置为原始副本数：

```
$ oc scale --replicas=2 \ 1
  machineset.machine.openshift.io <machine_set_name> \
  -n openshift-machine-api
```

1 原始示例值 **2**。

验证

- 要验证更新机器集创建的机器是否具有正确的配置，请运行以下命令检查 CR 中的相关字段是否有新机器：

```
$ oc describe machine.machine.openshift.io <machine_name_updated_1> \
  -n openshift-machine-api
```

- 要验证没有更新的配置的计算机器，请运行以下命令列出由更新的计算机器集管理的机器：

```
$ oc get machines.machine.openshift.io \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machineset=<machine_set_name>
```

删除时的输出示例是 AWS 集群的进度

```
NAME                        PHASE        TYPE       REGION      ZONE       AGE
<machine_name_original_1>   Deleting     m6i.xlarge   us-west-1   us-west-1a   4h
<machine_name_original_2>   Deleting     m6i.xlarge   us-west-1   us-west-1a   4h
<machine_name_updated_1>    Running      m6i.xlarge   us-west-1   us-west-1a   5m41s
<machine_name_updated_2>    Running      m6i.xlarge   us-west-1   us-west-1a   5m41s
```

为 AWS 集群完成删除时的输出示例

```
NAME                        PHASE        TYPE       REGION      ZONE       AGE
<machine_name_updated_1>    Running      m6i.xlarge   us-west-1   us-west-1a   6m30s
<machine_name_updated_2>    Running      m6i.xlarge   us-west-1   us-west-1a   6m30s
```

其他资源

- 机器删除阶段的生命周期 hook
- 手动扩展计算机器集
- 使用调度程序控制 pod 放置

第 5 章 机器阶段和生命周期

机器会经历一个*生命周期*，它包括了多个定义的阶段。了解机器生命周期及其各个阶段可帮助您验证流程是否完成或排除不必要的行为。对于 OpenShift Container Platform，在所有支持的云供应商中的机器生命周期是一致的。

5.1. 机器阶段

当机器进入其生命周期时，它会经历不同的阶段。每个阶段都是机器状态的基本表示。

置备

置备新机器的请求。机器尚不存在，且没有实例、提供程序 ID 或地址。

已置备

机器存在，并且具有供应商 ID 或地址。云供应商为机器创建了实例。机器还没有成为节点，机器对象的 **status.nodeRef** 部分还没有被填充。

Running

机器存在，并且具有供应商 ID 或地址。Ignition 成功运行，集群机器批准人已批准证书签名请求 (CSR)。机器已变为节点，机器对象的 **status.nodeRef** 部分包含节点详情。

删除

有一个删除机器的请求。machine 对象有一个 **DeletionTimestamp** 字段，代表删除请求的时间。

Failed

机器存在不可恢复的问题。例如，如果云供应商删除了机器的实例，可能会发生这种情况。

5.2. 机器生命周期

生命周期从置备机器的请求开始，直到机器不再存在为止。

机器生命周期所经历的顺序如下。此概述不包括因为错误或生命周期 hook 造成的中断。

1. 由于以下原因之一，需要置备新机器请求：
   - 集群管理员扩展机器集，使其需要额外的机器。
   - 自动扩展策略扩展机器集，使其需要额外的机器。
   - 由机器集管理的机器失败或被删除，机器集会创建一个替换来维护所需的机器数量。
2. 机器进入 **Provisioning** 阶段。
3. 基础架构供应商为机器创建一个实例。
4. 机器具有供应商 ID 或地址，并输入 **Provisioned** 阶段。
5. 处理 Ignition 配置文件。
6. kubelet 发布证书签名请求 (CSR)。
7. 集群机器批准者批准 CSR。
8. 机器变为节点，进入 **Running** 阶段。
9. 由于以下原因之一，现有机器被移除：

- 具有 **cluster-admin** 权限的用户使用 **oc delete machine** 命令。
- 机器获取 **machine.openshift.io/delete-machine** 注解。
- 管理机器的机器集会标记它，以减少副本数作为协调的一部分。
- 集群自动扩展会识别一个不需要满足集群的部署需求的节点。
- 机器健康检查被配置为替换不健康的机器。

10. 机器进入 **Deleting** 阶段，在其中标记为删除，但仍然存在于 API 中。
11. 机器控制器从基础架构供应商中删除实例。
12. 机器控制器会删除 **Node** 对象。

5.3. 确定机器的阶段

您可以使用 OpenShift CLI (**oc**) 或使用 Web 控制台来查找机器的阶段。您可以使用这些信息来验证流程是否已完成，或排除不需要的行为。

5.3.1. 使用 CLI 确定机器的阶段

您可以使用 OpenShift CLI (**oc**) 找到机器的阶段。

先决条件

- 可以使用具有 **cluster-admin** 权限的账户访问 OpenShift Container Platform 集群。
- 已安装 **oc** CLI。

流程

- 运行以下命令列出集群中的机器：

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

输**出示例**

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
mycluster-5kbsp-master-0            Running   m6i.xlarge  us-west-1   us-west-1a   4h55m
mycluster-5kbsp-master-1            Running   m6i.xlarge  us-west-1   us-west-1b   4h55m
mycluster-5kbsp-master-2            Running   m6i.xlarge  us-west-1   us-west-1a   4h55m
mycluster-5kbsp-worker-us-west-1a-fmx8t   Running   m6i.xlarge  us-west-1   us-west-1a
4h51m
mycluster-5kbsp-worker-us-west-1a-m889l   Running   m6i.xlarge  us-west-1   us-west-1a
4h51m
mycluster-5kbsp-worker-us-west-1b-c8qzm   Running   m6i.xlarge  us-west-1   us-west-1b
4h51m
```

输出的 **PHASE** 列包含每台机器的阶段。

5.3.2. 使用 Web 控制台确定机器的阶段

您可以使用 OpenShift Container Platform Web 控制台查找机器的阶段。

先决条件

- 可以使用具有 **cluster-admin** 权限的账户访问 OpenShift Container Platform 集群。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录 Web 控制台。
2. 进入到 **Compute → Machines**。
3. 在 **Machines** 页面中，选择要查找阶段的机器的名称。
4. 在 **Machine details** 页面中，选择 **YAML** 选项卡。
5. 在 YAML 块中，找到 **status.phase** 字段的值。

   YAML 片断示例

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  name: mycluster-5kbsp-worker-us-west-1a-fmx8t
# ...
status:
  phase: Running 1
```

   1 在本例中，阶段为 **Running**。

5.4. 其他资源

- 机器删除阶段的生命周期 hook

第 6 章 删除机器

您可以删除特定的机器。

6.1. 删除一个特定的机器

您可以删除特定的机器。

重要

除非集群使用 control plane 机器集，否则不要删除 control plane 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

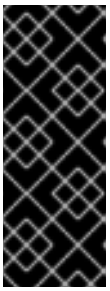
1. 运行以下命令，查看集群中的机器：

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

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

2. 找到您要删除的机器。
3. 运行以下命令来删除机器：

```
$ 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 中断预算，排空操作可能无法成功。如果排空操作失败，机器控制器无法继续删除机器。

您可以通过在特定机器上注解 **machine.openshift.io/exclude-node-draining** 来跳过排空节点。

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。

6.2. 机器删除阶段的生命周期 HOOK

机器生命周期 hook 协调一个集群的协调生命周期中的一个点，在其可以中断正常的生命周期。在机器 **Deleting** 阶段，这些中断为组件提供了修改机器删除过程的机会。

6.2.1. 术语和定义

要了解机器删除阶段的生命周期 hook 行为，您必须了解以下概念：

协调

协调是控制器尝试使集群实际状态及其组成的对象与对象规格中的要求匹配的过程。

机器控制器

机器控制器管理机器的协调生命周期。对于云平台上的机器，机器控制器是 OpenShift Container Platform 控制器和云供应商的特定操作器的组合。
在删除机器的情况下，机器控制器执行以下操作：

- 排空机器支持的节点。
- 从云供应商中删除机器实例。
- 删除 **Node** 对象。

生命周期 hook

生命周期 hook 是解决正常生命周期对象的协调生命周期中定义的点。组件可以使用生命周期 hook 将更改注入进程，以完成所需的结果。
机器 **删除** 阶段有两个生命周期 hook：

- 在机器支持的节点可以排空前，必须先解决 **preDrain** 生命周期 hook。
- 在从基础架构供应商中删除实例前，必须先解决 **preTerminate** 生命周期 hook。

hook 实现控制器

hook 实现控制器是机器控制器以外的控制器，可以与生命周期 hook 交互。hook 实现控制器可以执行以下操作之一或多个操作：

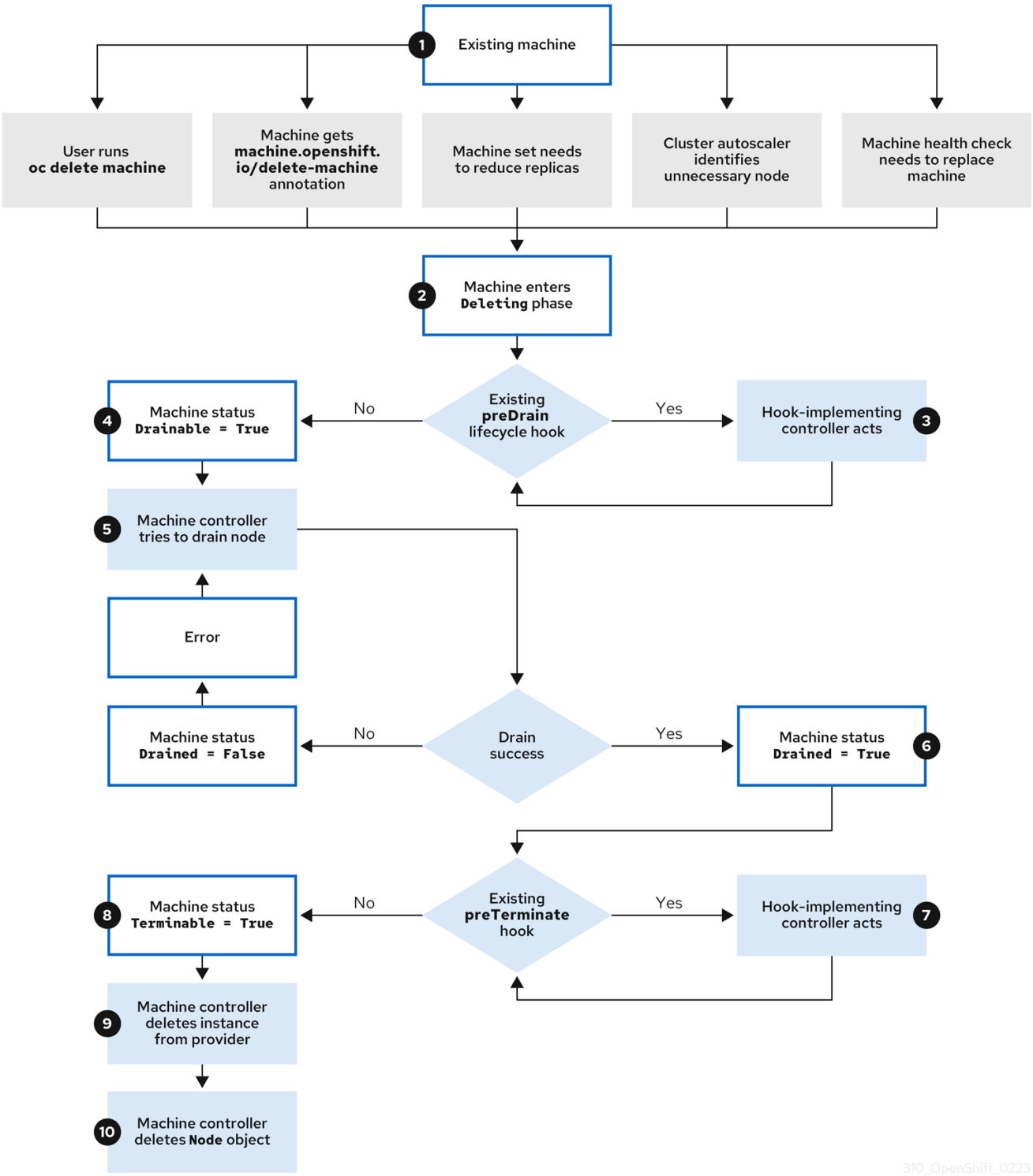
- 添加生命周期 hook。
- 响应生命周期 hook。
- 删除生命周期 hook。

每个生命周期 hook 都有一个 hook 实现控制器，但 hook 实现控制器可以管理一个或多个 hook。

6.2.2. 机器删除处理顺序

在 OpenShift Container Platform 4.16 中，机器删除阶段有两个生命周期 hook：**preDrain** 和 **preTerminate**。删除给定生命周期点的所有 hook 时，协调会正常进行。

图 6.1. 机器删除流



机器 **Deleting** 阶段按以下顺序进行：

1. 由于以下原因之一，现有机器被移除：
   - 具有 **cluster-admin** 权限的用户使用 **oc delete machine** 命令。
   - 机器获取 **machine.openshift.io/delete-machine** 注解。
   - 管理机器的机器集会标记它，以减少副本数作为协调的一部分。
   - 集群自动扩展会识别一个不需要满足集群的部署需求的节点。

- 机器健康检查被配置为替换不健康的机器。

2. 机器进入 **Deleting** 阶段，在其中标记为删除，但仍然存在于 API 中。
3. 如果存在 **preDrain** 生命周期 hook，则 hook 实现了控制器来管理它执行指定操作。
   在满足所有 **preDrain** 生命周期 hook 前，机器状态条件 **Drainable** 被设置为 **False**。
4. 没有未解析的 **preDrain** 生命周期 hook，机器状态条件 **Drainable** 被设置为 **True**。
5. 机器控制器尝试排空由机器支持的节点。
   - 如果排空失败，**Drained** 被设置为 **False**，机器控制器会尝试再次排空该节点。
   - 如果排空成功，**Drained** 被设置为 **True**。
6. 机器状态条件 **Drained** 设置为 **True**。
7. 如果存在 **preTerminate** 生命周期 hook，则管理它的 hook 实现控制器将执行指定操作。
   在满足所有 **preTerminate** 生命周期 hook 前，机器状态条件 **Terminable** 被设置为 **False**。
8. 没有未解析的 **preTerminate** 生命周期 hook，机器状态条件 **Terminable** 被设置为 **True**。
9. 机器控制器从基础架构供应商中删除实例。
10. 机器控制器会删除 **Node** 对象。

6.2.3. 删除生命周期 hook 配置

以下 YAML 片段演示了机器集内删除生命周期 hook 配置的格式和放置：

YAML 片断展示了一个 preDrain 生命周期 hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain:
    - name: <hook_name> 1
      owner: <hook_owner> 2
  ...
```

1. **preDrain** 生命周期 hook 的名称。
2. 管理 **preDrain** 生命周期 hook 的 hook 实施控制器。

YAML 片段展示了一个 preTerminate 生命周期 hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
```

```
lifecycleHooks:
  preTerminate:
  - name: <hook_name> 1
    owner: <hook_owner> 2
...
```

1 **preTerminate** 生命周期 hook 的名称。

2 管理 **preTerminate** 生命周期 hook 的 hook 实施控制器。

生命周期 hook 配置示例
以下示例演示了中断机器删除过程的多个特殊生命周期 hook 的实现：

生命周期 hook 配置示例

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain: 1
    - name: MigrateImportantApp
      owner: my-app-migration-controller
    preTerminate: 2
    - name: BackupFileSystem
      owner: my-backup-controller
    - name: CloudProviderSpecialCase
      owner: my-custom-storage-detach-controller 3
    - name: WaitForStorageDetach
      owner: my-custom-storage-detach-controller
  ...
```

1 包含单一生命周期 hook 的 **preDrain** 生命周期 hook 小节。

2 包含三个生命周期 hook 的 **preTerminate** 生命周期 hook 小节。

3 管理两个 **preTerminate** 生命周期 hook 的 hook 实现控制器：**CloudProviderSpecialCase** 和 **WaitForStorageDetach**。

6.2.4. Operator 开发人员的机器删除生命周期 hook 示例

Operator 可以在机器删除阶段使用生命周期 hook 来修改机器删除过程。以下示例演示了 Operator 可以使用此功能的方法。

preDrain 生命周期 hook 用例示例

主动替换机器

Operator 可以使用 **preDrain** 生命周期 hook 来确保，在删除一个已删除机器的实例前，替代的集群已成功创建并加入集群。这可降低机器替换或不立即初始化的替换实例期间中断的影响。

实现**自定义排空逻辑**

Operator 可以使用 **preDrain** 生命周期 hook 将机器控制器排空逻辑替换为不同的排空控制器。通过替换排空逻辑，Operator 会具有更大的灵活性并控制每个节点的工作负载生命周期。
例如，机器控制器排空库不支持排序，但自定义排空供应商可以提供此功能。通过使用自定义排空供应商，Operator 可以在排空节点前优先选择移动关键任务应用程序，以确保在集群容量有限的情况下服务中断最小化。

preTerminate 生命周期 hook 用例示例

验证存储分离

Operator 可以使用 **preTerminate** 生命周期 hook 来确保，附加到机器的存储在从基础架构供应商中移除前已分离。

提高日志可靠性

节点排空后，日志导出器守护进程需要一些时间才能将日志同步到集中式日志记录系统。
日志记录 Operator 可以使用 **preTerminate** 生命周期 hook，在节点排空的时间和机器从基础架构供应商中删除的时间直接添加一个延迟。此延迟为 Operator 提供了一个时间，以确保主工作服务被删除，且不再被添加到日志中。如果没有将新数据添加到日志，日志导出器可以在同步过程中捕获，从而确保捕获所有应用程序日志。

6.2.5. 使用机器生命周期 hook 进行仲裁保护

对于使用 Machine API Operator 的 OpenShift Container Platform 集群，etcd Operator 使用机器删除阶段的生命周期 hook 来实现仲裁保护机制。

通过使用 **preDrain** 生命周期 hook，etcd Operator 可以控制 control plane 机器上的 pod 排空和删除的时间。为了保护 etcd 仲裁，etcd Operator 会阻止删除 etcd 成员，直到该成员迁移到集群中的新节点。

此机制允许 etcd Operator 对 etcd 仲裁的成员进行精确控制，并允许 Machine API Operator 在不需要 etcd 集群的特定操作了解的情况下安全地创建和删除 control plane 机器。

6.2.5.1. 使用仲裁保护处理顺序删除 control plane

当在使用 control plane 机器集的集群中替换 control plane 机器时，集群会临时有四个 control plane 机器。当第四个 control plane 节点加入集群时，etcd Operator 会在替换节点上启动新的 etcd 成员。当 etcd Operator 观察到旧的 control plane 机器已被标记为删除时，它会停止旧节点上的 etcd 成员，并提升替换 etcd 成员以加入集群的仲裁。

control plane 机器 **Deleting** 阶段按以下顺序进行：

1. control plane 机器会停止以进行删除。
2. control plane 机器进入 **Deleting** 阶段。
3. 为了满足 **preDrain** 生命周期 hook，etcd Operator 会执行以下操作：
    a. etcd Operator 等待第四个 control plane 机器作为 etcd 成员添加到集群中。这个新 etcd 成员的状态为 **Running** 而不是 **ready**，直到它从 etcd leader 接收到了完整的数据库更新。
    b. 当新 etcd 成员收到完整数据库更新时，etcd Operator 会将新的 etcd 成员提升到投票成员，并从集群中移除旧的 etcd 成员。

    完成此转换后，旧的 etcd pod 及其数据是安全的，因此会删除 **preDrain** 生命周期 hook。
4. control plane 机器状态条件 **Drainable** 设置为 **True**。

5. 机器控制器尝试排空由 control plane 机器支持的节点。
   - 如果排空失败，**Drained** 被设置为 **False**，机器控制器会尝试再次排空该节点。
   - 如果排空成功，**Drained** 被设置为 **True**。
6. control plane 机器状态条件 **Drained** 设置为 **True**。
7. 如果没有其他 Operator 添加了 **preTerminate** 生命周期 hook，control plane 机器状态条件 **Terminable** 被设置为 **True**。
8. 机器控制器从基础架构供应商中删除实例。
9. 机器控制器会删除 **Node** 对象。

YAML 片断演示 etcd 仲裁保护 preDrain 生命周期 hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain:
    - name: EtcdQuorumOperator 1
      owner: clusteroperator/etcd 2
  ...
```

1 **preDrain** 生命周期 hook 的名称。

2 管理 **preDrain** 生命周期 hook 的 hook 实施控制器。

6.3. 其他资源

- 机器阶段和生命周期
- 替换不健康的 etcd 成员
- 使用 control plane 机器集管理 control plane 机器

第 7 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

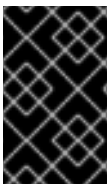
您只能在 Machine API Operator 操作的集群中配置集群自动扩展。

7.1. 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 worker 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。

集群自动扩展会计算集群中所有节点上的内存、CPU 和 GPU，即使它不管理 control plane 节点。这些值不是单计算机导向型。它们是整个集群中所有资源的聚合。例如，如果您设置最大内存资源限制，集群自动扩展在计算当前内存用量时包括集群中的所有节点。然后，该计算用于确定集群自动扩展是否具有添加更多 worker 资源的容量。



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。

每隔 10 秒，集群自动扩展会检查集群中不需要哪些节点，并移除它们。如果满足以下条件，集群自动扩展会考虑要删除的节点：

- 节点使用率低于集群的节点 *利用率级别* 阈值。节点使用率级别是请求的资源的总和，由分配给节点的资源划分。如果您没有在 **ClusterAutoscaler** 自定义资源中指定值，集群自动扩展会使用默认值 **0.5**，它对应于 50% 的利用率。
- 集群自动扩展可以将节点上运行的所有 pod 移到其他节点。Kubernetes 调度程序负责在节点上调度 pod。
- 集群自动扩展没有缩减禁用注解。

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。

例如，您可以将最大 CPU 限值设置为 64 个内核，并将集群自动扩展配置为每个创建具有 8 个内核的机器。如果您的集群从 30 个内核开始，集群自动扩展可最多添加具有 32 个内核的 4 个节点，共 62 个。

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。

集群自动扩展支持在其上有机器 API 的平台。

7.1.1. 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。

7.1.1.1. 集群自动扩展资源定义

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
      min: 0
      max: 4
logVerbosity: 4 10
scaleDown: 11
  enabled: true 12
  delayAfterAdd: 10m 13
  delayAfterDelete: 5m 14
  delayAfterFailure: 30s 15
  unneededTime: 5m 16
  utilizationThreshold: "0.4" 17
expanders: ["Random"] 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. 指定在集群中部署的最小内核数。

4. 指定集群中要部署的最大内核数。

5. 指定集群中最小内存量（以 GiB 为单位）。

6. 指定集群中的最大内存量（以 GiB 为单位）。

7. 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. 指定在集群中部署的最小 GPU 数。

9. 指定集群中要部署的最大 GPU 数量。

10. 指定 **0** 到 **10** 之间的日志记录详细程度。为指导提供了以下日志级别阈值：

    - **1**：（默认）有关更改的基本信息。
    - **4**：用于对典型问题进行故障排除的详细程度。
    - **9**：广泛的、协议级的故障排除信息。

    如果没有指定值，则使用默认值 **1**。

11 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**。

12 指定集群自动扩展是否可以删除不必要的节点。

13 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**。

14 可选：指定在最近*删除*节点之后要等待多久才能删除节点。如果没有指定值，则使用默认值 **0s**。

15 可选：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**。

16 可选：指定不必要的节点有资格删除前的时间。如果不指定值，则使用默认值 **10m**。

17 可选：指定 *节点使用率级别*。此使用率级别下的节点可以被删除。

节点使用率是请求的资源的总和（由节点分配的资源划分），且值必须大于 **"0"**，但小于 **"1"**。如果没有指定值，集群自动扩展会使用默认值 **"0.5"**，它对应于 50% 的使用率。您必须以字符串形式表示这个值。

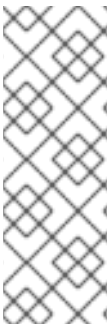
18 可选：指定您希望集群自动扩展使用的任何扩展器。以下值有效：

- **LeastWaste**: 选择机器集，可以在扩展后闲置的 CPU 最少。如果多个机器集都会产生相同的空闲 CPU，则选择会有最少未使用内存的机器集
- **Priority** ：选择具有最高用户分配的优先级的机器集。要使用此扩展器，您需要创建一个用于定义机器集的优先级的配置映射。如需更多信息，请参阅"为集群自动扩展配置优先级扩展程序"。
- **Random**: （默认）随机选择机器集。

如果没有指定值，则使用默认值 **Random**。

您可以使用 **[LeastWaste, Priority]** 格式指定多个扩展器。集群自动扩展会根据指定的顺序应用每个扩展器。

在 **[LeastWaste, Priority]** 示例中，集群自动扩展首先根据 **LeastWaste** 标准进行评估。如果多个机器集同样满足 **LeastWaste** 的标准，集群自动扩展会根据 **Priority** 标准进行评估。如果多个机器集可以同样满足所有指定的扩展器，集群自动扩展会随机选择一个使用。



注意

执行扩展操作时，集群自动扩展会保持在 **ClusterAutoscaler** 资源定义中设置的范围，如要部署的最小和最大内核数，或集群中的内存量。但是，集群自动扩展无法将集群中的当前值修正为在这些范围内。

最小和最大 CPU、内存和 GPU 值是通过计算集群中所有节点上的这些资源来确定，即使集群自动扩展无法管理该节点。例如，control plane 节点在集群的总内存中考虑，即使集群自动扩展不管理 control plane 节点。

7.1.1.2. 为集群自动扩展配置优先级扩展器

当集群自动扩展使用优先级扩展器时，它会使用具有最高用户分配优先级的机器集进行扩展。要使用此扩展器，您需要创建一个用于定义机器集的优先级的配置映射。

对于每个指定的优先级级别，您必须创建正则表达式来匹配在优先选择机器集时要使用的机器集。正则表达式必须与您希望集群自动扩展要考虑的任何计算机器集的名称匹配。

先决条件

- 您已部署了使用 Machine API 的 OpenShift Container Platform 集群。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 运行以下命令列出集群中的计算机器集：

```
$ oc get machinesets.machine.openshift.io
```

输**出示例**

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
archive-agl030519-vplxk-worker-us-east-1c   1         1         1       1           25m
fast-01-agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
fast-02-agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
fast-03-agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
fast-04-agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
prod-01-agl030519-vplxk-worker-us-east-1a   1         1         1       1           33m
prod-02-agl030519-vplxk-worker-us-east-1c   1         1         1       1           33m
```

2. 使用正则表达式，构建一个匹配模式，它被用来匹配您要设置优先级的计算机器集的名称。例如，使用正则表达式模式 ***fast*** 匹配任何在其名称中包含字符串 **fast** 的计算机器集。

3. 创建一个 **cluster-autoscaler-priority-expander.yml** YAML 文件，该文件定义了类似如下的配置映射：

优**先**级扩**展器配置映射示例**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander 1
  namespace: openshift-machine-api 2
data:
  priorities: |- 3
    10:
      - *fast*
      - *archive*
    40:
      - *prod*
```

1. 您必须将配置映射命名为 **cluster-autoscaler-priority-expander**。
2. 您必须在集群自动扩展 pod 所在的相同命名空间中（**openshift-machine-api** 命名空间）创建配置映射。
3. 定义机器集的优先级。

   priorities 值必须是正整数。集群自动扩展会首先使用值更高的优先级。

对于每个优先级级别，指定与您要使用的机器集对应的正则表达式。

4. 运行以下命令来创建配置映射：

```
$ oc create configmap cluster-autoscaler-priority-expander \
  --from-file=<location_of_config_map_file>/cluster-autoscaler-priority-expander.yml
```

验证

- 运行以下命令来查看配置映射：

```
$ oc get configmaps cluster-autoscaler-priority-expander -o yaml
```

后续步骤

- 要使用优先级扩展器，确保 **ClusterAutoscaler** 资源定义被配置为使用 **expanders: ["Priority"]** 参数。

7.1.2. 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. 为包含自定义资源定义的 **ClusterAutoscaler** 资源创建一个 YAML 文件。
2. 运行以下命令在集群中创建自定义资源：

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

1 **<filename>** 是自定义资源文件的名称。

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。

7.2. 关于机器自动扩展

机器自动扩展会调整您在 OpenShift Container Platform 集群中部署的计算机器集中的机器数量。您可以扩展默认 **worker** 计算机器集，以及您创建的任何其他计算机器集。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计算机器设置。

重要

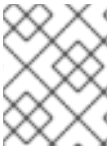
您必须部署机器自动扩展才能使用集群自动扩展功能来扩展机器。集群自动扩展使用机器自动扩展集上的注解来确定可扩展的资源。如果您在没有定义机器自动扩展的情况下定义集群自动扩展，集群自动扩展永远不会扩展集群。

7.2.1. 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计算机器集。

重要

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



注意

您必须为每个计算机器集配置单独的资源。请记住，每个地区中的计算机器集都不同，因此请考虑是否要在多个地区中启用机器扩展。扩展的计算机器必须至少有一个机器。

7.2.1.1. 机器自动扩展资源定义

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计算机器，请指定或包含要扩展的计算机器设置的名称。计算机器集名称采用以下形式： **<clusterid>-<machineset>-<region>**。

(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。如果在 AWS、GCP、Azure、RHOSP 或 vSphere 中运行，则此值可设为 **0**。对于其他供应商，请不要将此值设置为 **0**。

对于用于特殊工作负载的高价或有限使用硬件，或者扩展具有额外大型机器的计算机器集，您可以将此值设置为 **0** 来节约成本。如果机器没有使用，集群自动扩展会将计算机器设置缩减为零。

重要

对于安装程序置备的基础架构，请不要将 OpenShift Container Platform 安装过程中创建的三台计算机器集的 **spec.minReplicas** 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类型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，提供描述要扩展的现有计算机器设置的值。

(5) **kind** 参数值始终为 **MachineSet**。

(6) **name** 值必须与现有计算机器集的名称匹配，如 **metadata.name** 参数值中所示。

7.2.2. 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，其中包含自定义资源定义。
2. 运行以下命令在集群中创建自定义资源：

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

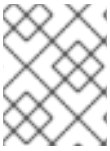
1 **<filename>** 是自定义资源文件的名称。

7.3. 禁用自动扩展

您可以在集群中禁用单独的机器自动扩展，或者完全禁用集群中的自动扩展。

7.3.1. 禁用机器自动扩展

要禁用机器自动扩展，您可以删除对应的 **MachineAutoscaler** 自定义资源(CR)。



注意

禁用机器自动扩展不会禁用集群自动扩展。要禁用集群自动扩展，请按照"禁用集群自动扩展"中的说明进行操作。

流程

1. 运行以下命令列出集群的 **MachineAutoscaler** CR：

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

输**出示例**

```
NAME                REF KIND     REF NAME            MIN   MAX   AGE
compute-us-east-1a  MachineSet   compute-us-east-1a  1     12    39m
compute-us-west-1a  MachineSet   compute-us-west-1a  2     4     37m
```

2. 可选：通过运行以下命令，创建 **MachineAutoscaler** CR 的 YAML 文件备份：

```
$ oc get MachineAutoscaler/<machine_autoscaler_name> \ 1
  -n openshift-machine-api \
  -o yaml> <machine_autoscaler_name_backup>.yaml 2
```

1 **<machine_autoscaler_name>** 是您要删除的 CR 的名称。

2 **<machine_autoscaler_name_backup>** 是 CR 备份的名称。

3. 运行以下命令来删除 **MachineAutoscaler** CR：

```
$ oc delete MachineAutoscaler/<machine_autoscaler_name> -n openshift-machine-api
```

输出示例

```
machineautoscaler.autoscaling.openshift.io "compute-us-east-1a" deleted
```

验证

- 要验证机器自动扩展是否已禁用，请运行以下命令：

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

禁用的机器自动扩展不会出现在机器自动扩展列表中。

后续步骤

- 如果您需要重新启用机器自动扩展，请使用 **<machine_autoscaler_name_backup>.yaml** 备份文件，并按照"Deploying a machine autoscaler"中的说明进行操作。

其他资源

- 禁用集群自动扩展
- 部署机器自动扩展

7.3.2. 禁用集群自动扩展

要禁用集群自动扩展，您可以删除对应的 **ClusterAutoscaler** 资源。

注意

禁用集群自动扩展会禁用集群中的自动扩展，即使集群有现有的机器自动扩展。

流程

1. 运行以下命令列出集群的 **ClusterAutoscaler** 资源：

```
$ oc get ClusterAutoscaler
```

输出示例

```
NAME      AGE
default   42m
```

2. 可选：通过运行以下命令，创建 **ClusterAutoscaler** CR 的 YAML 文件备份：

```
$ oc get ClusterAutoscaler/default \ 1
  -o yaml> <cluster_autoscaler_backup_name>.yaml 2
```

1. **默认为 ClusterAutoscaler** CR 的名称。

2 **<cluster_autoscaler_backup_name>** 是 CR 备份的名称。

3. 运行以下命令来删除 **ClusterAutoscaler** CR：

```
$ oc delete ClusterAutoscaler/default
```

输**出示例**

```
clusterautoscaler.autoscaling.openshift.io "default" deleted
```

验证

- 要验证集群自动扩展是否已禁用，请运行以下命令：

```
$ oc get ClusterAutoscaler
```

预**期**输**出**

```
No resources found
```

后续步骤

- 通过删除 **ClusterAutoscaler** CR 禁用集群自动扩展可防止集群自动扩展，但不会删除集群中的任何现有机器自动扩展。要清理不需要的机器自动扩展，请参阅"禁用机器自动扩展"。
- 如果您需要重新启用集群自动扩展，请使用 **<cluster_autoscaler_name_backup>.yaml** 备份文件，并按照"Deploying a cluster autoscaler"中的说明进行操作。

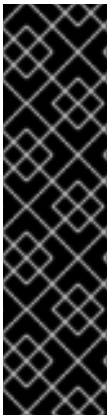
其他资源

- 禁用机器自动扩展
- 部署集群自动扩展

7.4. 其他资源

- 在 OpenShift Container Platform 中的 pod 调度决策中包含 pod 优先级

第 8 章 创建基础架构机器集



重要

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

您可以使用基础架构机器集来创建仅托管基础架构组件的机器，如默认路由器、集成的容器镜像 registry 以及集群指标和监控的组件。这些基础架构机器不会被计算为运行环境所需的订阅总数。

在生产部署中，建议您至少部署三个机器集来容纳基础架构组件。Red Hat OpenShift Service Mesh 部署 Elasticsearch，这需要三个实例安装到不同的节点上。这些节点都可以部署到不同的可用区以实现高可用性。此配置需要三个不同的机器集，每个可用区都有一个。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。

8.1. OPENSHIFT CONTAINER PLATFORM 基础架构组件

每个自我管理的 Red Hat OpenShift 订阅都包括 OpenShift Container Platform 和其他 OpenShift 相关组件的权利。这些权利包括在运行 OpenShift Container Platform control plane 和基础架构工作负载时，不需要在大小期间考虑这些权利。

要有资格成为基础架构节点并使用包含的权利，只有支持集群的组件，而不是最终用户应用程序的一部分，才能在这些实例上运行。示例包括以下组件：

- Kubernetes 和 OpenShift Container Platform control plane 服务
- 默认路由器
- 集成的容器镜像 registry
- 基于 HAProxy 的 Ingress Controller
- 集群指标集合或监控服务，包括监控用户定义的项目的组件
- 集群聚合日志
- Red Hat Quay
- Red Hat OpenShift Data Foundation
- Red Hat Advanced Cluster Management for Kubernetes
- Red Hat Advanced Cluster Security for Kubernetes
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines

- Red Hat OpenShift Service Mesh

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。

有关基础架构节点以及可在基础架构节点上运行，请参阅 OpenShift sizing and subscription guide for enterprise Kubernetes 文档中的 "Red Hat OpenShift control plane and infrastructure nodes"部分。

要创建基础架构节点，您可以使用机器集，标记节点，或使用机器配置池。

8.2. 为生产环境创建基础架构机器集

在生产部署中，建议您部署至少三个计算机器集来容纳基础架构组件。Red Hat OpenShift Service Mesh 部署 Elasticsearch，这需要三个实例安装到不同的节点上。这些节点都可以部署到不同的可用区以实现高可用性。类似的配置需要三个不同的计算机器集，每个可用区都有一个。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。

8.2.1. 为不同的云创建基础架构机器集

使用云的示例计算机器。

8.2.1.1. Alibaba Cloud 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在地区中指定的 Alibaba Cloud 区域中运行的计算机器，并创建通过 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 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: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-<infra>-<zone> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone> 10
    spec:
      metadata:
        labels:
```

```
      node-role.kubernetes.io/infra: ""
  providerSpec:
    value:
      apiVersion: machine.openshift.io/v1
      credentialsSecret:
        name: alibabacloud-credentials
      imageId: <image_id> 11
      instanceType: <instance_type> 12
      kind: AlibabaCloudMachineProviderConfig
      ramRoleName: <infrastructure_id>-role-worker 13
      regionId: <region> 14
      resourceGroup: 15
        id: <resource_group_id>
        type: ID
      securityGroups:
      - tags: 16
        - Key: Name
          Value: <infrastructure_id>-sg-<role>
        type: Tags
      systemDisk: 17
        category: cloud_essd
        size: <disk_size>
      tag: 18
      - Key: kubernetes.io/cluster/<infrastructure_id>
        Value: owned
      userDataSecret:
        name: <user_data_secret> 19
      vSwitch:
        tags: 20
        - Key: Name
          Value: <infrastructure_id>-vswitch-<zone>
        type: Tags
      vpcId: ""
      zoneId: <zone> 21
  taints: 22
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule
```

1 5 7 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（**oc**）软件包，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 指定 **<infra>** 节点标签。

4 6 10 指定基础架构 ID、**<infra>** 节点标签和区域。

11 指定要使用的镜像。使用集群的现有默认计算机器集中的镜像。

12 指定用于计算机器集的实例类型。

13 指定要用于计算机器设置的 RAM 角色的名称。使用安装程序在默认计算机器集中填充的值。

14 指定要放置机器的区域。

15 指定集群的资源组和类型。您可以使用安装程序在默认计算机器集中填充的值，或者指定不同的值。

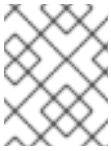
16 18 20 指定要用于计算机器设置的标签。最少，您必须包含本例中显示的标签，以及适当的集群值。您可以包括额外的标签，包括安装程序根据需要在默认的计算机器集中填充的标签。

17 指定根磁盘的类型和大小。使用安装程序在其创建的默认计算机器集中填充的 **category** 值。如果需要，以 GB 为单位指定**大小**的不同值。

19 指定 **openshift-machine-api** 命名空间中的用户数据 YAML 文件中的 secret 名称。使用安装程序在默认计算机器集中填充的值。

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

22 指定一个污点，以防止将用户工作负载调度到 infra 节点上。



注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

Alibaba Cloud 使用统计的机器集参数

安装程序为 Alibaba Cloud 集群创建的默认计算机器集包括 Alibaba Cloud 用来跟踪用量统计的 nonessential 标签值。这些标签在 **spec.template.spec.providerSpec.value** 列表的 **securityGroups**、**tag** 和 **vSwitch** 参数中填充。

在创建部署额外机器的计算机器集时，必须包含所需的 Kubernetes 标签。使用统计标签默认应用，即使它们没有在您创建的计算机器集中指定。您还可以根据需要包含其他标签。

以下 YAML 片段表示默认计算机器集中哪些标签是可选的，且需要哪些标签。

spec.template.spec.providerSpec.value.securityGroups 中的标签

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

1 2 可选：即使计算机器集中没有指定，也会应用该标签。

3 必需。

其中：

- **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。
- **<role>** 是要添加的节点标签。

spec.template.spec.providerSpec.value.tag 中的标签

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

2 3 可选：即使计算机器集中没有指定，也会应用该标签。

1 必需。

其中 **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。

spec.template.spec.providerSpec.value.vSwitch中的标签

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

1 2 3 可选：即使计算机器集中没有指定，也会应用该标签。

4 必需。

其中：

- **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。

- **<zone>** 是要放置机器的区域里的区。

8.2.1.2. AWS 上计算机器设置自定义资源的 YAML 示例

YAML 示例定义了一个在 **us-east-1a** Amazon Web Services (AWS) Local Zone 中运行的计算机器集，并创建通过 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 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
  name: <infrastructure_id>-infra-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: infra 6
        machine.openshift.io/cluster-api-machine-type: infra 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<zone> 8
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: "" 9
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 10
          apiVersion: machine.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
                volumeType: gp2
          credentialsSecret:
            name: aws-cloud-credentials
          deviceIndex: 0
          iamInstanceProfile:
            id: <infrastructure_id>-worker-profile 11
          instanceType: m6i.large
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: <zone> 12
```

```
      region: <region> 13
    securityGroups:
      - filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-worker-sg 14
    subnet:
      filters:
        - name: tag:Name
          values:
            - <infrastructure_id>-private-<zone> 15
    tags:
      - name: kubernetes.io/cluster/<infrastructure_id> 16
        value: owned
      - name: <custom_tag_name> 17
        value: <custom_tag_value> 18
    userDataSecret:
      name: worker-user-data
  taints: 19
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
```

1 3 5 11 14 16 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、**infra** 角色节点标签和区域。

6 7 9 指定 **infra** 角色节点标签。

10 为 OpenShift Container Platform 节点的 AWS 区域指定有效的 Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI)。如果要使用 AWS Marketplace 镜像，则必须从 AWS Marketplace 完成 OpenShift Container Platform 订阅来获取您所在地区的 AMI ID。

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

17 18 可选：为集群指定自定义标签数据。例如，您可以通过指定 **Email:admin-email@example.com** 的 **name:value** 对来添加管理员的电子邮件地址。

注意

也可以在 **install-config.yml** 文件中在安装过程中指定自定义标签。如果 **install-config.yml** 文件和机器集包含具有相同 **name** 数据的标签，则机器集的标签值优先于 **install-config.yml** 文件中的标签值。

12 指定区域，如 **us-east-1a**。

13 指定区域，如 **us-east-1**。

15 指定基础架构 ID 和区域。

19 指定一个污点，以防止将用户工作负载调度到 **infra** 节点上。

注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

在 AWS 上运行的机器集支持非保证的 Spot 实例。与 AWS 上的 On-Demand 实例相比，您可以使用 Spot 实例以较低价格来节约成本。通过将 **spotMarketOptions** 添加到 **MachineSet** YAML 文件来配置 Spot 实例。

8.2.1.3. Azure 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在区域(region)的 **1** Microsoft Azure 区域(zone)中运行的计算机器集，并创建通过 **node-role.kubernetes.io/infra: ""** 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 **infra** 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: infra 2
    machine.openshift.io/cluster-api-machine-type: infra
  name: <infrastructure_id>-infra-<region> 3
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region>
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: infra
        machine.openshift.io/cluster-api-machine-type: infra
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra-<region>
    spec:
      metadata:
        creationTimestamp: null
        labels:
          machine.openshift.io/cluster-api-machineset: <machineset_name>
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
```

```
          image: 4
            offer: ""
            publisher: ""
            resourceID: /resourceGroups/<infrastructure_id>-
rg/providers/Microsoft.Compute/galleries/gallery_<infrastructure_id>/images/<infrastructure_id>-
gen2/versions/latest 5
            sku: ""
            version: ""
          internalLoadBalancer: ""
          kind: AzureMachineProviderSpec
          location: <region> 6
          managedIdentity: <infrastructure_id>-identity
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: ""
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: ""
          resourceGroup: <infrastructure_id>-rg
          sshPrivateKey: ""
          sshPublicKey: ""
          tags:
            - name: <custom_tag_name> 7
              value: <custom_tag_value>
          subnet: <infrastructure_id>-<role>-subnet
          userDataSecret:
            name: worker-user-data
          vmSize: Standard_D4s_v3
          vnet: <infrastructure_id>-vnet
          zone: "1" 8
      taints: 9
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
```

1 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
```

您可以运行以下命令来获取子网：

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

您可以运行以下命令来获取 vnet：

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

2 指定 **infra** 节点标签。

3 指定基础架构 ID、**infra** 节点标签和地区。

4 指定计算机器设置的镜像详情。如果要使用 Azure Marketplace 镜像，请参阅"选择 Azure Marketplace 镜像"。

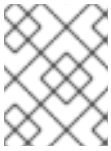
5 指定与实例类型兼容的镜像。安装程序创建的 Hyper-V 生成 V2 镜像具有 **-gen2** 后缀，而 V1 镜像则与没有后缀的名称相同。

6 指定要放置机器的区域。

7 可选：在机器集中指定自定义标签。在 **<custom_tag_name>** 字段中提供标签名称，并在 **<custom_tag_value>** 字段中提供对应的标签值。

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

9 指定一个污点，以防止将用户工作负载调度到 infra 节点上。



注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

在 Azure 上运行的机器集支持非保证的 Spot 虚拟机。与 Azure 上的标准虚拟机相比，您可以使用 Spot 虚拟机以较低价格节约成本。您可以通过将 **spotVMOptions** 添加到 **MachineSet** YAML 文件来配置 Spot VM。

其他资源

- 选择 Azure Marketplace 镜像

8.2.1.4. Azure Stack Hub 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在区域(region)的 **1** Microsoft Azure 区域(zone)中运行的计算机器集，并创建通过 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 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: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra-<region> 4
  namespace: openshift-machine-api
```

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

```
          subnet: <infrastructure_id>-<role>-subnet 18 19
          userDataSecret:
            name: worker-user-data 20
          vmSize: Standard_DS4_v2
          vnet: <infrastructure_id>-vnet 21
          zone: "1" 22
```

1 5 7 14 16 17 18 21 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
```

您可以运行以下命令来获取子网：

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

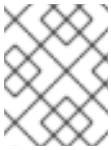
您可以运行以下命令来获取 vnet：

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

2 3 8 9 11 19 20 指定 **<infra>** 节点标签。

4 6 10 指定基础架构 ID、**<infra>** 节点标签和地区。

12 指定一个污点，以防止将用户工作负载调度到 infra 节点上。



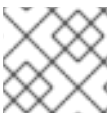
注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

15 指定要放置机器的区域。

13 指定集群的可用性集。

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



注意

在 Azure Stack Hub 上运行的机器集不支持非保证的 Spot 虚拟机。

8.2.1.5. IBM Cloud 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在地区中指定的 IBM Cloud® 区域中运行的计算机器集，并创建通过 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 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: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-<infra>-<region> 4
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<region> 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<region> 10
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: ibmcloudproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: ibmcloud-credentials
          image: <infrastructure_id>-rhcos 11
          kind: IBMCloudMachineProviderSpec
          primaryNetworkInterface:
            securityGroups:
            - <infrastructure_id>-sg-cluster-wide
            - <infrastructure_id>-sg-openshift-net
            subnet: <infrastructure_id>-subnet-compute-<zone> 12
          profile: <instance_profile> 13
          region: <region> 14
          resourceGroup: <resource_group> 15
          userDataSecret:
            name: <role>-user-data 16
          vpc: <vpc_name> 17
          zone: <zone> 18
        taints: 19
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
```

1 5 7 基于您在置备集群时设定的集群 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 16 **<infra>** 节点标签。

4 6 10 基础架构 ID、**<infra>** 节点标签和地区。

11 用于集群安装的自定义 Red Hat Enterprise Linux CoreOS(RHCOS)镜像。

12 您区域内的基础架构 ID 和区域来放置机器。确保您的地区支持您指定的区域。

13 指定 IBM Cloud® 实例配置集。

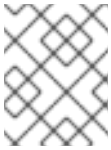
14 指定要放置机器的区域。

15 在其中放置机器资源的资源组。这是安装时指定的现有资源组，或根据基础架构 ID 命名的安装程序创建资源组。

17 VPC 名称。

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

19 污点(taint)，以防止将用户工作负载调度到 infra 节点上。



注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

8.2.1.6. GCP 上计算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) 中运行的计算机器集，并创建通过 **node-role.kubernetes.io/infra: ""** 标记的节点，其中 **infra** 是要添加的节点标签。

使用 OpenShift CLI 获取的值
在以下示例中，您可以使用 OpenShift CLI 获取集群的一些值。

基础架构 ID

<infrastructure_id> 字符串是基础架构 ID，它基于您在置备集群时设定的集群 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
```

镜像路径

<path_to_image> 字符串是用于创建磁盘的镜像的路径。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取镜像的路径：

```
$ oc -n openshift-machine-api \
  -o jsonpath='{.spec.template.spec.providerSpec.value.disks[0].image}{"\n"}' \
  get machineset/<infrastructure_id>-worker-a
```

GCP MachineSet 值示例

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

```
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-w-a
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <infra> 2
        machine.openshift.io/cluster-api-machine-type: <infra>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-w-a
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: gcpprovider.openshift.io/v1beta1
          canIPForward: false
          credentialsSecret:
            name: gcp-cloud-credentials
          deletionProtection: false
          disks:
          - autoDelete: true
            boot: true
            image: <path_to_image> 3
            labels: null
            sizeGb: 128
            type: pd-ssd
          gcpMetadata: 4
          - key: <custom_metadata_key>
            value: <custom_metadata_value>
          kind: GCPMachineProviderSpec
          machineType: n1-standard-4
          metadata:
            creationTimestamp: null
          networkInterfaces:
          - network: <infrastructure_id>-network
            subnetwork: <infrastructure_id>-worker-subnet
          projectID: <project_name> 5
          region: us-central1
          serviceAccounts:
          - email: <infrastructure_id>-w@<project_name>.iam.gserviceaccount.com
            scopes:
            - https://www.googleapis.com/auth/cloud-platform
          tags:
```

```
      - <infrastructure_id>-worker
    userDataSecret:
      name: worker-user-data
    zone: us-central1-a
  taints: 6
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule
```

1 其中 **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。

2 对于 **<infra>**，指定 **<infra>** 节点标签。

3 指定当前计算机器集中使用的镜像的路径。

要使用 GCP Marketplace 镜像，请指定要使用的功能：

- OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**
- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**

4 可选：以 **key:value** 对的形式指定自定义元数据。有关用例，请参阅 GCP 文档，以查看设置自定义元数据。

5 对于 **<project_name>**，请指定用于集群的 GCP 项目的名称。

6 指定一个污点，以防止将用户工作负载调度到 infra 节点上。

注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

在 GCP 上运行的机器集支持非保证的可抢占虚拟机实例。与 GCP 上的普通实例相比，您可以使用抢占虚拟机实例以较低价格节约成本。您可以通过将 **preemptible** 添加到 **MachineSet** YAML 文件来配置抢占 VM 实例。

8.2.1.7. Nutanix 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个 Nutanix 计算机器集，它创建标记为 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 则是要添加的节点标签。

使用 OpenShift CLI 获取的值
在以下示例中，您可以使用 OpenShift CLI (**oc**) 获取集群的一些值。

基础架构 ID

<infrastructure_id> 字符串是基础架构 ID，它基于您在置备集群时设定的集群 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
```

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra>
  name: <infrastructure_id>-<infra>-<zone> 3
  namespace: openshift-machine-api
  annotations: 4
    machine.openshift.io/memoryMb: "16384"
    machine.openshift.io/vCPU: "4"
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <infra>
        machine.openshift.io/cluster-api-machine-type: <infra>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<infra>-<zone>
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/infra: ""
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1
          bootType: "" 5
          categories: 6
          - key: <category_name>
            value: <category_value>
          cluster: 7
            type: uuid
            uuid: <cluster_uuid>
          credentialsSecret:
            name: nutanix-credentials
          image:
            name: <infrastructure_id>-rhcos 8
            type: name
          kind: NutanixMachineProviderConfig
          memorySize: 16Gi 9
          project: 10
            type: name
```

```
        name: <project_name>
      subnets:
      - type: uuid
        uuid: <subnet_uuid>
      systemDiskSize: 120Gi 11
      userDataSecret:
        name: <user_data_secret> 12
      vcpuSockets: 4 13
      vcpusPerSocket: 1 14
    taints: 15
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
```

1. 其中 **<infrastructure_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 ID。
2. 指定 **<infra>** 节点标签。
3. 指定基础架构 ID、**<infra>** 节点标签和区域。
4. 集群自动扩展的注解。
5. 指定计算机器使用的引导类型。有关引导类型的更多信息，请参阅虚拟环境中的了解 UEFI、安全引导和 TPM。有效值为 **Legacy**、**SecureBoot** 或 **UEFI**。默认值为 **Legacy**。

注意

您必须在 OpenShift Container Platform 4.16 中使用 **Legacy** 引导类型。

6. 指定一个或多个 Nutanix Prism 类别以应用到计算机器。此小节需要 **key** 和 **value** 参数代表存在于 Prism Central 中的类别的键值对。有关类别的更多信息，请参阅类别管理。
7. 指定 Nutanix Prism Element 集群配置。在本例中，集群类型是 **uuid**，因此有一个 **uuid** 小节。
8. 指定要使用的镜像。使用集群的现有默认计算机器集中的镜像。
9. 指定集群的内存量（以 Gi 为单位）。
10. 指定用于集群的 Nutanix 项目。在本例中，项目类型是 **name**，因此有一个 **name** 小节。
11. 指定系统磁盘大小（以 Gi 为单位）。
12. 指定 **openshift-machine-api** 命名空间中的用户数据 YAML 文件中的 secret 名称。安装程序在默认计算机器集中填充时使用的值。
13. 指定 vCPU 套接字数量。
14. 指定每个插槽的 vCPU 数量。
15. 指定一个污点，以防止将用户工作负载调度到 infra 节点上。

注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

8.2.1.8. RHOSP 上计算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) 上运行的计算机器集，并创建通过 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 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: <infra> 2
    machine.openshift.io/cluster-api-machine-type: <infra> 3
  name: <infrastructure_id>-infra 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 6
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
        machine.openshift.io/cluster-api-machine-role: <infra> 8
        machine.openshift.io/cluster-api-machine-type: <infra> 9
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 10
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: ""
      taints: 11
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: <optional_UUID_of_server_group> 12
          kind: OpenstackProviderSpec
          networks: 13
          - filter: {}
            subnets:
            - filter:
                name: <subnet_name>
```

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

1 5 7 14 16 17 18 19 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 20 指定 **<infra>** 节点标签。

4 6 10 指定基础架构 ID 和 **<infra>** 节点标签。

11 指定一个污点，以防止将用户工作负载调度到 infra 节点上。

注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

12 要为 MachineSet 设置服务器组策略，请输入从创建服务器组返回的值。对于大多数部署，推荐使用 **anti-affinity** 或 **soft-anti-affinity** 策略。

13 部署到多个网络需要。如果部署到多个网络，这个列表必须包含用作 **primarySubnet** 值的网络。

15 指定您要发布节点端点的 RHOSP 子网。通常，这与 **install-config.yaml** 文件中的 **machineSubnet** 值相同。

8.2.1.9. vSphere 上计算机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 VMware vSphere 上运行的计算机器集，并创建标记为 **node-role.kubernetes.io/infra: ""** 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 **<infra>** 则是要添加的节点标签。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  creationTimestamp: null
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
```

```
  name: <infrastructure_id>-infra 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 4
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: <infra> 6
        machine.openshift.io/cluster-api-machine-type: <infra> 7
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-infra 8
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/infra: "" 9
      taints: 10
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 120
          kind: VSphereMachineProviderSpec
          memoryMiB: 8192
          metadata:
            creationTimestamp: null
          network:
            devices:
            - networkName: "<vm_network_name>" 11
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <vm_template_name> 12
          userDataSecret:
            name: worker-user-data
          workspace:
            datacenter: <vcenter_datacenter_name> 13
            datastore: <vcenter_datastore_name> 14
            folder: <vcenter_vm_folder_path> 15
            resourcepool: <vsphere_resource_pool> 16
            server: <vcenter_server_ip> 17
```

1 3 5 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（**oc**）软件包，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 和 **<infra>** 节点标签。

6 7 9 指定 **<infra>** 节点标签。

10 指定一个污点，以防止将用户工作负载调度到 infra 节点上。

注意

在基础架构节点上添加 **NoSchedule** 污点后，在该节点上运行的现有 DNS pod 被标记为 **misscheduled**。您必须删除或在 **misscheduled** DNS pod 中添加容限。

11 指定要将计算机器设置为的 vSphere VM 网络。此虚拟机网络必须是集群中其他计算机器所处的位置。

12 指定要使用的 vSphere 虚拟机模板，如 **user-5ddjd-rhcos**。

13 指定要将计算机器设置为的 vCenter Datacenter。

14 指定要部署计算机器的 vCenter Datastore。

15 指定 vCenter 中 vSphere 虚拟机文件夹的路径，如 **/dc1/vm/user-inst-5ddjd**。

16 指定虚拟机的 vSphere 资源池。

17 指定 vCenter 服务器 IP 或完全限定域名。

8.2.2. 创建计算机器集

除了安装程序创建的计算机器集外，您还可以创建自己的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. 创建一个包含计算机器集自定义资源（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。
2. 可选：如果您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计算机器集：
   a. 要列出集群中的计算机器集，请运行以下命令：

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

输出示例

```
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. 要查看特定计算机器集自定义资源 (CR) 的值，请运行以下命令：

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

输出示例

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

1 集群基础架构 ID。

2 默认节点标签。

注意

对于具有用户置备的基础架构的集群，计算机器集只能创建 **worker** 和 **infra** 类型机器。

3 计算机器设置 CR 的 **<providerSpec>** 部分中的值是特定于平台的。有关 CR 中的 **<providerSpec>** 参数的更多信息，请参阅您的供应商计算机器设置 CR 配置示例。

3. 运行以下命令来创建 **MachineSet** CR：

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

验证

- 运行以下命令，查看计算机器集列表：

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

输出示例

```
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
```

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

8.2.3. 创建基础架构节点

重要

请参阅为安装程序置备的基础架构环境创建基础架构机器集，或为其 control plane 节点由机器 API 管理的任何集群创建基础架构机器集。

集群的基础架构系统（也称为 **infra 节点**）的要求已被置备。安装程序只为 control plane 和 worker 节点提供置备。Worker 节点可以通过标记来指定为基础架构节点或应用程序（也称为 **app** ）。

流程

1. 向您要充当应用程序节点的 worker 节点添加标签：

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

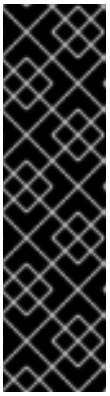
2. 向您要充当基础架构节点的 worker 节点添加标签：

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

3. 检查相关节点现在是否具有 **infra** 角色或 **app** 角色：

```
$ oc get nodes
```

4. 创建默认的集群范围节点选择器。默认节点选择器应用到在所有命名空间中创建的 pod。这会创建一个与 pod 上任何现有节点选择器交集的交集，这会额外限制 pod 的选择器。



重要

如果默认节点选择器键与 pod 标签的键冲突，则不会应用默认节点选择器。

但是，不要设置可能会导致 pod 变得不可调度的默认节点选择器。例如，当 pod 的标签被设置为不同的节点角色（如 **node-role.kubernetes.io/infra=""**）时，将默认节点选择器设置为特定的节点角色（如 **node-role.kubernetes.io/master=""**）可能会导致 pod 无法调度。因此，将默认节点选择器设置为特定节点角色时要小心。

您还可以使用项目节点选择器来避免集群范围节点选择器键冲突。

a. 编辑 **Scheduler** 对象：

```
$ oc edit scheduler cluster
```

b. 使用适当的节点选择器添加 **defaultNodeSelector** 字段：

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
spec:
  defaultNodeSelector: node-role.kubernetes.io/infra="" (1)
# ...
```

(1) 这个示例节点选择器默认在基础架构节点上部署 pod。

c. 保存文件以使改变生效。

现在，您可以将基础架构资源移到新标记的 **infra** 节点。

其他资源

- 将资源移到基础架构机器集

8.2.4. 为基础架构机器创建机器配置池

如果需要基础架构机器具有专用配置，则必须创建一个 infra 池。

重要

在创建一个默认自定义集群配置池时，如果它们引用同一文件或单元，则创建的自定义机器配置池会覆盖默认的 worker 池配置。

流程

1. 向您要分配为带有特定标签的 infra 节点的节点添加标签：

```
$ oc label node <node_name> <label>
```

```
$ oc label node ci-ln-n8mqwr2-f76d1-xscn2-worker-c-6fmtx node-role.kubernetes.io/infra=
```

2. 创建包含 worker 角色和自定义角色作为机器配置选择器的机器配置池：

```
$ cat infra.mcp.yaml
```

输出示例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: infra
spec:
  machineConfigSelector:
    matchExpressions:
      - {key: machineconfiguration.openshift.io/role, operator: In, values: [worker,infra]} (1)
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: "" (2)
```

(1) 添加 worker 角色和自定义角色。

(2) 将您添加的标签作为 **nodeSelector** 添加到节点。

注意

自定义机器配置池从 worker 池中继承机器配置。自定义池使用任何针对 worker 池的机器配置，但增加了部署仅针对自定义池的更改的功能。由于自定义池从 worker 池中继承资源，对 worker 池的任何更改也会影响自定义池。

3. 具有 YAML 文件后，您可以创建机器配置池：

```
$ oc create -f infra.mcp.yaml
```

4. 检查机器配置，以确保基础架构配置成功：

```
$ oc get machineconfig
```

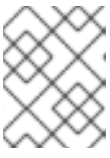
输出示例

```
NAME                                               GENERATEDBYCONTROLLER
IGNITIONVERSION   CREATED
00-master                                          365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0             31d
00-worker                                          365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0             31d
01-master-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             31d
01-master-kubelet                                  365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0             31d
01-worker-container-runtime
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             31d
01-worker-kubelet                                  365c1cfd14de5b0e3b85e0fc815b0060f36ab955
3.2.0             31d
```

```
99-master-1ae2a1e0-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             31d
99-master-ssh                                                 3.2.0             31d
99-worker-1ae64748-a115-11e9-8f14-005056899d54-registries
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             31d
99-worker-ssh                                                 3.2.0             31d
rendered-infra-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             23m
rendered-master-072d4b2da7f88162636902b074e9e28e
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0             31d
rendered-master-3e88ec72aed3886dec061df60d16d1af
02c07496ba0417b3e12b78fb32baf6293d314f79   3.2.0             31d
rendered-master-419bee7de96134963a15fdf9dd473b25
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             17d
rendered-master-53f5c91c7661708adce18739cc0f40fb
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             13d
rendered-master-a6a357ec18e5bce7f5ac426fc7c5ffcd
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             7d3h
rendered-master-dc7f874ec77fc4b969674204332da037
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0             31d
rendered-worker-1a75960c52ad18ff5dfa6674eb7e533d
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0             31d
rendered-worker-2640531be11ba43c61d72e82dc634ce6
5b6fb8349a29735e48446d435962dec4547d3090   3.2.0             31d
rendered-worker-4e48906dca84ee702959c71a53ee80e7
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             7d3h
rendered-worker-4f110718fe88e5f349987854a1147755
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             17d
rendered-worker-afc758e194d6188677eb837842d3b379
02c07496ba0417b3e12b78fb32baf6293d314f79   3.2.0             31d
rendered-worker-daa08cc1e8f5fcdeba24de60cd955cc3
365c1cfd14de5b0e3b85e0fc815b0060f36ab955   3.2.0             13d
```

您应该会看到一个新的机器配置，带有 **rendered-infra-*** 前缀。

5. 可选：要部署对自定义池的更改，请创建一个机器配置，该配置使用自定义池名称作为标签，如本例中的 **infra**。请注意，这不是必须的，在此包括仅用于指示目的。这样，您可以只应用特定于 infra 节点的任何自定义配置。



注意

创建新机器配置池后，MCO 会为该池生成一个新的呈现配置，以及该池重启的关联节点以应用新配置。

a. 创建机器配置：

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

输**出示例**

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 51-infra
```

```
    labels:
      machineconfiguration.openshift.io/role: infra 1
  spec:
    config:
      ignition:
        version: 3.2.0
      storage:
        files:
        - path: /etc/infratest
          mode: 0644
          contents:
            source: data:,infra
```

1 将您添加的标签作为 **nodeSelector** 添加到节点。

b. 将机器配置应用到 infra-labeled 节点：

```
$ oc create -f infra.mc.yaml
```

6. 确认您的新机器配置池可用：

```
$ oc get mcp
```

输**出示例**

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
infra    rendered-infra-60e35c2e99f42d976e084fa94da4d0fc    True      False      False      1
1              1                   0                    4m20s
master   rendered-master-9360fdb895d4c131c7c4bebbae099c90   True      False      False
3              3                   3                    0                    91m
worker   rendered-worker-60e35c2e99f42d976e084fa94da4d0fc   True      False      False
2              2                   2                    0                    91m
```

在本例中，worker 节点被改为一个 infra 节点。

其他资源

- 如需有关在自定义池中分组 infra 机器的更多信息，请参阅使用机器配置池进行节点配置管理。

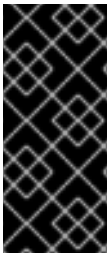
8.3. 为基础架构节点分配机器设置资源

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。

8.3.1. 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。



重要

建议您保留为 infra 节点创建的双 **infra,worker** 标签，并使用污点和容限来管理用户工作负载调度到的节点。如果从节点中删除 **worker** 标签，您必须创建一个自定义池来管理它。没有自定义池的 MCO 不能识别具有 **master** 或 **worker** 以外的标签的节点。如果不存在选择自定义标签的自定义池，维护 **worker** 标签可允许默认 worker 机器配置池管理节点。**infra** 标签与集群通信，它不计算订阅总数。

先决条件

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

流程

1. 向 infra 节点添加污点以防止在其上调度用户工作负载：

   a. 确定节点是否具有污点：

   ```
   $ oc describe nodes <node_name>
   ```

   输出示例

   ```
   oc describe node ci-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
   Name:               ci-ln-iyhx092-f76d1-nvdfm-worker-b-wln2l
   Roles:              worker
    ...
   Taints:             node-role.kubernetes.io/infra:NoSchedule
    ...
   ```

   本例显示节点具有污点。您可以在下一步中继续向 pod 添加容限。

   b. 如果您还没有配置污点以防止在其上调度用户工作负载：

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

   例如：

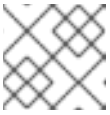
   ```
   $ oc adm taint nodes node1 node-role.kubernetes.io/infra=reserved:NoSchedule
   ```

提示

您还可以应用以下 YAML 来添加污点：

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

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。

c. 添加带有 NoExecute Effect 的污点，以及上述带有 NoSchedule Effect 的污点：

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

例如：

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

提示

您还可以应用以下 YAML 来添加污点：

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

本例在 **node1** 上放置一个键为 **node-role.kubernetes.io/infra** 的污点，污点是

本例在 **node1** 上放置一个键为 **node-role.kubernetes.io/infra** 的污点，污点是 **NoExecute**。带有 **NoExecute** 效果的节点仅调度容许该污点的 pod。该效果将从没有匹配容限的节点中删除任何现有 pod。

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)
   value: reserved (3)
 - effect: NoExecute (4)
   key: node-role.kubernetes.io/infra (5)
   operator: Exists (6)
   value: reserved (7)
```

(1) 指定添加到节点的效果。

(2) 指定添加到节点的键。

(3) 指定添加到节点的键值对污点的值。

(4) 指定添加到节点的效果。

(5) 指定添加到节点的键。

(6) 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。

(7) 指定添加到节点的键值对污点的值。

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。
- 如需有关不同污点效果的更多详情，请参阅了解污点和容限。

8.4. 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。您可以通过添加基础架构节点选择器来将其移至您创建的基础架构机器集，如下所示：

```
spec:
  nodePlacement: 1
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
```

1 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。如果您在 infrasructure 节点中添加了污点，还要添加匹配的容限。

将特定节点选择器应用到所有基础架构组件会导致 OpenShift Container Platform 使用 该标签将这些工作负载调度到具有该标签的节点。

8.4.1. 移动路由器

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

先决条件

- 在 OpenShift Container Platform 集群中配置额外的计算机器集。

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
```

```
spec:
  nodePlacement:
    nodeSelector: 1
      matchLabels:
        node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
```

1 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。如果您在 infrasructure 节点中添加了污点，还要添加匹配的容限。

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
```

   输出示例

```
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
```

   1 指定从 Pod 列表获得的 **<node_name>**。

输出示例

```
NAME                          STATUS  ROLES         AGE   VERSION
ip-10-0-217-226.ec2.internal  Ready   infra,worker  17h   v1.29.4
```

由于角色列表包含 **infra**，因此 Pod 在正确的节点上运行。

8.4.2. 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 集群中配置额外的计算机器集。

流程

1. 查看 **config/instance** 对象：

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

输出示例

```
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 configs.imageregistry.operator.openshift.io/cluster
```

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - podAffinityTerm:
          namespaces:
          - openshift-image-registry
          topologyKey: kubernetes.io/hostname
        weight: 100
  logLevel: Normal
  managementState: Managed
  nodeSelector: (1)
    node-role.kubernetes.io/infra: ""
  tolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
    value: reserved
  - effect: NoExecute
    key: node-role.kubernetes.io/infra
    value: reserved
```

(1) 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。如果您在 infrasructure 节点中添加了污点，还要添加匹配的容限。

3. 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。

8.4.3. 移动监控解决方案

监控堆栈包含多个组件，包括 Prometheus、Thanos Querier 和 Alertmanager。Cluster Monitoring Operator 管理此堆栈。要将监控堆栈重新部署到基础架构节点，您可以创建并应用自定义配置映射。

流程

1. 编辑 **cluster-monitoring-config** 配置映射，并更改 **nodeSelector** 以使用 **infra** 标签：

   ```
   $ oc edit configmap cluster-monitoring-config -n openshift-monitoring
   ```

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: cluster-monitoring-config
     namespace: openshift-monitoring
   ```

```
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector: (1)
        node-role.kubernetes.io/infra: ""
      tolerations:
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoSchedule
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoExecute
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoSchedule
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoExecute
    prometheusOperator:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoSchedule
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoExecute
    metricsServer:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoSchedule
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoExecute
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoSchedule
      - key: node-role.kubernetes.io/infra
        value: reserved
        effect: NoExecute
    telemeterClient:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
```

```
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
  openshiftStateMetrics:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
  thanosQuerier:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
  monitoringPlugin:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
```

1 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。如果您在 infrasructure 节点中添加了污点，还要添加匹配的容限。

2. 观察监控 pod 移至新机器：

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

3. 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。

8.4.4. 移动 Vertical Pod Autoscaler Operator 组件

Vertical Pod Autoscaler Operator (VPA)由三个组件组成：推荐器（recommender）、更新器

（updater）和准入控制器（admission controller）。Operator 和每个组件在 control plane 节点上的 VPA 命名空间中都有自己的 pod。您可以通过在 VPA 订阅和 **VerticalPodAutoscalerController** CR 中添加节点选择器，将 VPA Operator 和组件 pod 移到基础架构节点。

以下示例显示了 VPA pod 到 control plane 节点的默认部署。

输出示例

```
NAME                                        READY   STATUS    RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
vertical-pod-autoscaler-operator-6c75fcc9cd-5pb6z   1/1     Running   0          7m59s   10.128.2.24
c416-tfsbj-master-1   <none>           <none>
vpa-admission-plugin-default-6cb78d6f8b-rpcrj       1/1     Running   0          5m37s   10.129.2.22
c416-tfsbj-master-1   <none>           <none>
vpa-recommender-default-66846bd94c-dsmpp            1/1     Running   0          5m37s   10.129.2.20
c416-tfsbj-master-0   <none>           <none>
vpa-updater-default-db8b58df-2nkvf                  1/1     Running   0          5m37s   10.129.2.21   c416-
tfsbj-master-1   <none>           <none>
```

流程

1. 通过将节点选择器添加到 VPA Operator 的 **Subscription** 自定义资源 (CR)中来移动 VPA Operator pod：

   a. 编辑 CR：

   ```
   $ oc edit Subscription vertical-pod-autoscaler -n openshift-vertical-pod-autoscaler
   ```

   b. 添加节点选择器以匹配 infra 节点上的节点角色标签：

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     labels:
       operators.coreos.com/vertical-pod-autoscaler.openshift-vertical-pod-autoscaler: ""
     name: vertical-pod-autoscaler
   # ...
   spec:
     config:
       nodeSelector:
         node-role.kubernetes.io/infra: "" (1)
   ```

   (1) 指定 infra 节点的节点角色。

注意

如果 infra 节点使用污点，则需要为 **Subscription** CR 添加容限。

例如：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  labels:
    operators.coreos.com/vertical-pod-autoscaler.openshift-vertical-pod-
autoscaler: ""
  name: vertical-pod-autoscaler
# ...
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations: 1
    - key: "node-role.kubernetes.io/infra"
      operator: "Exists"
      effect: "NoSchedule"
```

1 为 infra 节点上的污点指定容限。

2. 通过将节点选择器添加到 **VerticalPodAutoscaler** 自定义资源 (CR) 来移动每个 VPA 组件：

   a. 编辑 CR：

   ```
   $ oc edit VerticalPodAutoscalerController default -n openshift-vertical-pod-autoscaler
   ```

   b. 添加节点选择器以匹配 infra 节点上的节点角色标签：

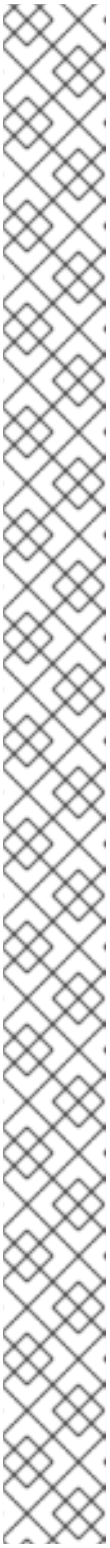
   ```
   apiVersion: autoscaling.openshift.io/v1
   kind: VerticalPodAutoscalerController
   metadata:
    name: default
     namespace: openshift-vertical-pod-autoscaler
   # ...
   spec:
     deploymentOverrides:
       admission:
         container:
           resources: {}
         nodeSelector:
           node-role.kubernetes.io/infra: "" 1
       recommender:
         container:
           resources: {}
         nodeSelector:
           node-role.kubernetes.io/infra: "" 2
       updater:
         container:
   ```

```
    resources: {}
  nodeSelector:
    node-role.kubernetes.io/infra: "" 3
```

1 可选：指定 VPA 准入 pod 的节点角色。

2 可选：指定 VPA recommender pod 的节点角色。

3 可选：指定 VPA updater pod 的节点角色。



注意

如果目标节点使用污点，则需要为 **VerticalPodAutoscalerController** CR 添加容限。

例如：

```
apiVersion: autoscaling.openshift.io/v1
kind: VerticalPodAutoscalerController
metadata:
 name: default
  namespace: openshift-vertical-pod-autoscaler
# ...
spec:
  deploymentOverrides:
    admission:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations: 1
      - key: "my-example-node-taint-key"
        operator: "Exists"
        effect: "NoSchedule"
    recommender:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations: 2
      - key: "my-example-node-taint-key"
        operator: "Exists"
        effect: "NoSchedule"
    updater:
      container:
        resources: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations: 3
      - key: "my-example-node-taint-key"
        operator: "Exists"
        effect: "NoSchedule"
```

1. 为 infra 节点上的污点指定准入控制器 pod 的容限。
2. 为 infra 节点上的污点指定 recommender pod 的容限。
3. 为 infra 节点上的污点指定 updater pod 的容限。

验证

- 您可以使用以下命令验证 pod 是否已移动：

```
$ oc get pods -n openshift-vertical-pod-autoscaler -o wide
```

pod 不再部署到 control plane 节点。

输出示例

```
NAME                                        READY   STATUS    RESTARTS   AGE     IP
NODE                          NOMINATED NODE   READINESS GATES
vertical-pod-autoscaler-operator-6c75fcc9cd-5pb6z   1/1     Running   0          7m59s
10.128.2.24   c416-tfsbj-infra-eastus3-2bndt   <none>           <none>
vpa-admission-plugin-default-6cb78d6f8b-rpcrj       1/1     Running   0          5m37s
10.129.2.22   c416-tfsbj-infra-eastus1-lrgj8   <none>           <none>
vpa-recommender-default-66846bd94c-dsmpp            1/1     Running   0          5m37s
10.129.2.20   c416-tfsbj-infra-eastus1-lrgj8   <none>           <none>
vpa-updater-default-db8b58df-2nkvf                  1/1     Running   0          5m37s   10.129.2.21
c416-tfsbj-infra-eastus1-lrgj8   <none>           <none>
```

其他资源

- 将监控组件移到其他节点
- 使用节点选择器移动日志记录资源
- 使用污点和容限来控制日志记录 pod 放置

第 9 章 在 OPENSHIFT CONTAINER PLATFORM 集群中添加 RHEL 计算机器

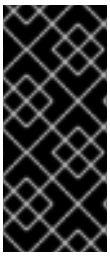
在 OpenShift Container Platform 中，您可以将 Red Hat Enterprise Linux(RHEL)计算机器添加到用户置备的基础架构集群或 **x86_64** 架构中的安装程序置备的基础架构集群中。计算机器上只能使用 RHEL 作为操作系统。

9.1. 关于在集群中添加 RHEL 计算节点

在 OpenShift Container Platform 4.16 中，如果 **x86_64** 架构中使用用户置备或安装程序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。您必须使用 Red Hat Enterprise Linux CoreOS (RHCOS) 机器作为集群中的控制平面机器。

如果您在集群中使用 RHEL 计算机器，则需要自己负责所有操作系统生命周期的管理和维护任务。您必须执行系统更新、应用补丁并完成所有其他必要的任务。

对于安装程序置备的基础架构集群，您必须手动添加 RHEL 计算机器，因为安装程序置备的基础架构集群中的自动扩展会默认添加 Red Hat Enterprise Linux CoreOS (RHCOS) 计算机器。



重要

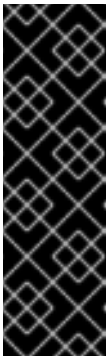
- 由于从集群中的机器上删除 OpenShift Container Platform 需要破坏操作系统，因此您必须对添加到集群中的所有 RHEL 机器使用专用的硬件。
- 添加到 OpenShift Container Platform 集群的所有 RHEL 机器上都禁用内存交换功能。您无法在这些机器上启用交换内存。

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

9.2. 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 8.6 及之后的版本。



重要

不支持将 RHEL 7 计算机器添加到 OpenShift Container Platform 集群。

如果您有在以前的 OpenShift Container Platform 版本中支持的 RHEL 7 计算机器，则无法将其升级到 RHEL 8。您必须部署新的 RHEL 8 主机，并且应该删除旧的 RHEL 7 主机。如需更多信息，请参阅"删除节点"部分。

有关 OpenShift Container Platform 中已弃用或删除的主要功能的最新列表，请参阅 OpenShift Container Platform 发行注记中*已弃用和删除的功能*部分。

- 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 8 文档中的启用 FIPS 模式安装 RHEL 8 系统 。

重要

要为集群启用 FIPS 模式，您必须从配置为以 FIPS 模式操作的 Red Hat Enterprise Linux (RHEL) 计算机运行安装程序。有关在 RHEL 中配置 FIPS 模式的更多信息，请参阅在 FIPS 模式中安装该系统。

当以 FIPS 模式运行 Red Hat Enterprise Linux (RHEL) 或 Red Hat Enterprise Linux CoreOS (RHCOS)时，OpenShift Container Platform 核心组件使用 RHEL 加密库，在 x86_64、ppc64le 和 s390x 架构上提交到 NIST FIPS 140-2/140-3 Validation。

- 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，用于包含其临时目录的文件系统。临时系统目录根据 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 DNS 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。

其他资源

- 删除节点

9.2.1. 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。

9.3. 为云准备镜像

需要 Amazon Machine Images（AMI），因为 AWS 无法直接使用各种镜像格式。您可以使用红帽提供的 AMI，也可以手动导入您自己的镜像。AMI 必须在置备 EC2 实例前就存在。您需要一个有效的 AMI ID，以便选择计算机器所需的正确 RHEL 版本。

9.3.1. 列出 AWS 中最新可用 RHEL 镜像

AMI ID 与 AWS 的原生引导镜像对应。因为 AMI 在置备 EC2 实例前必须存在，所以您需要在配置前知道 AMI ID。AWS 命令行界面（CLI）用于列出可用的 Red Hat Enterprise Linux（RHEL）镜像 ID。

先决条件

- 已安装 AWS CLI。

流程

- 使用这个命令列出 RHEL 8.4 Amazon Machine Images (AMI)：

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

1. **--owners** 命令选项显示基于帐户 ID **309956199498** 的红帽镜像。

重要

需要这个帐户 ID 来显示红帽提供的镜像的 AMI ID。

2. **--query** 命令选项设置如何根据参数 **'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]'** 对镜像进行排序。在本例中，镜像根据创建日期进行排序，表会显示创建日期、镜像名称和 AMI ID。
3. **--filter** 命令选项设定显示哪个 RHEL 版本。在本例中，由于过滤器是由 **"Name=name,Values=RHEL-8.4*"** 设置的，因此会显示 RHEL 8.4 AMI。
4. **--region** 命令选项设定存储 AMI 的区域。
5. **--output** 命令选项设定结果的显示方式。

注意

为 AWS 创建 RHEL 计算机器时，请确保 AMI 是 RHEL 8.4 或 8.5。

输**出示例**

```
----------------------------------------------------------------------------------------------------
|                                          DescribeImages                                          |
+--------------------------+-----------------------------------------------------+-----------------+
| 2021-03-18T14:23:11.000Z | RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2 | ami-
```

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

其他资源

- 您还可以手动将 RHEL 镜像导入到 AWS。

9.4. 准备机器以运行 PLAYBOOK

在将使用 Red Hat Enterprise Linux(RHEL)作为操作系统的计算机器添加到 OpenShift Container Platform 4.16 集群之前，必须准备一个 RHEL 8 机器，以运行向集群添加新节点的 Ansible 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. 确保用于安装集群的 **kubeconfig** 文件和用于安装集群的安装程序位于 RHEL 8 机器中。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 4.16 所需的仓库：

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.16-for-rhel-8-x86_64-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。

9.5. 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 \*
```

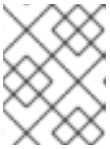
请注意，有大量可用存储库时可能需要花费几分钟

6. 仅启用 OpenShift Container Platform 4.16 需要的仓库：

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.16-for-rhel-8-x86_64-rpms" \
    --enable="fast-datapath-for-rhel-8-x86_64-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。

9.6. 将角色权限附加到 AWS 中的 RHEL 实例

在浏览器中使用 Amazon IAM 控制台，您可以选择所需的角色并将其分配到 worker 节点。

流程

1. 从 AWS IAM 控制台创建所需的 IAM 角色。
2. 将 IAM 角色附加到所需的 worker 节点。

其他资源

- 请参阅 IAM 角色所需的 AWS 权限。

9.7. 将 RHEL WORKER 节点标记为拥有或共享

集群使用 **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** 标签的值来决定与 AWS 集群相关的资源的生命周期。

- 如果在销毁集群时该资源应该被销毁，则应该添加 **owned** 标签值。
- 如果在集群销毁后资源仍然存在，则应添加 **shared** 标签值。此标记表示集群使用了此资源，但对该资源有单独的拥有者。

流程

- 使用 RHEL 计算机器时，RHEL worker 实例必须标记为 **kubernetes.io/cluster/<clusterid>=owned** 或 **kubernetes.io/cluster/<cluster-id>=shared**。

注意

不要使用 **kubernetes.io/cluster/<name>,Value=<clusterid>** 标签标记所有现有的安全组，否则 Elastic Load Balancing（ELB）将无法创建负载均衡器。

9.8. 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 4.16 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-rhel8-0.example.com
mycluster-rhel8-1.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机器的正确公共或私有名称。

2. 进入到 Ansible playbook 目录：

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

3. 运行 playbook：

```
$ 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。

9.9. 批准机器的证书签名请求

当您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。必须首先批准客户端请求，然后批准服务器请求。

先决条件

- 您已将机器添加到集群中。

流程

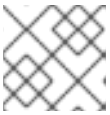
1. 确认集群可以识别这些机器：

```
$ oc get nodes
```

输**出示例**

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

输出中列出了您创建的所有机器。



注意

在有些 CSR 被批准前，前面的输出可能不包括计算节点（也称为 worker 节点）。

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

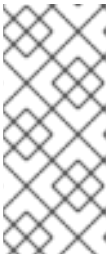
```
$ oc get csr
```

输**出示例**

```
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，在您添加的机器的所有待处理 CSR 都处于 **Pending 状态** 后，请批准集群机器的 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它们，证书将会轮转，每个节点会存在多个证书。您必须批准所有这些证书。批准客户端 CSR 后，Kubelet 为服务证书创建一个二级 CSR，这需要手动批准。然后，如果 Kubelet 请求具有相同参数的新证书，则后续提供证书续订请求由 **machine-approver** 自动批准。

注意

对于在未启用机器 API 的平台上运行的集群，如裸机和其他用户置备的基础架构，您必须实施一种方法来自动批准 kubelet 提供证书请求(CSR)。如果没有批准请求，则 **oc exec**、ocrsh 和 **oc logs** 命令将无法成功，因为 API 服务器连接到 kubelet 时需要服务证书。与 Kubelet 端点联系的任何操作都需要此证书批准。该方法必须监视新的 CSR，确认 CSR 由 system: **node** 或 **system:admin** 组中的 **node-bootstrapper** 服务帐户提交，并确认节点的身份。

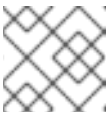
- 要单独批准，请对每个有效的 CSR 运行以下命令：

```
$ 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 --no-run-if-empty oc adm certificate approve
```



注意

在有些 CSR 被批准前，一些 Operator 可能无法使用。

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：

- 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，机器将 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  v1.29.4
master-1 Ready    master  73m  v1.29.4
master-2 Ready    master  74m  v1.29.4
worker-0 Ready    worker  11m  v1.29.4
worker-1 Ready    worker  11m  v1.29.4
```

注意

批准服务器 CSR 后可能需要几分钟时间让机器过渡到 **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。

9.10. 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。

9.10.1. 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-emptydir-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。

第 10 章 在 OPENSHIFT CONTAINER PLATFORM 集群中添加更多 RHEL 计算机器

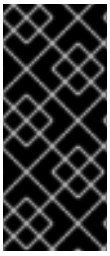
如果 OpenShift Container Platform 集群中已经包含 Red Hat Enterprise Linux (RHEL) 计算机器（也称为 worker），您可以向其中添加更多 RHEL 计算机器。

10.1. 关于在集群中添加 RHEL 计算节点

在 OpenShift Container Platform 4.16 中，如果 **x86_64** 架构中使用用户置备或安装程序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。您必须使用 Red Hat Enterprise Linux CoreOS (RHCOS) 机器作为集群中的控制平面机器。

如果您在集群中使用 RHEL 计算机器，则需要自己负责所有操作系统生命周期的管理和维护任务。您必须执行系统更新、应用补丁并完成所有其他必要的任务。

对于安装程序置备的基础架构集群，您必须手动添加 RHEL 计算机器，因为安装程序置备的基础架构集群中的自动扩展会默认添加 Red Hat Enterprise Linux CoreOS (RHCOS) 计算机器。



重要

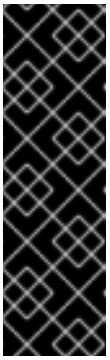
- 由于从集群中的机器上删除 OpenShift Container Platform 需要破坏操作系统，因此您必须对添加到集群中的所有 RHEL 机器使用专用的硬件。
- 添加到 OpenShift Container Platform 集群的所有 RHEL 机器上都禁用内存交换功能。您无法在这些机器上启用交换内存。

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

10.2. 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 8.6 及之后的版本。



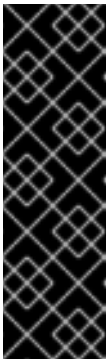
重要

不支持将 RHEL 7 计算机器添加到 OpenShift Container Platform 集群。

如果您有在以前的 OpenShift Container Platform 版本中支持的 RHEL 7 计算机器，则无法将其升级到 RHEL 8。您必须部署新的 RHEL 8 主机，并且应该删除旧的 RHEL 7 主机。如需更多信息，请参阅"删除节点"部分。

有关 OpenShift Container Platform 中已弃用或删除的主要功能的最新列表，请参阅 OpenShift Container Platform 发行注记中*已弃用和删除的功能*部分。

- 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 8 文档中的启用 FIPS 模式安装 RHEL 8 系统 。



重要

要为集群启用 FIPS 模式，您必须从配置为以 FIPS 模式操作的 Red Hat Enterprise Linux (RHEL) 计算机运行安装程序。有关在 RHEL 中配置 FIPS 模式的更多信息，请参阅在 FIPS 模式中安装该系统。

当以 FIPS 模式运行 Red Hat Enterprise Linux (RHEL) 或 Red Hat Enterprise Linux CoreOS (RHCOS)时，OpenShift Container Platform 核心组件使用 RHEL 加密库，在 x86_64、ppc64le 和 s390x 架构上提交到 NIST FIPS 140-2/140-3 Validation。

- 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，用于包含其临时目录的文件系统。临时系统目录根据 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 DNS 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。

其他资源

- 删除节点

10.2.1. 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。

10.3. 为云准备镜像

需要 Amazon Machine Images（AMI），因为 AWS 无法直接使用各种镜像格式。您可以使用红帽提供的 AMI，也可以手动导入您自己的镜像。AMI 必须在置备 EC2 实例前就存在。您必须列出 AMI ID，以便选择计算机器所需的正确 RHEL 版本。

10.3.1. 列出 AWS 中最新可用 RHEL 镜像

AMI ID 与 AWS 的原生引导镜像对应。因为 AMI 在置备 EC2 实例前必须存在，所以您需要在配置前知道 AMI ID。AWS 命令行界面（CLI）用于列出可用的 Red Hat Enterprise Linux（RHEL）镜像 ID。

先决条件

- 已安装 AWS CLI。

流程

- 使用这个命令列出 RHEL 8.4 Amazon Machine Images (AMI)：

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

1. **--owners** 命令选项显示基于帐户 ID **309956199498** 的红帽镜像。

重要

需要这个帐户 ID 来显示红帽提供的镜像的 AMI ID。

2. **--query** 命令选项设置如何根据参数 **'sort_by(Images, &CreationDate)[*].[CreationDate,Name,ImageId]'** 对镜像进行排序。在本例中，镜像根据创建日期进行排序，表会显示创建日期、镜像名称和 AMI ID。
3. **--filter** 命令选项设定显示哪个 RHEL 版本。在本例中，由于过滤器是由 **"Name=name,Values=RHEL-8.4*"** 设置的，因此会显示 RHEL 8.4 AMI。
4. **--region** 命令选项设定存储 AMI 的区域。
5. **--output** 命令选项设定结果的显示方式。

注意

为 AWS 创建 RHEL 计算机器时，请确保 AMI 是 RHEL 8.4 或 8.5。

输**出示例**

```
----------------------------------------------------------------------------------------------------
|                                          DescribeImages                                          |
+--------------------------+-----------------------------------------------------+-----------------+
| 2021-03-18T14:23:11.000Z | RHEL-8.4.0_HVM_BETA-20210309-x86_64-1-Hourly2-GP2 | ami-
```

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

其他资源

- 您还可以手动将 RHEL 镜像导入到 AWS。

10.4. 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 \*
```

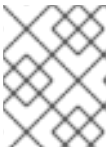
请注意，有大量可用存储库时可能需要花费几分钟

6. 仅启用 OpenShift Container Platform 4.16 需要的仓库：

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.16-for-rhel-8-x86_64-rpms" \
    --enable="fast-datapath-for-rhel-8-x86_64-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。

10.5. 将角色权限附加到 AWS 中的 RHEL 实例

在浏览器中使用 Amazon IAM 控制台，您可以选择所需的角色并将其分配到 worker 节点。

流程

1. 从 AWS IAM 控制台创建所需的 IAM 角色。
2. 将 IAM 角色附加到所需的 worker 节点。

其他资源

- 请参阅 IAM 角色所需的 AWS 权限。

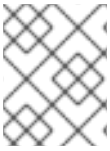
10.6. 将 RHEL WORKER 节点标记为拥有或共享

集群使用 **kubernetes.io/cluster/<clusterid>,Value=(owned|shared)** 标签的值来决定与 AWS 集群相关的资源的生命周期。

- 如果在销毁集群时该资源应该被销毁，则应该添加 **owned** 标签值。
- 如果在集群销毁后资源仍然存在，则应添加 **shared** 标签值。此标记表示集群使用了此资源，但对该资源有单独的拥有者。

流程

- 使用 RHEL 计算机器时，RHEL worker 实例必须标记为 **kubernetes.io/cluster/<clusterid>=owned** 或 **kubernetes.io/cluster/<cluster-id>=shared**。



注意

不要使用 **kubernetes.io/cluster/<name>,Value=<clusterid>** 标签标记所有现有的安全组，否则 Elastic Load Balancing（ELB）将无法创建负载均衡器。

10.7. 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 4.16 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-rhel8-0.example.com
mycluster-rhel8-1.example.com

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

   在本例中，**mycluster-rhel8-0.example.com** 和 **mycluster-rhel8-1.example.com** 机器已在集群中，您要添加 **mycluster-rhel8-2.example.com** 和 **mycluster-rhel8-3.example.com** 机器。

4. 进入到 Ansible playbook 目录：

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

5. 运行扩展 playbook：

```
$ 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。

10.8. 批准机器的证书签名请求

当您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。必须首先批准客户端请求，然后批准服务器请求。

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
```

输**出示例**

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

输出中列出了您创建的所有机器。

注意

在有些 CSR 被批准前，前面的输出可能不包括计算节点（也称为 worker 节点）。

2. 检查待处理的 CSR，并确保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
```

输**出示例**

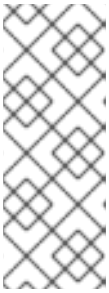
```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending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

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，在您添加的机器的所有待处理 CSR 都处于 **Pending 状态** 后，请批准集群机器的 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它们，证书将会轮转，每个节点会存在多个证书。您必须批准所有这些证书。批准客户端 CSR 后，Kubelet 为服务证书创建一个二级 CSR，这需要手动批准。然后，如果 Kubelet 请求具有相同参数的新证书，则后续提供证书续订请求由 **machine-approver** 自动批准。



注意

对于在未启用机器 API 的平台上运行的集群，如裸机和其他用户置备的基础架构，您必须实施一种方法来自动批准 kubelet 提供证书请求(CSR)。如果没有批准请求，则 **oc exec**、ocrsh 和 **oc logs** 命令将无法成功，因为 API 服务器连接到 kubelet 时需要服务证书。与 Kubelet 端点联系的任何操作都需要此证书批准。该方法必须监视新的 CSR，确认 CSR 由 system: **node** 或 **system:admin** 组中的 **node-bootstrapper** 服务帐户提交，并确认节点的身份。

- 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 --no-run-if-empty oc adm certificate approve
```

注意

在有些 CSR 被批准前，一些 Operator 可能无法使用。

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：

- 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，机器将 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  v1.29.4
master-1 Ready    master  73m  v1.29.4
master-2 Ready    master  74m  v1.29.4
worker-0 Ready    worker  11m  v1.29.4
worker-1 Ready    worker  11m  v1.29.4
```

注意

批准服务器 CSR 后可能需要几分钟时间让机器过渡到 **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。

10.9. 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。

第 11 章 手动管理用户置备基础架构

11.1. 手动使用用户置备的基础架构在集群中添加计算机器

您可以将计算机器添加到用户置备的基础架构上的集群，作为安装过程的一部分或安装后。安装后过程需要一些在安装过程中使用的相同配置文件和参数。

11.1.1. 将计算机器添加到 Amazon Web Services

要在 Amazon Web Services (AWS) 上的 OpenShift Container Platform 集群中添加更多计算机器，请参阅使用 CloudFormation 模板将计算机器添加到 AWS。

11.1.2. 将计算机器添加到 Microsoft Azure

要在 Microsoft Azure 上的 OpenShift Container Platform 集群中添加更多计算机器，请参阅在 Azure 中创建额外的 worker 机器。

11.1.3. 将计算机器添加到 Azure Stack Hub

要将更多计算机器添加到 Azure Stack Hub 上的 OpenShift Container Platform 集群中，请参阅在 Azure Stack Hub 中创建额外的 worker 机器。

11.1.4. 将计算机器添加到 Google Cloud Platform

要在 Google Cloud Platform (GCP) 上的 OpenShift Container Platform 集群中添加更多计算机器，请参阅在 GCP 中创建额外的 worker 机器。

11.1.5. 将计算机器添加到 vSphere

您可以使用计算机器集自动为 vSphere 上的 OpenShift Container Platform 集群创建额外的计算机器。

要手动在集群中添加更多计算机器，请参阅 手动将计算机器添加到 vSphere。

11.1.6. 在裸机中添加计算机器

要在裸机上的 OpenShift Container Platform 集群中添加更多计算机器，请参阅将计算机器添加到裸机。

11.2. 使用 CLOUDFORMATION 模板向 AWS 添加计算机器

您可以使用示例 CloudFormation 模板在 Amazon Web Services（AWS）上的 OpenShift Container Platform 集群中添加更多计算机器。

11.2.1. 先决条件

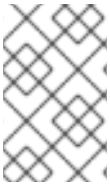
- 已使用提供的 AWS CloudFormation 模板在 AWS 上安装集群。
- 您有 JSON 文件和 CloudFormation 模板，用于在集群安装过程中创建计算机器。如果您没有这些文件，必须按照安装过程的说明重新创建这些文件。

11.2.2. 使用 CloudFormation 模板向 AWS 集群添加更多计算机器

您可以使用示例 CloudFormation 模板在 Amazon Web Services（AWS）上的 OpenShift Container Platform 集群中添加更多计算机器。

重要

CloudFormation 模板会创建一个堆栈，其代表一台计算机器。您必须为每个计算机器创建一个堆栈。



注意

如果不使用提供的 CloudFormation 模板来创建计算节点，您必须检查提供的信息并手动创建基础架构。如果您的集群没有正确初始化，您可能需要联系红帽支持并提供您的安装日志。

先决条件

- 已使用 CloudFormation 模板安装 OpenShift Container Platform 集群，并可以访问用于在集群安装过程中创建计算机器的 JSON 文件和 CloudFormation 模板。
- 已安装 AWS CLI。

流程

1. 创建另一个计算堆栈。
   a. 启动模板：

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

1 **<name>** 是 CloudFormation 堆栈的名称，如 **cluster-workers**。如果删除集群，则必须提供此堆栈的名称。

2 **<template>** 是您保存的 CloudFormation 模板 YAML 文件的相对路径和名称。

3 **<parameters>** 是 CloudFormation 参数 JSON 文件的相对路径和名称。

   b. 确认模板组件已存在：

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

2. 继续创建计算堆栈，直到为集群创建足够计算机器。

11.2.3. 批准机器的证书签名请求

当您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。必须首先批准客户端请求，然后批准服务器请求。

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
```

输出示例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.29.4
master-1  Ready     master  63m  v1.29.4
master-2  Ready     master  64m  v1.29.4
```

输出中列出了您创建的所有机器。

注意

在有些 CSR 被批准前，前面的输出可能不包括计算节点（也称为 worker 节点）。

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

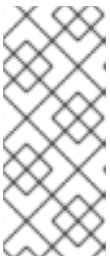
```
$ oc get csr
```

输出示例

```
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，在您添加的机器的所有待处理 CSR 都处于 **Pending 状态** 后，请批准集群机器的 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它们，证书将会轮转，每个节点会存在多个证书。您必须批准所有这些证书。批准客户端 CSR 后，Kubelet 为服务证书创建一个二级 CSR，这需要手动批准。然后，如果 Kubelet 请求具有相同参数的新证书，则后续提供证书续订请求由 **machine-approver** 自动批准。

注意

对于在未启用机器 API 的平台上运行的集群，如裸机和其他用户置备的基础架构，您必须实施一种方法来自动批准 kubelet 提供证书请求(CSR)。如果没有批准请求，则 **oc exec**、oc**rsh** 和 **oc logs** 命令将无法成功，因为 API 服务器连接到 kubelet 时需要服务证书。与 Kubelet 端点联系的任何操作都需要此证书批准。该方法必须监视新的 CSR，确认 CSR 由 system: **node** 或 **system:admin** 组中的 **node-bootstrapper** 服务帐户提交，并确认节点的身份。

- 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 --no-run-if-empty oc adm certificate approve
```

注意

在有些 CSR 被批准前，一些 Operator 可能无法使用。

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：

- 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，机器将 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  v1.29.4
master-1 Ready    master  73m  v1.29.4
master-2 Ready    master  74m  v1.29.4
worker-0 Ready    worker  11m  v1.29.4
worker-1 Ready    worker  11m  v1.29.4
```

注意

批准服务器 CSR 后可能需要几分钟时间让机器过渡到 **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。

11.3. 手动将计算机器添加到 VSPHERE

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

注意

您还可以使用计算机器集自动为集群创建额外的 VMware vSphere 计算机器。

11.3.1. 先决条件

- 您在 vSphere 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，可以按照安装过程的说明获得这些文件。

重要

如果您无法访问用于创建集群的 Red Hat Enterprise Linux CoreOS（RHCOS）镜像，您可以使用较新版本的 Red Hat Enterprise Linux CoreOS（RHCOS）镜像向 OpenShift Container Platform 集群添加更多计算机器。具体步骤，请参阅升级到 OpenShift 4.6+ 后向 UPI 集群添加新节点会失败。

11.3.2. 将更多计算机器添加到 vSphere 中的集群

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

在 OpenShift Container Platform 集群中部署 vSphere 模板后，您可以为该集群中的机器部署虚拟机 (VM)。

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. 右键点击模板的名称，再点击 **Clone → Clone to Virtual Machine**。
2. 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**。

注意

确保 vSphere 安装中的所有虚拟机名称都是唯一的。

3. 在 **Select a name and folder**选项卡中，选择您为集群创建的文件夹名称。
4. 在 **Select a compute resource** 选项卡中，选择数据中心中的主机名称。
5. 在 **Select storage** 选项卡中，为您的配置和磁盘文件选择存储。
6. 在 **Select clone options** 选项卡中，选择 **Customize this virtual machine's hardware**。
7. 在 **Customize hardware** 选项卡上，点 **Advanced Parameters**。
   - 通过在 **Attribute** 和 **Values** 字段中指定数据来添加以下配置参数名称和值。确保为您创建的每个参数选择 **Add** 按钮。
     - **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**。
8. 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。如果存在多个网络，请选择 **Add New Device** > **Network Adapter**，然后在 **New Network** 菜单项提供的字段中输入您的网络信息。
9. 完成剩余的配置步骤。点 **Finish** 按钮，您已完成克隆操作。
10. 在 **Virtual Machines** 选项卡中，右键点您的虚拟机，然后选择 **Power → Power On**。

后续步骤

- 继续为集群创建更多计算机器。

11.3.3. 批准机器的证书签名请求

当您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。必须首先批准客户端请求，然后批准服务器请求。

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
```

输出示例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

输出中列出了您创建的所有机器。

注意

在有些 CSR 被批准前，前面的输出可能不包括计算节点（也称为 worker 节点）。

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

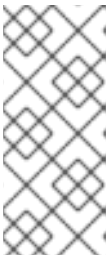
```
$ oc get csr
```

输出示例

```
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
...
```

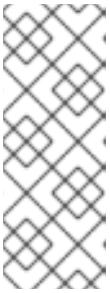
在本例中，两台机器加入集群。您可能会在列表中看到更多已批准的 CSR。

3. 如果 CSR 没有获得批准，在您添加的机器的所有待处理 CSR 都处于 **Pending 状态** 后，请批准集群机器的 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它们，证书将会轮转，每个节点会存在多个证书。您必须批准所有这些证书。批准客户端 CSR 后，Kubelet 为服务证书创建一个二级 CSR，这需要手动批准。然后，如果 Kubelet 请求具有相同参数的新证书，则后续提供证书续订请求由 **machine-approver** 自动批准。



注意

对于在未启用机器 API 的平台上运行的集群，如裸机和其他用户置备的基础架构，您必须实施一种方法来自动批准 kubelet 提供证书请求(CSR)。如果没有批准请求，则 **oc exec**、oc**rsh** 和 **oc logs** 命令将无法成功，因为 API 服务器连接到 kubelet 时需要服务证书。与 Kubelet 端点联系的任何操作都需要此证书批准。该方法必须监视新的 CSR，确认 CSR 由 system: **node** 或 **system:admin** 组中的 **node-bootstrapper** 服务帐户提交，并确认节点的身份。

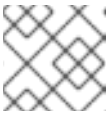
- 要单独批准，请对每个有效的 CSR 运行以下命令：

```
$ 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 --no-run-if-empty oc adm certificate approve
```



注意

在有些 CSR 被批准前，一些 Operator 可能无法使用。

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：

- 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，机器将 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  v1.29.4
master-1  Ready    master  73m  v1.29.4
```

```
master-2  Ready    master  74m  v1.29.4
worker-0  Ready    worker  11m  v1.29.4
worker-1  Ready    worker  11m  v1.29.4
```

注意

批准服务器 CSR 后可能需要几分钟时间让机器过渡到 **Ready** 状态。

其他信息

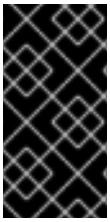
- 如需有关 CSR 的更多信息，请参阅证书签名请求。

11.4. 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。

11.4.1. 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，可以按照安装过程的说明获得这些文件。
- 如果 DHCP 服务器可用于用户置备的基础架构，您可以将额外计算机器的详情添加到您的 DHCP 服务器配置中。这包括每个机器的永久 IP 地址、DNS 服务器信息和主机名。
- 您已更新了 DNS 配置，使其包含您要添加的每个计算机器的记录名称和 IP 地址。您已验证 DNS 查找和反向 DNS 查找正确解析。



重要

如果您无法访问用于创建集群的 Red Hat Enterprise Linux CoreOS（RHCOS）镜像，您可以使用较新版本的 Red Hat Enterprise Linux CoreOS（RHCOS）镜像向 OpenShift Container Platform 集群添加更多计算机器。具体步骤，请参阅升级到 OpenShift 4.6+ 后向 UPI 集群添加新节点会失败。

11.4.2. 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。您可以使用 ISO 镜像或网络 PXE 引导来创建机器。

注意

您必须使用安装集群的相同 ISO 镜像来部署集群中的所有新节点。建议您使用相同的 Ignition 配置文件。在运行工作负载前，节点会在第一次引导时自动升级自己。您可以在升级后或之后添加节点。

11.4.2.1. 使用 ISO 镜像创建 RHCOS 机器

您可以使用 ISO 镜像为裸机集群创建更多 Red Hat Enterprise Linux CoreOS（RHCOS）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。
- 已安装 OpenShift CLI (**oc**)。

流程

1. 运行以下命令，从集群中删除 Ignition 配置文件：

```
$ oc extract -n openshift-machine-api secret/worker-user-data-managed --keys=userData --to=- > worker.ign
```

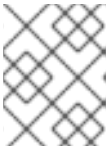
2. 将您从集群导出的 **worker.ign** Ignition 配置文件上传到 HTTP 服务器。注意这些文件的 URL。
3. 您可以验证 ignition 文件是否在 URL 上可用。以下示例获取计算节点的 Ignition 配置文件：

```
$ curl -k http://<HTTP_server>/worker.ign
```

4. 您可以运行以下命令来访问 ISO 镜像以引导新机器：

```
RHCOS_VHD_ORIGIN_URL=$(oc -n openshift-machine-config-operator get configmap/coreos-bootimages -o jsonpath='{.data.stream}' | jq -r '.architectures.<architecture>.artifacts.metal.formats.iso.disk.location')
```

5. 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。
6. 在不指定任何选项或中断实时引导序列的情况下引导 RHCOS ISO 镜像。等待安装程序在 RHCOS live 环境中引导进入 shell 提示符。



注意

您可以中断 RHCOS 安装引导过程来添加内核参数。但是，在这个 ISO 过程中，您应该使用以下步骤中所述的 **coreos-installer** 命令，而不是添加内核参数。

7. 运行 **coreos-installer** 命令并指定满足您的安装要求的选项。您至少必须指定指向节点类型的 Ignition 配置文件的 URL，以及您要安装到的设备：

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device> --ignition-hash=sha512-<digest> 1 2
```

1. 您必须使用 **sudo** 运行 **coreos-installer** 命令，因为 **core** 用户没有执行安装所需的 root 权限。
2. 当 Ignition 配置文件通过 HTTP URL 获取时，需要 **--ignition-hash** 选项来验证集群节点上 Ignition 配置文件的真实性。**<digest>** 是上一步中获取的 Ignition 配置文件 SHA512 摘要。

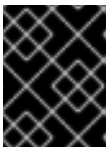
注意

如果要通过使用 TLS 的 HTTPS 服务器提供 Ignition 配置文件，您可以在运行 **coreos-installer** 前将内部证书颁发机构(CA)添加到系统信任存储中。

以下示例将引导节点安装初始化到 **/dev/sda** 设备。bootstrap 节点的 Ignition 配置文件从 IP 地址 192.168.1.2 的 HTTP Web 服务器获取：

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

8. 在机器的控制台上监控 RHCOS 安装的进度。



重要

在开始安装 OpenShift Container Platform 之前，确保每个节点中安装成功。观察安装过程可以帮助确定可能会出现 RHCOS 安装问题的原因。

9. 继续为集群创建更多计算机器。

11.4.2.2. 通过 PXE 或 iPXE 启动来创建 RHCOS 机器

您可以使用 PXE 或 iPXE 引导为裸机集群创建更多 Red Hat Enterprise Linux CoreOS（RHCOS）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。

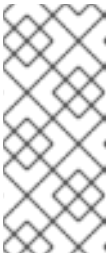
流程

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>-live-kernel-<architecture> (1)
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
```

```
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img 2
```

1. 指定上传到 HTTP 服务器的 live **kernel** 文件位置。
2. 指定上传到 HTTP 服务器的 RHCOS 文件的位置。**initrd** 参数值是 live **initramfs** 文件的位置，**coreos.inst.ignition_url** 参数值是 worker Ignition 配置文件的位置，**coreos.live.rootfs_url** 参数值是 live **rootfs** 文件的位置。**coreos.inst.ignition_url** 和 **coreos.live.rootfs_url** 参数仅支持 HTTP 和 HTTPS。



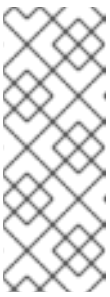
注意

此配置不会在图形控制台的机器上启用串行控制台访问。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** 以将第一个 PC 串口设置为主控制台，并将图形控制台设置为二级控制台。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 (**x86_64** + **aarch64**)：

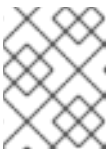
```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign 1 2
initrd --name main http://<HTTP_server>/rhcos-<version>-live-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，在 UEFI 系统中引导时需要 **initrd=main** 参数，**coreos.live.rootfs_url** 参数值是 **rootfs** 文件的位置，**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

此配置不会在使用图形控制台配置不同的控制台的机器上启用串行控制台访问，请在 **kernel** 行中添加一个或多个 **console=** 参数。例如，添加 **console=tty0 console=ttyS0** 以将第一个 PC 串口设置为主控制台，并将图形控制台设置为二级控制台。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？ 和"启用 PXE 和 ISO 安装的串行控制台"部分。



注意

要在 **aarch64** 架构中网络引导 CoreOS **内核**，您需要使用启用了 **IMAGE_GZIP** 选项的 iPXE 构建版本。请参阅 iPXE 中的**IMAGE_GZIP** 选项。

- 对于 **aarch64** 中的 PXE（使用 UEFI 和 GRUB 作为第二阶段）：

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/worker.ign 1 2
    initrd rhcos-<version>-live-initramfs.<architecture>.img 3
}
```

1. 指定上传到 HTTP/TFTP 服务器的 RHCOS 文件的位置。**kernel** 参数值是 TFTP 服务器中的 **kernel** 文件的位置。**coreos.live.rootfs_url** 参数值是 **rootfs** 文件的位置，**coreos.inst.ignition_url** 参数值则是 HTTP 服务器上的 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 TFTP 服务器的 **initramfs** 文件的位置。

2. 使用 PXE 或 iPXE 基础架构为集群创建所需的计算机器。

11.4.3. 批准机器的证书签名请求

当您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。必须首先批准客户端请求，然后批准服务器请求。

先决条件

- 您已将机器添加到集群中。

流程

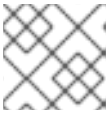
1. 确认集群可以识别这些机器：

```
$ oc get nodes
```

输**出示例**

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  63m  v1.29.4
master-1 Ready    master  63m  v1.29.4
master-2 Ready    master  64m  v1.29.4
```

输出中列出了您创建的所有机器。



注意

在有些 CSR 被批准前，前面的输出可能不包括计算节点（也称为 worker 节点）。

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

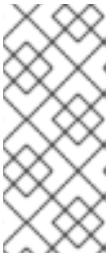
```
$ oc get csr
```

输**出示例**

```
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，在您添加的机器的所有待处理 CSR 都处于 **Pending 状态** 后，请批准集群机器的 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它们，证书将会轮转，每个节点会存在多个证书。您必须批准所有这些证书。批准客户端 CSR 后，Kubelet 为服务证书创建一个二级 CSR，这需要手动批准。然后，如果 Kubelet 请求具有相同参数的新证书，则后续提供证书续订请求由 **machine-approver** 自动批准。

注意

对于在未启用机器 API 的平台上运行的集群，如裸机和其他用户置备的基础架构，您必须实施一种方法来自动批准 kubelet 提供证书请求(CSR)。如果没有批准请求，则 **oc exec**、**oc rsh** 和 **oc logs** 命令将无法成功，因为 API 服务器连接到 kubelet 时需要服务证书。与 Kubelet 端点联系的任何操作都需要此证书批准。该方法必须监视新的 CSR，确认 CSR 由 system: **node** 或 **system:admin** 组中的 **node-bootstrapper** 服务帐户提交，并确认节点的身份。

- 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 --no-run-if-empty oc adm certificate approve
```

注意

在有些 CSR 被批准前，一些 Operator 可能无法使用。

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：

- 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，机器将处于 **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  v1.29.4
master-1  Ready     master  73m  v1.29.4
master-2  Ready     master  74m  v1.29.4
worker-0  Ready     worker  11m  v1.29.4
worker-1  Ready     worker  11m  v1.29.4
```

注意

批准服务器 CSR 后可能需要几分钟时间让机器过渡到 **Ready** 状态。

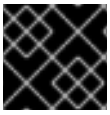
其他信息

- 如需有关 CSR 的更多信息，请参阅 证书签名请求。

第 12 章 管理 CONTROL PLANE 机器

12.1. 关于 CONTROL PLANE 机器集

使用 Control Plane 集群集，您可以自动管理 OpenShift Container Platform 集群中的 control plane 机器资源。



重要

control plane 机器集无法管理计算机器，计算机器集无法管理 control plane 机器。

control plane 机器集为 control plane 机器类似管理功能提供，因为计算机器提供的计算机器集一样。但是，这两种类型的机器集是在 Machine API 中定义的独立自定义资源，并在其架构和功能中有几个基本区别。

12.1.1. Control Plane Machine Set Operator 概述

Control Plane Machine Set Operator 使用 **ControlPlaneMachineSet** 自定义资源(CR) 自动管理 OpenShift Container Platform 集群中的 control plane 机器资源。

当集群 control plane 机器集设置为 **Active** 时，Operator 可确保集群具有正确的带有指定配置的 control plane 机器数量。这允许自动替换降级 control plane 机器，并推出对 control plane 的更改。

集群只有一个 control plane 机器集，Operator 则仅管理 **openshift-machine-api** 命名空间中的对象。

12.1.1.1. Control Plane Machine Set Operator 限制

Control Plane Machine Set Operator 有以下限制：

- 仅支持 Amazon Web Services (AWS)、Google Cloud Platform (GCP)、IBM Power® Virtual Server、Microsoft Azure、Nanix、VMware vSphere 和 Red Hat OpenStack Platform (RHOSP) 集群。
- 没有代表 control plane 节点的预先存在的机器的集群无法使用 control plane 机器集，或者在安装后启用使用 control plane 机器集。通常，只有在使用安装程序置备的基础架构安装集群时才存在已存在的 control plane 机器。
  要确定集群是否有所需的已存在的 control plane 机器，请以具有管理员特权的用户身份运行以下命令：

```
$ oc get machine \
  -n openshift-machine-api \
  -l machine.openshift.io/cluster-api-machine-role=master
```

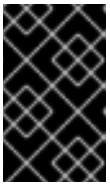
显示预先存在的 control plane 机器的输出示例

```
NAME                 PHASE     TYPE        REGION      ZONE         AGE
<infrastructure_id>-master-0   Running   m6i.xlarge   us-west-1   us-west-1a   5h19m
<infrastructure_id>-master-1   Running   m6i.xlarge   us-west-1   us-west-1b   5h19m
<infrastructure_id>-master-2   Running   m6i.xlarge   us-west-1   us-west-1a   5h19m
```

缺少预先存在的 control plane 机器的输出示例

```
No resources found in openshift-machine-api namespace.
```

- Operator 需要 Machine API Operator 可以正常工作，因此不支持使用手动置备的机器。当安装带有为一个平台手动置备的机器（这会创建一个主动生成的 **ControlPlaneMachineSet** CR）的 OpenShift Container Platform 集群时，您必须删除定义 control plane 机器集的 Kubernetes 清单文件，如安装过程所示。
- 仅支持具有三个 control plane 机器的集群。
- 不支持 control plane 的水平扩展。
- 在 Ephemeral OS 磁盘上部署 Azure control plane 机器会增加数据丢失的风险，且不被支持。
- 不支持将 control plane 机器部署为 AWS Spot 实例、GCP 抢占虚拟机或 Azure Spot 虚拟机。



重要

尝试将 control plane 机器部署为 AWS Spot 实例、GCP 抢占虚拟机或 Azure Spot 虚拟机可能会导致集群丢失 etcd 仲裁。丢失了所有 control plane 机器的集群无法恢复。

- 不支持在安装过程中或之前更改 control plane 机器集。您必须仅在安装后对 control plane 机器集进行任何更改。

12.1.2. 其他资源

- Control Plane Machine Set Operator 参考
- **ControlPlaneMachineSet** 自定义资源

12.2. CONTROL PLANE 机器集入门

control plane 机器集入门的过程取决于集群中的 **ControlPlaneMachineSet** 自定义资源 (CR) 的状态。

生成的具有活跃状态的 CR 的集群

生成的具有活跃状态的 CR 的集群默认使用 control plane 机器集。不需要管理员操作。

具有生成的不活跃 CR 的集群

对于包含具有生成的不活跃 CR 的集群，您必须检查 CR 配置并激活 CR。

没有生成的 CR 的集群

对于不包含生成的 CR 的集群，您必须为集群 创建并激活具有适当配置的 CR。

如果不确定集群中 **ControlPlaneMachineSet** CR 的状态，您可以验证 CR 状态。

12.2.1. 支持的云供应商

在 OpenShift Container Platform 4.16 中，Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、Nutanix 和 VMware vSphere 集群支持 control plane 机器集。

安装后 control plane 机器集的状态取决于您的云供应商和集群中安装的 OpenShift Container Platform 版本。

表 12.1. OpenShift Container Platform 4.16 的 control plane 机器集实现

云供应商	默认激活	生成的 CR	需要手动 CR
Amazon Web Services (AWS)	X [1]	X	
Google Cloud Platform (GCP)	X [2]	X	
Microsoft Azure	X [2]	X	
Nutanix	X [3]	X	
Red Hat OpenStack Platform(RHOSP)	X [3]	X	
VMware vSphere	X [4]	X	

1. 从 4.11 或更早版本升级的 AWS 集群需要激活 CR。
2. 从 4.12 或更早版本升级的 GCP 和 Azure 集群需要激活 CR。
3. 从 4.13 或更早版本升级的 Nutanix 和 RHOSP 集群 需要激活 CR。
4. 从 4.15 或更早版本升级的 vSphere 集群需要激活 CR。

12.2.2. 检查 control plane 机器设置自定义资源状态

您可以验证 **ControlPlaneMachineSet** 自定义资源 (CR) 是否存在以及其状态。

流程

- 运行以下命令确定 CR 的状态：

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
  --namespace openshift-machine-api
```

  - **Active** 的结果表示 **ControlPlaneMachineSet** CR 存在并被激活。不需要管理员操作。
  - **Inactive** 表示 **ControlPlaneMachineSet** CR 存在但没有激活。
  - **NotFound** 表示没有现有的 **ControlPlaneMachineSet** CR。

后续步骤

要使用 control plane 机器集，您必须确保集群有正确设置的 **ControlPlaneMachineSet** CR。

- 如果您的集群有一个现有的 CR，您必须验证 CR 中的配置是否正确。
- 如果集群没有现有的 CR，则必须为集群创建一个带有正确配置的 CR。

12.2.3. 激活 control plane 机器集自定义资源

要使用 control plane 机器集，您必须确保集群有正确设置的 **ControlPlaneMachineSet** 自定义资源 (CR)。在具有生成的 CR 集群中，您必须验证 CR 中的配置对于您的集群是否是正确的，并激活它。

注意

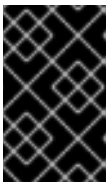
如需有关 CR 中参数的更多信息，请参阅"Control plane 机器集配置"。

流程

1. 运行以下命令，查看 CR 的配置：

```
$ oc --namespace openshift-machine-api edit controlplanemachineset.machine.openshift.io cluster
```

2. 更改集群配置不正确的字段的值。
3. 当配置正确时，通过将 **.spec.state** 字段设置为 **Active** 并保存您的更改来激活 CR。



重要

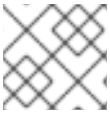
要激活 CR，您必须在用来更新 CR 配置的同一 **oc edit** 会话中将 **.spec.state** 字段改为 **Active**。如果 CR 保存为 **Inactive**，control plane 机器集生成器会将 CR 重置为其原始设置。

其他资源

- control plane 机器集配置

12.2.4. 创建 control plane 机器集自定义资源

要使用 control plane 机器集，您必须确保集群有正确设置的 **ControlPlaneMachineSet** 自定义资源 (CR)。在没有生成的 CR 的集群中，您必须手动创建 CR 并激活它。



注意

如需有关 CR 的结构和参数的更多信息，请参阅"Control plane 机器集配置"。

流程

1. 使用以下模板创建 YAML 文件：

control plane 机器集 CR YAML 文件模板

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
  replicas: 3
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <cluster_id> 1
```

```
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
state: Active 2
strategy:
  type: RollingUpdate 3
template:
  machineType: machines_v1beta1_machine_openshift_io
  machines_v1beta1_machine_openshift_io:
    failureDomains:
      platform: <platform> 4
      <platform_failure_domains> 5
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <cluster_id> 6
        machine.openshift.io/cluster-api-machine-role: master
        machine.openshift.io/cluster-api-machine-type: master
    spec:
      providerSpec:
        value:
          <platform_provider_spec> 7
```

1. 指定基于置备集群时所设置的集群 ID 的基础架构 ID。在创建 **ControlPlaneMachineSet** CR 时，您必须指定这个值。如果已安装 OpenShift CLI（**oc**）软件包，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. 指定 Operator 的状态。当状态为 **Inactive** 时，Operator 无法正常工作。您可以通过将值设置为 **Active** 来激活 Operator。

   重要

   在激活 CR 前，您必须确保其配置正确并满足集群要求。

3. 指定集群的更新策略。有效值为 **OnDelete** 和 **RollingUpdate**。默认值为 **RollingUpdate**。有关更新策略的更多信息，请参阅"更新 control plane 配置"。

4. 指定您的云供应商平台名称。有效值为 **AWS**、**Azure**、**GCP**、**Nutanix**、**VSphere** 和 **OpenStack**。

5. 为集群添加 **<platform_failure_domains>** 配置。本节的格式和值特定于提供程序。如需更多信息，请参阅您的云供应商的故障域配置示例。

6. 指定基础架构 ID。

7. 为集群添加 **<platform_provider_spec>** 配置。本节的格式和值特定于提供程序。如需更多信息，请参阅云供应商的供应商规格示例。

2. 请参阅 control plane 机器集 CR 的示例 YAML，并使用适合集群配置的值填充该文件。

3. 请参阅云供应商的示例故障域配置和示例供应商规格，并使用适当的值更新您的文件的这些部分。

4. 当配置正确时，通过将 **.spec.state** 字段设置为 **Active** 并保存您的更改来激活 CR。
5. 运行以下命令，从 YAML 文件创建 CR：

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

其中 **<control_plane_machine_set>** 是包含 CR 配置的 YAML 文件的名称。

其他资源

- 更新 control plane 配置
- control plane 机器集配置
- 特定于供应商的配置选项

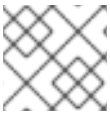
12.3. 使用 CONTROL PLANE 机器集管理 CONTROL PLANE 机器

control plane 机器集自动执行 control plane 管理的几个重要方面。

12.3.1. 更新 control plane 配置

您可以通过更新 control plane 机器集自定义资源 (CR) 中的规格来更改 control plane 中机器的配置。

Control Plane Machine Set Operator 监控 control plane 机器，并将其配置与 control plane 机器集 CR 中的规格进行比较。当 CR 中的规格和 control plane 机器的配置之间存在不同时，Operator 会标记 control plane 机器以进行替换。



注意

如需有关 CR 中参数的更多信息，请参阅"Control plane 机器集配置"。

先决条件

- 集群有一个已激活并可正常工作的 Control Plane Machine Set Operator。

流程

1. 运行以下命令来编辑 control plane 机器集 CR：

```
$ oc edit controlplanemachineset.machine.openshift.io cluster \
  -n openshift-machine-api
```

2. 更改您要在集群配置中更新的任何字段的值。
3. 保存您的更改。

后续步骤

- 对于使用默认 **RollingUpdate** 更新策略的集群，control plane 机器集会自动将更改传播到 control plane 配置。
- 对于配置为使用 **OnDelete** 更新策略的集群，您必须手动替换 control plane 机器。

12.3.1.1. 自动更新 control plane 配置

RollingUpdate 更新策略会自动将更改传播到 control plane 配置。此更新策略是 control plane 机器集的默认配置。

对于使用 **RollingUpdate** 更新策略的集群，Operator 会使用 CR 中指定的配置创建一个替代 control plane 机器。当替换的 control plane 机器就绪时，Operator 会删除标记为替换的 control plane 机器。然后，替换机器加入 control plane。

如果多个 control plane 机器标记为替换，Operator 会一次重复这个替换过程来防止 etcd 健康状况，直到替换每台机器为止。

12.3.1.2. 手动更新 control plane 配置

您可以通过手动替换机器，使用 **OnDelete** 更新策略将更改传播到 control plane 配置。手动替换机器允许您在更广泛地应用更改前在单个机器上测试对配置的更改。

对于配置为使用 **OnDelete** 更新策略的集群，Operator 会在删除现有机器时创建一个替换 control plane 机器。当替换的 control plane 机器就绪时，etcd Operator 允许删除现有机器。然后，替换机器加入 control plane。

如果删除了多个 control plane 机器，Operator 会同时创建所有必需的替换机器。Operator 通过防止一次从 control plane 中删除多个机器来维护 etcd 健康状况。

12.3.2. 替换 control plane 机器

要替换具有 control plane 机器集的集群中的 control plane 机器，请手动删除机器。control plane 机器集使用 control plane 机器集自定义资源 (CR) 中的规格替换已删除的机器。

先决条件

- 如果您的集群在 Red Hat OpenStack Platform (RHOSP)上运行，且需要撤离计算服务器，如升级，您必须运行以下命令来禁用机器运行的 RHOSP 计算节点：

  ```
  $ openstack compute service set <target_node_host_name> nova-compute --disable
  ```

  如需更多信息，请参阅 RHOSP 文档中的准备迁移。

流程

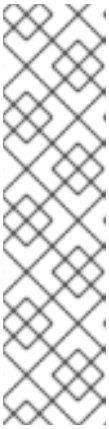
1. 运行以下命令列出集群中的 control plane 机器：

   ```
   $ oc get machines \
     -l machine.openshift.io/cluster-api-machine-role==master \
     -n openshift-machine-api
   ```

2. 运行以下命令来删除 control plane 机器：

   ```
   $ oc delete machine \
     -n openshift-machine-api \
     <control_plane_machine_name> 1
   ```

   1 指定要删除的 control plane 机器的名称。



注意

如果删除了多个 control plane 机器，control plane 机器集会根据配置的更新策略替换它们：

- 对于使用默认 **RollingUpdate** 更新策略的集群，Operator 会一次替换一台机器，直到替换每台机器为止。
- 对于配置为使用 **OnDelete** 更新策略的集群，Operator 会同时创建所有所需的替换机器。

这两种策略在 control plane 机器替换过程中维护 etcd 健康状况。

12.3.3. 其他资源

- control plane 机器集配置
- 特定于供应商的配置选项

12.4. CONTROL PLANE 机器集配置

此 YAML 片断示例显示 control plane 机器集自定义资源 (CR) 的基本结构。

12.4.1. control plane 机器集自定义资源的 YAML 示例

ControlPlaneMachineSet CR 的基础构建方式与所有平台相同。

ControlPlaneMachineSet CR YAML 文件示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster 1
  namespace: openshift-machine-api
spec:
  replicas: 3 2
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <cluster_id> 3
      machine.openshift.io/cluster-api-machine-role: master
      machine.openshift.io/cluster-api-machine-type: master
  state: Active 4
  strategy:
    type: RollingUpdate 5
  template:
    machineType: machines_v1beta1_machine_openshift_io
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: <platform> 6
        <platform_failure_domains> 7
      metadata:
        labels:
          machine.openshift.io/cluster-api-cluster: <cluster_id>
          machine.openshift.io/cluster-api-machine-role: master
```

```
    machine.openshift.io/cluster-api-machine-type: master
spec:
  providerSpec:
    value:
      <platform_provider_spec> 8
```

1. 指定 **ControlPlaneMachineSet** CR 的名称，即 **集群**。不要更改这个值。
2. 指定 control plane 机器的数量。仅支持具有三个 control plane 机器的集群，因此 **replicas** 值为 **3**。不支持水平扩展。不要更改这个值。
3. 指定基于置备集群时所设置的集群 ID 的基础架构 ID。在创建 **ControlPlaneMachineSet** CR 时，您必须指定这个值。如果已安装 OpenShift CLI（**oc**）软件包，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
   ```

4. 指定 Operator 的状态。当状态为 **Inactive** 时，Operator 无法正常工作。您可以通过将值设置为 **Active** 来激活 Operator。

   重要

   在激活 Operator 前，您必须确保 **ControlPlaneMachineSet** CR 配置适合您的集群要求。有关激活 Control Plane Machine Set Operator 的更多信息，请参阅"使用 control plane 机器集入门"。

5. 指定集群的更新策略。允许的值是 **OnDelete** 和 **RollingUpdate**。默认值为 **RollingUpdate**。有关更新策略的更多信息，请参阅"更新 control plane 配置"。
6. 指定云供应商平台名称。不要更改这个值。
7. 为集群指定 **<platform_failure_domains>** 配置。本节的格式和值特定于提供程序。如需更多信息，请参阅您的云供应商的故障域配置示例。
8. 为集群指定 **<platform_provider_spec>** 配置。本节的格式和值特定于提供程序。如需更多信息，请参阅云供应商的供应商规格示例。

其他资源

- control plane 机器集入门
- 更新 control plane 配置

12.4.2. 特定于供应商的配置选项

control plane 机器集清单的 **<platform_provider_spec>** 和 **<platform_failure_domains>** 部分是特定于供应商的。有关集群的特定于供应商的配置选项，请参阅以下资源：

- Amazon Web Services 的 control plane 配置选项
- Google Cloud Platform 的 control plane 配置选项
- Microsoft Azure 的 control plane 配置选项

- Nutanix 的 control plane 配置选项
- Red Hat OpenStack Platform (RHOSP) 的 control plane 配置选项
- VMware vSphere 的 control plane 配置选项

12.5. CONTROL PLANE 机器的配置选项

12.5.1. Amazon Web Services 的 control plane 配置选项

您可以更改 Amazon Web Services (AWS) control plane 机器的配置，并通过更新 control plane 机器集中的值来启用功能。当您将更新保存到 control plane 机器集时，Control Plane Machine Set Operator 会根据您配置的更新策略 control plane 机器。

12.5.1.1. 用于配置 Amazon Web Services 集群的 YAML 示例

以下示例 YAML 片断显示 AWS 集群的供应商规格和故障域配置。

12.5.1.1.1. AWS 供应商规格示例

当您为现有集群创建 control plane 机器集时，供应商规格必须与安装程序创建的 control plane 机器自定义资源 (CR) 中的 **providerSpec** 配置匹配。您可以省略 CR 的故障域部分中设置的任何字段。

在以下示例中，**<cluster_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 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：

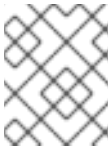
```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

AWS providerSpec 值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            ami:
              id: ami-<ami_id_string> (1)
            apiVersion: machine.openshift.io/v1beta1
            blockDevices:
            - ebs: (2)
                encrypted: true
                iops: 0
                kmsKey:
                  arn: ""
                volumeSize: 120
                volumeType: gp3
```

```
      credentialsSecret:
        name: aws-cloud-credentials 3
      deviceIndex: 0
      iamInstanceProfile:
        id: <cluster_id>-master-profile 4
      instanceType: m6i.xlarge 5
      kind: AWSMachineProviderConfig 6
      loadBalancers: 7
      - name: <cluster_id>-int
        type: network
      - name: <cluster_id>-ext
        type: network
      metadata:
        creationTimestamp: null
      metadataServiceOptions: {}
      placement: 8
        region: <region> 9
        availabilityZone: "" 10
        tenancy: 11
      securityGroups:
      - filters:
        - name: tag:Name
          values:
          - <cluster_id>-master-sg 12
      subnet: {} 13
      userDataSecret:
        name: master-user-data 14
```

1. 指定集群的 Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Images (AMI) ID。AMI 必须与集群属于同一区域。如果要使用 AWS Marketplace 镜像，则必须从 AWS Marketplace 完成 OpenShift Container Platform 订阅来获取您所在地区的 AMI ID。
2. 指定加密的 EBS 卷的配置。
3. 指定集群的 secret 名称。不要更改这个值。
4. 指定 AWS Identity and Access Management (IAM)实例配置集。不要更改这个值。
5. 为 control plane 指定 AWS 实例类型。
6. 指定云供应商平台类型。不要更改这个值。
7. 指定集群的内部 (**int**) 和外部 (**ext**) 负载均衡器。



注意

您可以在私有 OpenShift Container Platform 集群中省略外部 (**ext**) 负载均衡器参数。

8. 指定在 AWS 中创建 control plane 实例的位置。
9. 指定集群的 AWS 区域。
10.

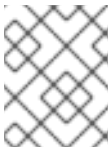
此参数在故障域中配置，此处显示了一个空值。如果为此参数指定的值与故障域中的值不同，Control Plane Machine Set Operator 会使用故障域中的值覆盖它。

11 指定 control plane 的 AWS Dedicated 实例配置。如需更多信息，请参阅 AWS 文档有关 Dedicated 实例。以下值有效：

- **default** ：Dedicated 实例在共享硬件上运行。
- **dedicated** ：Dedicated 实例在单租户硬件上运行。
- **host** ：Dedicated 实例在 Dedicated 主机上运行，它是一个带有您可以控制的配置的隔离服务器。

12 指定 control plane 机器安全组。

13 此参数在故障域中配置，此处显示了一个空值。如果为此参数指定的值与故障域中的值不同，Control Plane Machine Set Operator 会使用故障域中的值覆盖它。



注意

如果故障域配置没有指定值，则会使用供应商规格中的值。在故障域中配置子网会覆盖供应商规格中的子网值。

14 指定 control plane 用户数据 secret。不要更改这个值。

12.5.1.1.2. AWS 故障域配置示例

故障域的 control plane 机器集概念与现有 *可用区(AZ)* 的 AWS 概念类似。**ControlPlaneMachineSet** CR 尽可能将 control plane 机器分散到多个故障域中。

在 control plane 机器集中配置 AWS 故障域时，您必须指定可用区名称和要使用的子网。

AWS 故障域值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        aws:
        - placement:
            availabilityZone: <aws_zone_a> 1
          subnet: 2
            filters:
            - name: tag:Name
              values:
              - <cluster_id>-private-<aws_zone_a> 3
            type: Filters 4
```

```
      - placement:
          availabilityZone: <aws_zone_b> 5
        subnet:
          filters:
          - name: tag:Name
            values:
            - <cluster_id>-private-<aws_zone_b> 6
          type: Filters
        platform: AWS 7
# ...
```

1. 为第一个故障域指定一个 AWS 可用区。
2. 指定子网配置。在本例中，子网类型是 **Filters**，因此有一个 **filters** 片段。
3. 使用基础架构 ID 和 AWS 可用性区域指定第一个故障域的子网名称。
4. 指定子网类型。允许的值包括：**ARN**、**Filters** 和 **ID**。默认值为 **Filters**。
5. 使用基础架构 ID 和 AWS 可用域指定额外故障域的子网名称。
6. 为额外的故障域指定集群的基础架构 ID 和 AWS 可用区。
7. 指定云供应商平台名称。不要更改这个值。

12.5.1.2. 为 control plane 机器启用 Amazon Web Services 功能

您可以通过更新 control plane 机器集中的值来启用功能。

12.5.1.2.1. 将 API 服务器限制为私有

将集群部署到 Amazon Web Services (AWS) 后，您可以将 API 服务器重新配置为只使用私有区。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 使用具有 **admin** 权限的用户登陆到 web 控制台。

流程

1. 在云供应商的 web 门户或控制台中，执行以下操作：
   a. 找到并删除相关的负载均衡器组件：
      - 对于 AWS，删除外部负载均衡器。私有区的 API DNS 条目已指向内部负载均衡器，它使用相同的配置，因此您无需修改内部负载均衡器。
   b. 在公共区中删除 **api.$clustername.$yourdomain** DNS 条目。
2. 通过删除 control plane 机器集自定义资源中的以下指示行来删除外部负载均衡器：

```
# ...
providerSpec:
  value:
```

```
# ...
    loadBalancers:
    - name: lk4pj-ext 1
      type: network 2
    - name: lk4pj-int
      type: network
# ...
```

1 删除外部负载均衡器的 **name** 值，它以 **-ext** 结尾。

2 删除外部负载均衡器的 **type** 值。

其他资源

- 将 Ingress Controller 端点发布范围配置为 Internal

12.5.1.2.2. 使用 control plane 机器集更改 Amazon Web Services 实例类型

您可以通过更新 control plane 机器集自定义资源 (CR) 中的规格来更改 control plane 机器使用的 Amazon Web Services (AWS) 实例类型。

先决条件

- 您的 AWS 集群使用 control plane 机器集。

流程

1. 编辑 **providerSpec** 字段中的以下行：

```
providerSpec:
  value:
    ...
    instanceType: <compatible_aws_instance_type> 1
```

1 使用与之前选择相同的基础指定较大的 AWS 实例类型。例如，您可以将 **m6i.xlarge** 更改为 **m6i.2xlarge** 或 **m6i.4xlarge**。

2. 保存您的更改。

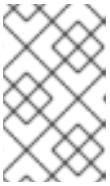
12.5.1.2.3. 使用机器集，为 Elastic Fabric Adapter 实例分配机器到放置组

您可以配置机器集，在现有 AWS PG 中的 Elastic Fabric Adapter (EFA) 实例上部署机器。

EFA 实例不需要放置组，您可以使用放置组作为配置 EFA 以外的目的。本例使用这两者来演示能够提高指定放置组内机器的网络性能的配置。

先决条件

- 您在 AWS 控制台中创建了放置组。



注意

确保您创建的放置组类型的规则和限制与预期的用例兼容。control plane 机器集尽可能将 control plane 机器分散到多个故障域中。要将放置组用于 control plane，您必须使用可跨越多个可用区的放置组类型。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 编辑 **providerSpec** 字段中的以下行：

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          instanceType: <supported_instance_type> 1
          networkInterfaceType: EFA 2
          placement:
            availabilityZone: <zone> 3
            region: <region> 4
          placementGroupName: <placement_group> 5
# ...
```

1. 指定支持 EFA 的实例类型。
2. 指定 **EFA** 网络接口类型。
3. 指定区域，如 **us-east-1a**。
4. 指定区域，如 **us-east-1**。
5. 指定要部署机器的现有 AWS PG 的名称。

验证

- 在 AWS 控制台中，找到机器集创建的机器，并在机器属性中验证以下内容：
  - placement group 字段具有您为机器集中的 **placementGroupName** 参数指定的值。
  - 接口类型字段表示它使用 EFA。

12.5.1.2.4. Amazon EC2 实例元数据服务的机器集选项

您可以使用机器集创建使用 Amazon EC2 实例元数据服务 (IMDS) 的特定版本的机器。机器集可以创建允许使用 IMDSv1 和 IMDSv2 的机器或需要使用 IMDSv2 的机器。

注意

只有在 OpenShift Container Platform 版本 4.7 或更高版本中创建的 AWS 集群上才支持使用 IMDSv2。

重要

在配置机器集来创建需要 IMDSv2 的机器前，请确保与 AWS 元数据服务交互的工作负载都支持 IMDSv2。

12.5.1.2.4.1. 使用机器集配置 IMDS

您可以通过在机器集 YAML 文件中添加或编辑 **metadataServiceOptions.authentication**，来指定是否需要使用 IMDSv2。

先决条件

- 要使用 IMDSv2，您的 AWS 集群必须使用 OpenShift Container Platform 版本 4.7 或更高版本创建。

流程

- 在 **providerSpec** 字段中添加或编辑以下行：

```
providerSpec:
  value:
    metadataServiceOptions:
      authentication: Required 1
```

1 为了要求 IMDSv2，请将参数值设置为 **Required**。要允许使用 IMDSv1 和 IMDSv2，请将参数值设置为 **Optional**。如果没有指定值，则允许 IMDSv1 和 IMDSv2。

12.5.1.2.5. 将机器部署为 Dedicated 实例的机器集

您可以创建在 AWS 上运行的机器集，该机器将机器部署为 Dedicated 实例。专用实例在专用于单一客户的硬件上运行虚拟私有云（VPC）。这些 Amazon EC2 实例在主机硬件级别被物理隔离。Dedicated 实例的隔离也会存在，即使实例属于链接到一个 Forer 帐户的不同 AWS 帐户。但是，其他未专用实例如果属于同一 AWS 帐户，则可以与 Dedicated 实例共享硬件。

Machine API 支持具有公共或专用租期的实例。具有公共租期的实例在共享硬件上运行。公共租期是默认租期。具有专用租期的实例在单租户硬件上运行。

12.5.1.2.5.1. 使用机器集创建 Dedicated 实例

您可以使用 Machine API 集成来运行由 Dedicated 实例支持的机器。设置机器设置 YAML 文件中的 **tenancy** 字段，以便在 AWS 上启动 Dedicated 实例。

流程

- 在 **providerSpec** 字段中指定专用租户：

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

12.5.2. Microsoft Azure 的 control plane 配置选项

您可以通过更新 control plane 机器集中的值来更改 Microsoft Azure control plane 机器的配置并启用功能。当您将更新保存到 control plane 机器集时，Control Plane Machine Set Operator 会根据您配置的更新策略 control plane 机器。

12.5.2.1. 用于配置 Microsoft Azure 集群的 YAML 示例

以下示例 YAML 片断显示 Azure 集群的供应商规格和故障域配置。

12.5.2.1.1. Azure 供应商规格示例

当您为现有集群创建 control plane 机器集时，供应商规格必须与安装程序创建的 control plane **Machine** CR 中的 **providerSpec** 配置匹配。您可以省略 CR 的故障域部分中设置的任何字段。

在以下示例中，**<cluster_id>** 是基础架构 ID，它基于您在置备集群时设定的集群 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：

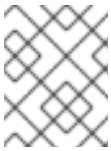
```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

Azure providerSpec 值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            acceleratedNetworking: true
            apiVersion: machine.openshift.io/v1beta1
            credentialsSecret:
              name: azure-cloud-credentials 1
              namespace: openshift-machine-api
            diagnostics: {}
            image: 2
              offer: ""
              publisher: ""
              resourceID: /resourceGroups/<cluster_id>-
rg/providers/Microsoft.Compute/galleries/gallery_<cluster_id>/images/<cluster_id>-
gen2/versions/412.86.20220930 3
              sku: ""
              version: ""
            internalLoadBalancer: <cluster_id>-internal 4
```

```
    kind: AzureMachineProviderSpec 5
    location: <region> 6
    managedIdentity: <cluster_id>-identity
    metadata:
      creationTimestamp: null
      name: <cluster_id>
    networkResourceGroup: <cluster_id>-rg
    osDisk: 7
      diskSettings: {}
      diskSizeGB: 1024
      managedDisk:
        storageAccountType: Premium_LRS
      osType: Linux
    publicIP: false
    publicLoadBalancer: <cluster_id> 8
    resourceGroup: <cluster_id>-rg
    subnet: <cluster_id>-master-subnet 9
    userDataSecret:
      name: master-user-data 10
    vmSize: Standard_D8s_v3
    vnet: <cluster_id>-vnet
    zone: "1" 11
```

1. 指定集群的 secret 名称。不要更改这个值。
2. 指定 control plane 机器集的镜像详情。
3. 指定与实例类型兼容的镜像。安装程序创建的 Hyper-V 生成 V2 镜像具有 **-gen2** 后缀，而 V1 镜像则与没有后缀的名称相同。
4. 指定 control plane 的内部负载均衡器。此字段可能无法预先配置，但在 **ControlPlaneMachineSet** 和 control plane **Machine** CR 中都是必需的。
5. 指定云供应商平台类型。不要更改这个值。
6. 指定要放置 control plane 机器的区域。
7. 指定 control plane 的磁盘配置。
8. 指定 control plane 的公共负载均衡器。



注意

您可以在具有用户定义的出站路由的私有 OpenShift Container Platform 集群上省略 **publicLoadBalancer** 参数。

9. 指定 control plane 的子网。
10. 指定 control plane 用户数据 secret。不要更改这个值。
11. 指定将单个区用于所有故障域的集群区配置。

注意

如果集群配置为为每个故障域使用不同的区，则此参数会在故障域中配置。如果您在为每个故障域使用不同区时在供应商规格中指定这个值，则 Control Plane Machine Set Operator 会忽略它。

12.5.2.1.2. Azure 故障域配置示例

故障域的 control plane 机器集概念与 *Azure 可用区*的现有 Azure 概念类似。**ControlPlaneMachineSet** CR 尽可能将 control plane 机器分散到多个故障域中。

在 control plane 机器集中配置 Azure 故障域时，您必须指定可用区名称。Azure 集群使用跨越多个区的单一子网。

Azure 故障域值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        azure:
        - zone: "1" (1)
        - zone: "2"
        - zone: "3"
        platform: Azure (2)
# ...
```

(1) 每个区实例都指定故障域的 Azure 可用区。

注意

如果集群配置为对所有故障域使用单个区，则 **zone** 参数在供应商规格中配置，而不是在故障域配置中。

(2) 指定云供应商平台名称。不要更改这个值。

12.5.2.2. 为 control plane 机器启用 Microsoft Azure 功能

您可以通过更新 control plane 机器集中的值来启用功能。

12.5.2.2.1. 将 API 服务器限制为私有

将集群部署到 Amazon Web Services (AWS) 后，您可以将 API 服务器重新配置为只使用私有区。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 使用具有 **admin** 权限的用户登陆到 web 控制台。

流程

1. 在云供应商的 web 门户或控制台中，执行以下操作：
   a. 找到并删除相关的负载均衡器组件：
   b. 在公共区中删除 **api.$clustername.$yourdomain** DNS 条目。
2. 通过删除 control plane 机器集自定义资源中的以下指示行来删除外部负载均衡器：

```
# ...
providerSpec:
  value:
# ...
    loadBalancers:
    - name: lk4pj-ext 1
      type: network 2
    - name: lk4pj-int
      type: network
# ...
```

1 删除外部负载均衡器的 **name** 值，它以 **-ext** 结尾。

2 删除外部负载均衡器的 **type** 值。

其他资源

- 将 Ingress Controller 端点发布范围配置为 Internal

12.5.2.2.2. 使用 Azure Marketplace 产品

您可以创建在 Azure 上运行的机器集，以部署使用 Azure Marketplace 产品的机器。要使用此产品，您必须首先获取 Azure Marketplace 镜像。在获取您的镜像时，请考虑以下事项：

- 虽然镜像相同，但 Azure Marketplace publisher 根据您的区域。如果您位于北美，请将 **redhat** 指定为发布者。如果您位于 EMEA，请将 **redhat-limited** 指定为发布者。
- 此项优惠包括 **rh-ocp-worker** SKU 和 **rh-ocp-worker-gen1** SKU。**rh-ocp-worker** SKU 代表 Hyper-V 生成版本 2 虚拟机镜像。OpenShift Container Platform 中使用的默认实例类型与版本 2 兼容。如果您计划使用与版本 1 兼容的实例类型，请使用与 **rh-ocp-worker-gen1** SKU 关联的镜像。**rh-ocp-worker-gen1** SKU 代表 Hyper-V 版本 1 虚拟机镜像。

重要

在使用 64 位 ARM 实例的集群上不支持使用 Azure marketplace 安装镜像。

先决条件

- 已安装 Azure CLI 客户端 **(az)**。
- 您的 Azure 帐户为产品授权，您使用 Azure CLI 客户端登录到此帐户。

流程

1. 运行以下命令之一，显示所有可用的 OpenShift Container Platform 镜像：
   - 北美：

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat -o table
     ```

     输**出示例**

     ```
     Offer          Publisher      Sku                Urn                                                                  Version
     -------------  -------------  -----------------  -------------------------------------------------------------------
     ----  -----------------
     rh-ocp-worker  RedHat         rh-ocp-worker      RedHat:rh-ocp-worker:rh-ocp-
     worker:413.92.2023101700           413.92.2023101700
     rh-ocp-worker  RedHat         rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
     gen1:413.92.2023101700       413.92.2023101700
     ```

   - 欧洲、中东和非洲地区：

     ```
     $  az vm image list --all --offer rh-ocp-worker --publisher redhat-limited -o table
     ```

     输**出示例**

     ```
     Offer          Publisher      Sku                Urn
     Version
     -------------  -------------  -----------------  -------------------------------------------------------------------
     ----         -----------------
     rh-ocp-worker  redhat-limited  rh-ocp-worker      redhat-limited:rh-ocp-worker:rh-ocp-
     worker:413.92.2023101700           413.92.2023101700
     rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
     worker-gen1:413.92.2023101700       413.92.2023101700
     ```

     注意

     使用可用于 compute 和 control plane 节点的最新镜像。如果需要，您的虚拟机会在安装过程中自动升级。

2. 运行以下命令之一检查您的所提供的镜像：
   - 北美：

     ```
     $ az vm image show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
     ```

   - 欧洲、中东和非洲地区：

     ```
     $ az vm image show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
     ```

3. 运行以下命令之一查看提供的术语：

- 北美：

```
$ az vm image terms show --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- 欧洲、中东和非洲地区：

```
$ az vm image terms show --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

4. 运行以下命令之一接受产品条款：

- 北美：

```
$ az vm image terms accept --urn redhat:rh-ocp-worker:rh-ocp-worker:<version>
```

- 欧洲、中东和非洲地区：

```
$ az vm image terms accept --urn redhat-limited:rh-ocp-worker:rh-ocp-worker:<version>
```

5. 记录您所提供的镜像详情，特别是 **publisher**, **offer**, **sku**, 和 **version** 的值。

6. 使用您提供的镜像详情，在机器集 YAML 文件的 **providerSpec** 部分添加以下参数：

Azure Marketplace 机器的 providerSpec 镜像值示例

```
providerSpec:
  value:
    image:
      offer: rh-ocp-worker
      publisher: redhat
      resourceID: ""
      sku: rh-ocp-worker
      type: MarketplaceWithPlan
      version: 413.92.2023101700
```

12.5.2.2.3. 启用 Azure 引导诊断

您可以在机器集创建的 Azure 机器上启用引导诊断。

先决条件

- 已有 Microsoft Azure 集群。

流程

- 将适用于您的存储类型的 **diagnostics** 配置添加到机器集 YAML 文件中的 **providerSpec** 字段中：

  - 对于 Azure Managed 存储帐户：

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: AzureManaged 1
```

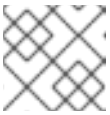
1 指定 Azure Managed 存储帐户。

- 对于 Azure Unmanaged 存储帐户：

```
providerSpec:
  diagnostics:
    boot:
      storageAccountType: CustomerManaged 1
      customerManaged:
        storageAccountURI: https://<storage-account>.blob.core.windows.net 2
```

1 指定 Azure Unmanaged 存储帐户。

2 将 **<storage-account>** 替换为存储帐户的名称。



注意

仅支持 Azure Blob Storage 数据服务。

验证

- 在 Microsoft Azure 门户上，查看机器集部署的机器的 **Boot diagnostics** 页面，并验证您可以看到机器的串行日志。

12.5.2.2.4. 使用计算磁盘部署机器的机器集作为数据磁盘

您可以创建在 Azure 上运行的机器集，该机器集用来部署带有巨型磁盘的机器。ultra 磁盘是高性能存储，用于要求最苛刻的数据工作负载。

其他资源

- Microsoft Azure ultra 磁盘文档

12.5.2.2.4.1. 使用机器集创建带有巨型磁盘的机器

您可以通过编辑机器集 YAML 文件在 Azure 上部署带有巨型磁盘的机器。

先决条件

- 已有 Microsoft Azure 集群。

流程

1. 运行以下命令，使用 **master** 数据 secret 在 **openshift-machine-api** 命名空间中创建自定义 secret：

```
$ oc -n openshift-machine-api \
get secret <role>-user-data \ 1
--template='{{index .data.userData | base64decode}}' | jq > userData.txt 2
```

1 将 **<role>** 替换为 **master**。

2 指定 **userData.txt** 作为新自定义 secret 的名称。

2. 在文本编辑器中，打开 **userData.txt** 文件，并在文件中找到最后的 **}** 字符。

   a. 在紧接下来的行中，添加一个 **,**

   b. 在 **,** 之后创建一个新行并添加以下配置详情：

```
"storage": {
  "disks": [ 1
    {
      "device": "/dev/disk/azure/scsi1/lun0", 2
      "partitions": [ 3
        {
          "label": "lun0p1", 4
          "sizeMiB": 1024, 5
          "startMiB": 0
        }
      ]
    }
  ],
  "filesystems": [ 6
    {
      "device": "/dev/disk/by-partlabel/lun0p1",
      "format": "xfs",
      "path": "/var/lib/lun0p1"
    }
  ]
},
"systemd": {
  "units": [ 7
    {
      "contents": "[Unit]\nBefore=local-
fs.target\n[Mount]\nWhere=/var/lib/lun0p1\nWhat=/dev/disk/by-
partlabel/lun0p1\nOptions=defaults,pquota\n[Install]\nWantedBy=local-fs.target\n", 8
      "enabled": true,
      "name": "var-lib-lun0p1.mount"
    }
  ]
}
```

1 您要作为 ultra 磁盘附加到节点的磁盘的配置详情。

2 指定您使用的机器集的 **dataDisks** 小节中定义的 **lun** 值。例如，如果机器集包含 **lun: 0**，请指定 **lun0**。您可以通过在这个配置文件中指定多个 **"disks"** 条目来初始化多个数据磁盘。如果您指定多个 **"disks"** 条目，请确保每个条目的 **lun** 值与机器集中的值匹配。

3 磁盘上新分区的配置详情。

4 为分区指定标签。使用分层的名称可能会有帮助，如 **lun0p1** 代表 **lun0** 的第一个分区。

5 指定分区的总大小（以 MiB 为单位）。

6 指定在格式化分区时要使用的文件系统。使用分区标签来指定分区。

7 指定一个 **systemd** 单元来在引导时挂载分区。使用分区标签来指定分区。您可以通过在这个配置文件中指定多个 **"partitions"** 条目来创建多个分区。如果指定多个 **"partitions"** 条目，则必须为每个条目指定一个 **systemd** 单元。

8 对于 **where**，指定 **storage.filesystems.path** 的值。对于 **What**，指定 **storage.filesystems.device** 的值。

3. 运行以下命令，将禁用模板值提取到名为 **disableTemplating.txt** 的文件：

```
$ oc -n openshift-machine-api get secret <role>-user-data \ 1
--template='{{index .data.disableTemplating | base64decode}}' | jq > disableTemplating.txt
```

1 将 **<role>** 替换为 **master**。

4. 运行以下命令组合 **userData.txt** 文件和 **disableTemplating.txt** 文件来创建数据 secret 文件：

```
$ oc -n openshift-machine-api create secret generic <role>-user-data-x5 \ 1
--from-file=userData=userData.txt \
--from-file=disableTemplating=disableTemplating.txt
```

1 对于 **<role>-user-data-x5**，请指定 secret 的名称。将 **<role>** 替换为 **master**。

5. 运行以下命令来编辑 control plane 机器集 CR：

```
$ oc --namespace openshift-machine-api edit controlplanemachineset.machine.openshift.io
cluster
```

6. 在指示的位置中添加以下行：

```
apiVersion: machine.openshift.io/v1beta1
kind: ControlPlaneMachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd 1
      providerSpec:
        value:
          ultraSSDCapability: Enabled 2
          dataDisks: 3
          - nameSuffix: ultrassd
            lun: 0
            diskSizeGB: 4
            deletionPolicy: Delete
            cachingType: None
            managedDisk:
              storageAccountType: UltraSSD_LRS
          userDataSecret:
            name: <role>-user-data-x5 4
```

-

1. 指定标签，用于选择此机器集创建的节点。此流程使用 **disk.ulssd** 用于这个值。
2. 3. 这些行支持使用 ultra 磁盘。对于 **dataDisks**，请包括整个小节。
4. 指定之前创建的用户数据 secret。将 **<role>** 替换为 **master**。

7. 保存您的更改。
   - 对于使用默认 **RollingUpdate** 更新策略的集群，Operator 会自动将更改传播到 control plane 配置。
   - 对于配置为使用 **OnDelete** 更新策略的集群，您必须手动替换 control plane 机器。

验证

1. 运行以下命令验证机器是否已创建：

```
$ oc get machines
```

   机器应处于 **Running** 状态。

2. 对于正在运行并附加节点的机器，请运行以下命令验证分区：

```
$ oc debug node/<node-name> -- chroot /host lsblk
```

   在这个命令中，**oc debug node/<node-name>** 会在节点 **<node-name>** 上启动一个 debugging shell，并传递一个带有 **--** 的命令。传递的命令 **chroot /host** 提供对底层主机操作系统二进制文件的访问，**lsblk** 显示连接至主机操作系统计算机的块设备。

后续步骤

- 要在 control plane 上使用 ultra 磁盘，请重新配置工作负载以使用 control plane 的 ultra 磁盘挂载点。

12.5.2.2.4.2. 启用 ultra 磁盘的机器集的故障排除资源

使用本节中的信息从您可能会遇到的问题了解和恢复。

12.5.2.2.4.2.1. 不正确的 ultra 磁盘配置

如果在机器集中指定 **ultraSSDCapability** 参数的配置不正确，则机器置备会失败。

例如，如果 **ultraSSDCapability** 参数设置为 **Disabled**，但在 **dataDisks** 参数中指定了 ultra 磁盘，则会出现以下出错信息：

```
StorageAccountType UltraSSD_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.
```

- 要解决这个问题，请验证机器集配置是否正确。

12.5.2.2.4.2.2. 不支持的磁盘参数

如果在机器集中指定与 ultra 磁盘不兼容的区域、可用性区域或实例大小，则机器置备会失败。检查日志中的以下出错信息：

```
failed to create vm <machine_name>: failure sending request for machine <machine_name>: cannot create vm: compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request: StatusCode=400 -- Original Error: Code="BadRequest" Message="Storage Account type 'UltraSSD_LRS' is not supported <more_information_about_why>."
```

- 要解决这个问题，请验证您是否在受支持的环境中使用此功能，以及机器设置配置是否正确。

12.5.2.2.4.2.3. 无法删除磁盘

如果因为数据磁盘无法按预期工作，则会删除大量磁盘，则机器会被删除，数据磁盘会孤立。如果需要，您必须手动删除孤立的磁盘。

12.5.2.2.5. 为机器集启用客户管理的加密密钥

您可以为 Azure 提供加密密钥，以便加密受管磁盘上的数据。您可以使用 Machine API 使用客户管理的密钥启用服务器端加密。

使用客户管理的密钥需要 Azure Key Vault、磁盘加密集和加密密钥。磁盘加密集必须在 Cloud Credential Operator（CCO）授予权限的资源组中。如果没有，则需要在磁盘加密集中授予额外的 reader 角色。

先决条件

- 创建 Azure Key Vault 实例。
- 创建磁盘加密集的实例。
- 授予磁盘加密集对密钥 vault 的访问权限。

流程

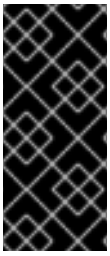
- 在机器集 YAML 文件中的 **providerSpec** 字段中配置磁盘加密集。例如：

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

其他资源

- Azure 文档中有关客户管理的密钥

12.5.2.2.6. 使用机器集为 Azure 虚拟机配置可信启动



重要

为 Azure 虚拟机使用可信启动只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

OpenShift Container Platform 4.16 支持 Azure 虚拟机 (VM) 的可信启动。通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的可信启动选项。例如，您可以将这些机器配置为使用 UEFI 安全功能，如安全引导或专用虚拟信任平台模块 (vTPM) 实例。

注意

有些功能组合会导致配置无效。

表 12.2. UEFI 功能组合兼容性

安全引导 [1]	vTPM[2]	有效配置
Enabled	Enabled	是
Enabled	Disabled	是
Enabled	省略	是
Disabled	Enabled	是
省略	Enabled	是
Disabled	Disabled	否
省略	Disabled	否
省略	省略	否

1. 使用 **secureBoot** 字段。
2. 使用 **virtualizedTrustedPlatformModule** 字段。

有关相关特性和功能的更多信息，请参阅 Microsoft Azure 文档中有关 Azure 虚拟机的受信任的启动 文档。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 编辑 **providerSpec** 字段下的以下部分以提供有效的配置：

   启用 UEFI 安全引导和 vTPM 的有效配置示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          securityProfile:
            settings:
              securityType: TrustedLaunch (1)
              trustedLaunch:
                uefiSettings: (2)
                  secureBoot: Enabled (3)
                  virtualizedTrustedPlatformModule: Enabled (4)
# ...
```

1. 为 Azure 虚拟机启用可信启动。所有有效配置都需要这个值。
2. 指定要使用的 UEFI 安全功能。所有有效配置都需要这个部分。
3. 启用 UEFI 安全引导。
4. 启用使用 vTPM。

验证

- 在 Azure 门户中，查看机器集部署的机器的详情，并验证可信启动选项是否与您配置的值匹配。

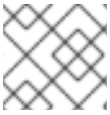
12.5.2.2.7. 使用机器集配置 Azure 机密虚拟机

重要

使用 Azure 机密虚拟机只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

OpenShift Container Platform 4.16 支持 Azure 机密虚拟机 (VM)。



注意

64 位 ARM 架构目前不支持机密虚拟机。

通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的机密虚拟机选项。例如，您可以将这些机器配置为使用 UEFI 安全功能，如安全引导或专用虚拟信任平台模块 (vTPM) 实例。

警告

并非所有实例类型都支持机密虚拟机。不要更改配置为使用机密虚拟机的 control plane 机器集的实例类型，使其不兼容的类型。使用不兼容的实例类型可能会导致集群不稳定。

有关相关特性和功能的更多信息，请参阅 Microsoft Azure 文档中有关机密虚拟机的信息。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 在 **providerSpec** 字段中编辑以下部分：

配置示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          osDisk:
            # ...
            managedDisk:
              securityProfile: 1
                securityEncryptionType: VMGuestStateOnly 2
            # ...
          securityProfile: 3
            settings:
              securityType: ConfidentialVM 4
              confidentialVM:
                uefiSettings: 5
                  secureBoot: Disabled 6
                  virtualizedTrustedPlatformModule: Enabled 7
          vmSize: Standard_DC16ads_v5 8
# ...
```

1. 在使用机密虚拟机时，指定受管磁盘的安全配置集设置。
2. 启用 Azure VM Guest State (VMGS) blob 加密。此设置需要使用 vTPM。
3. 指定机密虚拟机的安全配置集设置。
4. 启用使用机密虚拟机。所有有效配置都需要这个值。
5. 指定要使用的 UEFI 安全功能。所有有效配置都需要这个部分。

6 禁用 UEFI 安全引导。

7 启用使用 vTPM。

8 指定支持机密虚拟机的实例类型。

验证

- 在 Azure 门户中，查看机器集部署的机器的详情，并验证机密虚拟机选项是否与您配置的值匹配。

12.5.2.2.8. Microsoft Azure 虚拟机的加速网络

加速网络使用单一根 I/O 虚拟化(SR-IOV)为 Microsoft Azure 虚拟机提供更直接的路径到交换机。这提高了网络性能。这个功能可以在安装后启用。

12.5.2.2.8.1. 限制

在决定是否使用加速网络时，请考虑以下限制：

- 只有在 Machine API 操作的集群中支持加速网络。
- 加速网络需要一个 Azure 虚拟机，其大小需要可以包括最少四个 vCPU。为了满足此要求，您可以在机器集中更改 **vmSize** 的值。有关 Azure VM 大小的信息，请参阅 Microsoft Azure 文档。

12.5.2.2.8.2. 在现有 Microsoft Azure 集群上启用加速网络

您可以通过在机器集 YAML 文件中添加 **acceleratedNetworking**，在 Azure 上启用加速网络。

先决条件

- 有一个现有的 Microsoft Azure 集群，其中的 Machine API 正常运行。

流程

- 在 **providerSpec** 字段中添加以下内容：

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

1 此行启用加速网络。

2 指定包含至少四个 vCPU 的 Azure VM 大小。有关 VM 大小的信息，请参阅 Microsoft Azure 文档。

验证

- 在 Microsoft Azure 门户上，查看机器集调配的机器的 **Networking** 设置页面，并验证 **Accelerated networking** 字段设置为 **Enabled**。

12.5.3. Google Cloud Platform 的 control plane 配置选项

您可以更改 Google Cloud Platform (GCP) control plane 机器的配置，并通过更新 control plane 机器集中的值来启用功能。当您将更新保存到 control plane 机器集时，Control Plane Machine Set Operator 会根据您配置的更新策略 control plane 机器。

12.5.3.1. 用于配置 Google Cloud Platform 集群的 YAML 示例

以下 YAML 片断示例显示 GCP 集群的供应商规格和故障域配置。

12.5.3.1.1. GCP 供应商规格示例

当您为现有集群创建 control plane 机器集时，供应商规格必须与安装程序创建的 control plane 机器自定义资源 (CR) 中的 **providerSpec** 配置匹配。您可以省略 CR 的故障域部分中设置的任何字段。

使用 OpenShift CLI 获取的值
在以下示例中，您可以使用 OpenShift CLI 获取集群的一些值。

基础架构 ID

<cluster_id> 字符串是基础架构 ID，它基于您在置备集群时设定的集群 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
```

镜像路径

<path_to_image> 字符串是用于创建磁盘的镜像的路径。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取镜像的路径：

```
$ oc -n openshift-machine-api \
  -o
jsonpath='{.spec.template.machines_v1beta1_machine_openshift_io.spec.providerSpec.value.disks[
].image}{"\n"}' \
  get ControlPlaneMachineSet/cluster
```

GCP providerSpec 值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1beta1
            canIPForward: false
            credentialsSecret:
              name: gcp-cloud-credentials (1)
            deletionProtection: false
```

```
        disks:
        - autoDelete: true
          boot: true
          image: <path_to_image> 2
          labels: null
          sizeGb: 200
          type: pd-ssd
        kind: GCPMachineProviderSpec 3
        machineType: e2-standard-4
        metadata:
          creationTimestamp: null
        metadataServiceOptions: {}
        networkInterfaces:
        - network: <cluster_id>-network
          subnetwork: <cluster_id>-master-subnet
        projectID: <project_name> 4
        region: <region> 5
        serviceAccounts:
        - email: <cluster_id>-m@<project_name>.iam.gserviceaccount.com
          scopes:
          - https://www.googleapis.com/auth/cloud-platform
        shieldedInstanceConfig: {}
        tags:
        - <cluster_id>-master
        targetPools:
        - <cluster_id>-api
        userDataSecret:
          name: master-user-data 6
        zone: "" 7
```

1. 指定集群的 secret 名称。不要更改这个值。
2. 指定用于创建磁盘的镜像的路径。

   要使用 GCP Marketplace 镜像，请指定要使用的功能：

   - OpenShift Container Platform: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736**
   - OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736**
   - OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736**
3. 指定云供应商平台类型。不要更改这个值。
4. 指定用于集群的 GCP 项目的名称。
5. 指定集群的 GCP 区域。
6. 指定 control plane 用户数据 secret。不要更改这个值。
7.

此参数在故障域中配置，此处显示了一个空值。如果为此参数指定的值与故障域中的值不同，Operator 会使用故障域中的值覆盖它。

12.5.3.1.2. GCP 故障域配置示例

故障域的 control plane 机器集概念与现有 GCP 概念*区（zone）*类似。**ControlPlaneMachineSet** CR 尽可能将 control plane 机器分散到多个故障域中。

在 control plane 机器集中配置 GCP 故障域时，您必须指定要使用的区名称。

GCP 故障域值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        gcp:
        - zone: <gcp_zone_a> 1
        - zone: <gcp_zone_b> 2
        - zone: <gcp_zone_c>
        - zone: <gcp_zone_d>
        platform: GCP 3
# ...
```

1. 为第一个故障域指定 GCP 区。
2. 指定额外的故障域。其它故障域的方式相同。
3. 指定云供应商平台名称。不要更改这个值。

12.5.3.2. 为 control plane 机器启用 Google Cloud Platform 功能

您可以通过更新 control plane 机器集中的值来启用功能。

12.5.3.2.1. 使用机器集配置持久性磁盘类型

您可以通过编辑机器集 YAML 文件，配置机器集在其中部署机器的持久性磁盘类型。

有关持久性磁盘类型、兼容性、区域可用性和限制的更多信息，请参阅 GCP Compute Engine 文档有关持久性磁盘的信息。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 编辑 **providerSpec** 字段中的以下行：

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
            type: pd-ssd (1)
```

(1) control plane 节点必须使用 **pd-ssd** 磁盘类型。

验证

- 在 Google Cloud 控制台中，查看机器集部署的机器的详情，并验证 **Type** 字段是否与配置的磁盘类型匹配。

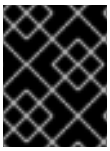
12.5.3.2.2. 使用机器集配置机密虚拟机

通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的机密虚拟机选项。

有关机密虚拟机功能、功能和兼容性的更多信息，请参阅 GCP Compute Engine 文档有关 机密虚拟机 的信息。

注意

64 位 ARM 架构目前不支持机密虚拟机。



重要

OpenShift Container Platform 4.16 不支持一些机密计算功能，如使用 AMD Secure Encrypted Virtualization Secure Nested Paging (SEV-SNP) 的机密虚拟机。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 在 **providerSpec** 字段中编辑以下部分：

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          confidentialCompute: Enabled (1)
          onHostMaintenance: Terminate (2)
          machineType: n2d-standard-8 (3)
...
```

1. 指定是否启用机密虚拟机。有效值为 **Disabled** 或 **Enabled**。
2. 指定主机维护事件期间虚拟机的行为，如硬件或软件更新。对于使用机密虚拟机的机器，此值必须设置为 **Terminate**，这会停止虚拟机。机密虚拟机不支持实时迁移。
3. 指定支持机密虚拟机的机器类型。机密虚拟机支持 N2D 和 C2D 系列机器类型。

验证

- 在 Google Cloud 控制台中，查看机器集部署的机器的详情，并验证 Confidential VM 选项是否与您配置的值匹配。

12.5.3.2.3. 使用机器集配置 Shielded VM 选项

通过编辑机器集 YAML 文件，您可以配置机器集用于部署的机器的 Shielded VM 选项。

有关 Shielded VM 特性和功能的更多信息，请参阅有关 Shielded VM 的 GCP Compute Engine 文档。

流程

1. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。
2. 在 **providerSpec** 字段中编辑以下部分：

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          shieldedInstanceConfig: 1
            integrityMonitoring: Enabled 2
            secureBoot: Disabled 3
            virtualizedTrustedPlatformModule: Enabled 4
# ...
```

1. 在本节中，指定您想要的 Shielded VM 选项。
2. 指定是否启用了完整性监控。有效值为 **Disabled** 或 **Enabled**。

注意

启用完整性监控后，不得禁用虚拟可信平台模块 (vTPM)。

3. 指定是否启用了 UEFI 安全引导。有效值为 **Disabled** 或 **Enabled**。
4. 指定是否启用了 vTPM。有效值为 **Disabled** 或 **Enabled**。

验证

- 使用 Google Cloud 控制台，查看机器集部署的机器的详情，并验证 Shielded VM 选项是否与您配置的值匹配。

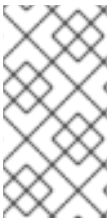
其他资源

- 什么是 Shielded VM?
  - 安全引导
  - 虚拟受信任的平台模块 (vTPM)
  - 完整性监控

12.5.3.2.4. 为机器集启用客户管理的加密密钥

Google Cloud Platform（GCP）Compute Engine 允许用户提供加密密钥来加密磁盘上的数据。密钥用于对数据加密密钥进行加密，而不是加密客户的数据。默认情况下，Compute Engine 使用 Compute Engine 密钥加密这些数据。

您可以使用 Machine API 的集群中使用客户管理的密钥启用加密。您必须首先创建 KMS 密钥并为服务帐户分配正确的权限。需要 KMS 密钥名称、密钥环名称和位置来允许服务帐户使用您的密钥。



注意

如果您不想将专用服务帐户用于 KMS 加密，则使用 Compute Engine 默认服务帐户。如果没有使用专用服务帐户，则必须授予默认服务帐户权限来访问密钥。Compute Engine 默认服务帐户名称遵循 **service-<project_number>@compute-system.iam.gserviceaccount.com** 模式。

流程

1. 要允许特定服务帐户使用 KMS 密钥，并为服务帐户授予正确的 IAM 角色，请使用您的 KMS 密钥名称、密钥环名称和位置运行以下命令：

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

2. 在机器集 YAML 文件中的 **providerSpec** 字段中配置加密密钥。例如：

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
...
spec:
  template:
    spec:
      providerSpec:
        value:
          disks:
          - type:
            encryptionKey:
              kmsKey:
```

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

1. 用于磁盘加密的客户管理的加密密钥名称。
2. KMS 密钥所属的 KMS 密钥环的名称。
3. KMS 密钥环存在的 GCP 位置。
4. 可选：存在 KMS 密钥环的项目 ID。如果没有设置项目 ID，则会使用创建机器设置的机器设置 **projectID**。
5. 可选：用于给定 KMS 密钥加密请求的服务帐户。如果没有设置服务帐户，则使用 Compute Engine 默认服务帐户。

当使用更新的 **providerSpec** 对象配置创建新机器后，磁盘加密密钥就会使用 KMS 密钥加密。

12.5.4. Nutanix 的 control plane 配置选项

您可以通过更新 control plane 机器集中的值来更改 Nutanix control plane 机器的配置。当您将更新保存到 control plane 机器集时，Control Plane Machine Set Operator 会根据您配置的更新策略 control plane 机器。

12.5.4.1. 用于配置 Nutanix 集群的 YAML 示例

以下示例 YAML 片断显示了 Nutanix 集群的供应商规格配置。

12.5.4.1.1. Nutanix 供应商规格示例

当您为现有集群创建 control plane 机器集时，供应商规格必须与安装程序创建的 control plane 机器自定义资源 (CR) 中的 **providerSpec** 配置匹配。

使用 OpenShift CLI 获取的值
在以下示例中，您可以使用 OpenShift CLI 获取集群的一些值。

基础架构 ID

<cluster_id> 字符串是基础架构 ID，它基于您在置备集群时设定的集群 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：

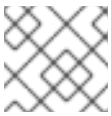
```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
```

Nutanix providerSpec 值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
```

```
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1
            bootType: "" (1)
            categories: (2)
            - key: <category_name>
              value: <category_value>
            cluster: (3)
              type: uuid
              uuid: <cluster_uuid>
            credentialsSecret:
              name: nutanix-credentials (4)
            image: (5)
              name: <cluster_id>-rhcos
              type: name
            kind: NutanixMachineProviderConfig (6)
            memorySize: 16Gi (7)
            metadata:
              creationTimestamp: null
            project: (8)
              type: name
              name: <project_name>
            subnets: (9)
            - type: uuid
              uuid: <subnet_uuid>
            systemDiskSize: 120Gi (10)
            userDataSecret:
              name: master-user-data (11)
            vcpuSockets: 8 (12)
            vcpusPerSocket: 1 (13)
```

(1) 指定 control plane 机器使用的引导类型。有关引导类型的更多信息，请参阅虚拟环境中的了解 UEFI、安全引导和 TPM。有效值为 **Legacy**、**SecureBoot** 或 **UEFI**。默认值为 **Legacy**。

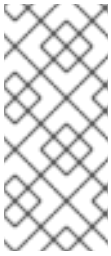


注意

您必须在 OpenShift Container Platform 4.16 中使用 **Legacy** 引导类型。

(2) 指定一个或多个 Nutanix Prism 类别以应用到 control plane 机器。此小节需要 **key** 和 **value** 参数代表存在于 Prism Central 中的类别的键值对。有关类别的更多信息，请参阅类别管理。

(3) 指定 Nutanix Prism Element 集群配置。在本例中，集群类型是 **uuid**，因此有一个 **uuid** 小节。

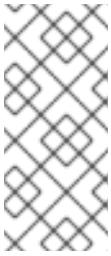


注意

使用 OpenShift Container Platform 版本 4.15 或更高版本的集群可以使用故障域配置。

如果集群被配置为使用故障域，则此参数会在故障域中配置。如果您在使用故障域时在供应商规格中指定这个值，则 Control Plane Machine Set Operator 会忽略它。

4. 指定集群的 secret 名称。不要更改这个值。
5. 指定用于创建磁盘的镜像。
6. 指定云供应商平台类型。不要更改这个值。
7. 指定为 control plane 机器分配的内存。
8. 指定用于集群的 Nutanix 项目。在本例中，项目类型是 **name**，因此有一个 **name** 小节。
9. 指定子网配置。在本例中，子网类型是 **uuid**，因此有一个 **uuid** 小节。



注意

使用 OpenShift Container Platform 版本 4.15 或更高版本的集群可以使用故障域配置。

如果集群被配置为使用故障域，则此参数会在故障域中配置。如果您在使用故障域时在供应商规格中指定这个值，则 Control Plane Machine Set Operator 会忽略它。

10. 指定 control plane 机器的 VM 磁盘大小。
11. 指定 control plane 用户数据 secret。不要更改这个值。
12. 指定为 control plane 机器分配的 vCPU 套接字数量。
13. 指定每个 control plane vCPU 套接字的 vCPU 数量。

12.5.4.1.2. Nutanix 集群的故障域

要在 Nutanix 集群中添加或更新故障域配置，您必须对几个资源进行协调更改。需要以下操作：

1. 修改集群基础架构自定义资源 (CR)。
2. 修改集群 control plane 机器集 CR。
3. 修改或替换计算机器设置 CR。

如需更多信息，请参阅 *安装后配置* 内容中的"将故障域添加到现有 Nutanix 集群"。

其他资源

- 在现有的 Nutanix 集群中添加故障域

12.5.5. Red Hat OpenStack Platform 的 control plane 配置选项

您可以更改 Red Hat OpenStack Platform (RHOSP) control plane 机器的配置，并通过更新 control plane 机器集中的值来启用功能。当您将更新保存到 control plane 机器集时，Control Plane Machine Set Operator 会根据您配置的更新策略 control plane 机器。

12.5.5.1. 用于配置 Red Hat OpenStack Platform (RHOSP) 集群的 YAML 示例

以下示例 YAML 片断显示 RHOSP 集群的供应商规格和故障域配置。

12.5.5.1.1. RHOSP 供应商规格示例

当您为现有集群创建 control plane 机器集时，供应商规格必须与安装程序创建的 control plane 机器自定义资源 (CR) 中的 **providerSpec** 配置匹配。

OpenStack providerSpec 值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1alpha1
            cloudName: openstack
            cloudsSecret:
              name: openstack-cloud-credentials 1
              namespace: openshift-machine-api
            flavor: m1.xlarge 2
            image: ocp1-2g2xs-rhcos
            kind: OpenstackProviderSpec 3
            metadata:
              creationTimestamp: null
            networks:
            - filter: {}
              subnets:
              - filter:
                  name: ocp1-2g2xs-nodes
                  tags: openshiftClusterID=ocp1-2g2xs
            securityGroups:
            - filter: {}
              name: ocp1-2g2xs-master 4
            serverGroupName: ocp1-2g2xs-master
            serverMetadata:
              Name: ocp1-2g2xs-master
              openshiftClusterID: ocp1-2g2xs
            tags:
            - openshiftClusterID=ocp1-2g2xs
            trunk: true
            userDataSecret:
              name: master-user-data
```

-

1. 集群的 secret 名称。不要更改这个值。
2. control plane 的 RHOSP 类别类型。
3. RHOSP 云供应商平台类型。不要更改这个值。
4. control plane 机器安全组。

12.5.5.1.2. RHOSP 故障域配置示例

故障域的 control plane 机器集概念与现有 可用区（availability zone） 的 Red Hat OpenStack Platform (RHOSP) 概念类似。**ControlPlaneMachineSet** CR 尽可能将 control plane 机器分散到多个故障域中。

以下示例演示了使用多个 Nova 可用性区域和 Cinder 可用性区域。

OpenStack 故障域值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains:
        platform: OpenStack
        openstack:
        - availabilityZone: nova-az0
          rootVolume:
            availabilityZone: cinder-az0
        - availabilityZone: nova-az1
          rootVolume:
            availabilityZone: cinder-az1
        - availabilityZone: nova-az2
          rootVolume:
            availabilityZone: cinder-az2
# ...
```

12.5.5.2. 为 control plane 机器启用 Red Hat OpenStack Platform (RHOSP) 功能

您可以通过更新 control plane 机器集中的值来启用功能。

12.5.5.2.1. 使用 control plane 机器集更改 RHOSP 计算类型

您可以通过更新 control plane 机器集自定义资源中的规格来更改 control plane 机器使用的 Red Hat OpenStack Platform (RHOSP)计算服务(Nova)类别。

在 RHOSP 中，类别定义计算实例的计算、内存和存储容量。通过增加或减少类别大小，您可以垂直扩展 control plane。

先决条件

- 您的 RHOSP 集群使用 control plane 机器集。

流程

1. 编辑 **providerSpec** 字段中的以下行：

```
providerSpec:
  value:
# ...
    flavor: m1.xlarge (1)
```

(1) 指定与现有选择相同的 RHOSP 类别类型。例如，您可以将 **m6i.xlarge** 更改为 **m6i.2xlarge** 或 **m6i.4xlarge**。您可以根据垂直扩展需求选择大或较小的类别。

2. 保存您的更改。

保存更改后，计算机将被替换为您选择的类别。

12.5.6. VMware vSphere 的 control plane 配置选项

您可以通过更新 control plane 机器集中的值来更改 VMware vSphere control plane 机器的配置。当您将更新保存到 control plane 机器集时，Control Plane Machine Set Operator 会根据您配置的更新策略 control plane 机器。

12.5.6.1. 用于配置 VMware vSphere 集群的 YAML 示例

以下示例 YAML 片断显示 vSphere 集群的供应商规格和故障域配置。

12.5.6.1.1. VMware vSphere 供应商规格示例

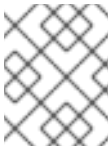
当您为现有集群创建 control plane 机器集时，供应商规格必须与安装程序创建的 control plane 机器自定义资源 (CR) 中的 **providerSpec** 配置匹配。

vSphere providerSpec 值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
      spec:
        providerSpec:
          value:
            apiVersion: machine.openshift.io/v1beta1
            credentialsSecret:
              name: vsphere-cloud-credentials (1)
            diskGiB: 120 (2)
```

```
    kind: VSphereMachineProviderSpec 3
    memoryMiB: 16384 4
    metadata:
      creationTimestamp: null
    network: 5
      devices:
      - networkName: <vm_network_name>
    numCPUs: 4 6
    numCoresPerSocket: 4 7
    snapshot: ""
    template: <vm_template_name> 8
    userDataSecret:
      name: master-user-data 9
    workspace: 10
      datacenter: <vcenter_datacenter_name> 11
      datastore: <vcenter_datastore_name> 12
      folder: <path_to_vcenter_vm_folder> 13
      resourcePool: <vsphere_resource_pool> 14
      server: <vcenter_server_ip> 15
```

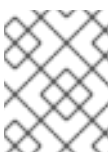
1. 指定集群的 secret 名称。不要更改这个值。
2. 指定 control plane 机器的 VM 磁盘大小。
3. 指定云供应商平台类型。不要更改这个值。
4. 指定为 control plane 机器分配的内存。
5. 指定在其上部署 control plane 的网络。



注意

如果集群被配置为使用故障域，则此参数会在故障域中配置。如果您在使用故障域时在供应商规格中指定这个值，则 Control Plane Machine Set Operator 会忽略它。

6. 指定为 control plane 机器分配的 CPU 数量。
7. 指定每个 control plane CPU 的内核数。
8. 指定要使用的 vSphere 虚拟机模板，如 **user-5ddjd-rhcos**。
9. 指定 control plane 用户数据 secret。不要更改这个值。
10. 指定 control plane 的工作区详情。



注意

如果集群被配置为使用故障域，则这些参数会在故障域中配置。如果您在使用故障域时在供应商规格中指定这个值，则 Control Plane Machine Set Operator 会忽略它。

11. 指定 control plane 的 vCenter Datacenter。
12. 为 control plane 指定 vCenter Datastore。

13 指定 vCenter 中 vSphere 虚拟机文件夹的路径，如 **/dc1/vm/user-inst-5ddjd**。

14 指定虚拟机的 vSphere 资源池。

15 指定 vCenter 服务器 IP 或完全限定域名。

12.5.6.1.2. VMware vSphere 故障域配置示例

在 VMware vSphere 基础架构上，集群范围的基础架构自定义资源定义(CRD) **infrastructures.config.openshift.io** 定义集群的故障域。**ControlPlaneMachineSet** 自定义资源 (CR) 中的 **providerSpec** 指定 control plane 机器集使用的故障域的名称，以确保 control plane 节点部署到适当的故障域中。故障域是由 control plane 机器集、vCenter 数据中心、vCenter 数据存储和网络组成的基础架构资源。

通过使用故障域资源，您可以使用 control plane 机器集在单独的集群或数据中心中部署 control plane 机器。control plane 机器集还在定义的故障域间平衡 control plane 机器，以便为基础架构提供容错功能。

注意

如果您修改了 **ControlPlaneMachineSet** CR 中的 **ProviderSpec** 配置，control plane 机器集会更新在主基础架构上部署的所有 control plane 机器和每个故障域基础架构。

VMware vSphere 故障域值示例

```
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
metadata:
  name: cluster
  namespace: openshift-machine-api
spec:
# ...
  template:
# ...
    machines_v1beta1_machine_openshift_io:
      failureDomains: 1
        platform: VSphere
        vsphere: 2
        - name: <failure_domain_name1>
        - name: <failure_domain_name2>
# ...
```

1 指定 OpenShift Container Platform 集群节点的 vCenter 位置。

2 根据 control plane 机器集的名称指定故障域。

重要

本节中的每个 **name** 字段值必须与集群范围的基础架构 CRD 的 **failureDomains.name** 字段中对应的值匹配。您可以运行以下命令来找到 **failureDomains.name** 字段的值：

```
$ oc get infrastructure cluster -o=jsonpath=
{.spec.platformSpec.vsphere.failureDomains[0].name
```

name 字段是您可以在 **ControlPlaneMachineSet** CR 中指定的唯一支持的故障域字段。

有关为每个故障域定义资源的集群范围基础架构 CRD 的示例，请参阅"在 vSphere 上为集群指定多个区域和区域"

其他资源

- 在 vSphere 上为集群指定多个区域和区域

12.5.6.2. 为 control plane 机器启用 VMware vSphere 功能

您可以通过更新 control plane 机器集中的值来启用功能。

12.5.6.2.1. 使用机器集向机器添加标签

OpenShift Container Platform 为其创建的每个虚拟机 (VM) 添加特定于集群的标签。安装程序使用这些标签在卸载集群时选择要删除的虚拟机。

除了分配给虚拟机的特定于集群的标签外，您还可以将机器集配置为向它置备的虚拟机中添加最多 10 个额外的 vSphere 标签。

先决条件

- 您可以使用具有 **cluster-admin** 权限的账户访问在 vSphere 上安装的 OpenShift Container Platform 集群。
- 您可以访问与集群关联的 VMware vCenter 控制台。
- 您已在 vCenter 控制台中创建了标签。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 使用 vCenter 控制台查找您要添加到机器的任何标签的标签 ID：
   a. 登录到 vCenter 控制台。
   b. 在 **Home** 菜单中，点 **Tags & Custom Attributes**。
   c. 选择您要添加到机器的标签。
   d. 使用您选择的标签的浏览器 URL 来识别标签 ID。

标签 **URL 示例**

```
https://vcenter.example.com/ui/app/tags/tag/urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL/permissions
```

标签 **ID 示例**

```
urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL
```

2. 在文本编辑器中，为现有机器集打开 YAML 文件或创建新机器。

3. 编辑 **providerSpec** 字段中的以下行：

```yaml
apiVersion: machine.openshift.io/v1
kind: ControlPlaneMachineSet
# ...
spec:
  template:
    spec:
      providerSpec:
        value:
          tagIDs: 1
          - <tag_id_value> 2
# ...
```

1. 指定要添加到此机器集置备的机器上最多 10 个标签列表。
2. 指定要添加到机器的标签值。例如，**urn:vmomi:InventoryServiceTag:208e713c-cae3-4b7f-918e-4051ca7d1f97:GLOBAL**.

12.6. CONTROL PLANE 弹性和恢复

您可以使用 control plane 机器集提高 OpenShift Container Platform 集群的 control plane 的弹性。

12.6.1. 使用故障域的高可用性和容错功能

在可能的情况下，control plane 机器集会将 control plane 机器分散到多个故障域中。此配置在 control plane 中提供高可用性和容错功能。当基础架构供应商出现问题时，此策略可以帮助保护 control plane。

12.6.1.1. 故障域平台支持和配置

故障域的 control plane 机器集概念与云供应商的现有概念类似。并非所有平台都支持使用故障域。

表 12.3. 故障域支持列表

云供应商	支持故障域	Provider nomenclature
Amazon Web Services (AWS)	X	可用区域 (AZ)
Google Cloud Platform (GCP)	X	zone

云供应商	支持故障域	Provider nomenclature
Microsoft Azure	X	Azure 可用区
Nutanix	X	故障域
Red Hat OpenStack Platform(RHOSP)	X	OpenStack Nova 可用区 和 OpenStack Cinder 可用区
VMware vSphere	X	映射到 vSphere 区的故障域 [1]

1. 如需更多信息，请参阅"VMware vCenter 的区域和区域"。

control plane 机器集自定义资源 (CR) 中的故障域配置是特定于平台的。有关 CR 中故障域参数的更多信息，请参阅您的供应商故障域配置示例。

其他资源

- Amazon Web Services 故障域配置示例
- Google Cloud Platform 故障域配置示例
- Microsoft Azure 故障域配置示例
- 在现有的 Nutanix 集群中添加故障域
- Red Hat OpenStack Platform (RHOSP) 故障域配置示例
- VMware vSphere 故障域配置示例
- VMware vCenter 的区域和区域

12.6.1.2. 平衡 control plane 机器

control plane 机器集在自定义资源 (CR) 中指定的故障域间平衡 control plane 机器。

在可能的情况下，control plane 机器集会平等地使用每个故障域来确保适当的容错能力。如果故障域的数量小于 control plane 机器，则会按照其名称的字母顺序选择故障域来重复使用。对于没有指定故障域的集群，所有 control plane 机器都放在单个故障域中。

故障域配置的一些更改会导致 control plane 机器集重新平衡 control plane 机器。例如，如果您向数量小于 control plane 机器数量的故障域的集群中添加故障域，control plane 机器集会在所有可用故障域间重新平衡机器。

12.6.2. 恢复失败的 control plane 机器

Control Plane Machine Set Operator 会自动恢复 control plane 机器。删除 control plane 机器时，Operator 会使用 **ControlPlaneMachineSet** 自定义资源(CR) 中指定的配置创建替换。

对于使用 control plane 机器集的集群，您可以配置机器健康检查。机器健康检查会删除不健康的 control plane 机器，以便替换它们。

重要

如果您为 control plane 配置 **MachineHealthCheck** 资源，请将 **maxUnhealthy** 的值设置为 **1**。

此配置可确保当多个 control plane 机器显示为不健康时，机器健康检查不会采取任何操作。多个不健康的 control plane 机器可能会表示 etcd 集群已降级或扩展操作来替换失败的机器。

如果 etcd 集群降级，可能需要手动干预。如果扩展操作正在进行，机器健康检查应该允许它完成。

其他资源

- 部署机器健康检查

12.6.3. 使用机器生命周期 hook 进行仲裁保护

对于使用 Machine API Operator 的 OpenShift Container Platform 集群，etcd Operator 使用机器删除阶段的生命周期 hook 来实现仲裁保护机制。

通过使用 **preDrain** 生命周期 hook，etcd Operator 可以控制 control plane 机器上的 pod 排空和删除的时间。为了保护 etcd 仲裁，etcd Operator 会阻止删除 etcd 成员，直到该成员迁移到集群中的新节点。

此机制允许 etcd Operator 对 etcd 仲裁的成员进行精确控制，并允许 Machine API Operator 在不需要 etcd 集群的特定操作了解的情况下安全地创建和删除 control plane 机器。

12.6.3.1. 使用仲裁保护处理顺序删除 control plane

当在使用 control plane 机器集的集群中替换 control plane 机器时，集群会临时有四个 control plane 机器。当第四个 control plane 节点加入集群时，etcd Operator 会在替换节点上启动新的 etcd 成员。当 etcd Operator 观察到旧的 control plane 机器已被标记为删除时，它会停止旧节点上的 etcd 成员，并提升替换 etcd 成员以加入集群的仲裁。

control plane 机器 **Deleting** 阶段按以下顺序进行：

1. control plane 机器会停止以进行删除。
2. control plane 机器进入 **Deleting** 阶段。
3. 为了满足 **preDrain** 生命周期 hook，etcd Operator 会执行以下操作：
   a. etcd Operator 等待第四个 control plane 机器作为 etcd 成员添加到集群中。这个新 etcd 成员的状态为 **Running** 而不是 **ready**，直到它从 etcd leader 接收到了完整的数据库更新。
   b. 当新 etcd 成员收到完整数据库更新时，etcd Operator 会将新的 etcd 成员提升到投票成员，并从集群中移除旧的 etcd 成员。

   完成此转换后，旧的 etcd pod 及其数据是安全的，因此会删除 **preDrain** 生命周期 hook。
4. control plane 机器状态条件 **Drainable** 设置为 **True**。
5. 机器控制器尝试排空由 control plane 机器支持的节点。
   - 如果排空失败，**Drained** 被设置为 **False**，机器控制器会尝试再次排空该节点。
   - 如果排空成功，**Drained** 被设置为 **True**。

6. control plane 机器状态条件 **Drained** 设置为 **True**。
7. 如果没有其他 Operator 添加了 **preTerminate** 生命周期 hook，control plane 机器状态条件 **Terminable** 被设置为 **True**。
8. 机器控制器从基础架构供应商中删除实例。
9. 机器控制器会删除 **Node** 对象。

YAML 片断演示 etcd 仲裁保护 preDrain 生命周期 hook

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
spec:
  lifecycleHooks:
    preDrain:
    - name: EtcdQuorumOperator (1)
      owner: clusteroperator/etcd (2)
  ...
```

(1) **preDrain** 生命周期 hook 的名称。

(2) 管理 **preDrain** 生命周期 hook 的 hook 实施控制器。

其他资源

- 机器删除阶段的生命周期 hook

12.7. CONTROL PLANE 机器集故障排除

使用本节中的信息从您可能会遇到的问题了解和恢复。

12.7.1. 检查 control plane 机器设置自定义资源状态

您可以验证 **ControlPlaneMachineSet** 自定义资源 (CR) 是否存在以及其状态。

流程

- 运行以下命令确定 CR 的状态：

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
  --namespace openshift-machine-api
```

  - **Active** 的结果表示 **ControlPlaneMachineSet** CR 存在并被激活。不需要管理员操作。
  - **Inactive** 表示 **ControlPlaneMachineSet** CR 存在但没有激活。
  - **NotFound** 表示没有现有的 **ControlPlaneMachineSet** CR。

后续步骤

要使用 control plane 机器集，您必须确保集群有正确设置的 **ControlPlaneMachineSet** CR。

- 如果您的集群有一个现有的 CR，您必须验证 CR 中的配置是否正确。
- 如果集群没有现有的 CR，则必须为集群创建一个带有正确配置的 CR。

其他资源

- 激活 control plane 机器集自定义资源
- 创建 control plane 机器集自定义资源

12.7.2. 添加缺少的 Azure 内部负载均衡器

Azure 的 **ControlPlaneMachineSet** 和 control plane **Machine** 自定义资源(CR) 都需要 **internalLoadBalancer** 参数。如果集群上没有预配置此参数，则必须将其添加到两个 CR 中。

有关此参数位于 Azure 供应商规格中的更多信息，请参阅 Azure 供应商规格示例。control plane **Machine** CR 中的放置类似。

流程

1. 运行以下命令列出集群中的 control plane 机器：

```
$ oc get machines \
  -l machine.openshift.io/cluster-api-machine-role==master \
  -n openshift-machine-api
```

2. 对于每个 control plane 机器，运行以下命令编辑 CR：

```
$ oc edit machine <control_plane_machine_name>
```

3. 添加 **internalLoadBalancer** 参数，使其包含您的集群的正确详情，并保存您的更改。
4. 运行以下命令来编辑 control plane 机器集 CR：

```
$ oc edit controlplanemachineset.machine.openshift.io cluster \
  -n openshift-machine-api
```

5. 添加 **internalLoadBalancer** 参数，使其包含您的集群的正确详情，并保存您的更改。

后续步骤

- 对于使用默认 **RollingUpdate** 更新策略的集群，Operator 会自动将更改传播到 control plane 配置。
- 对于配置为使用 **OnDelete** 更新策略的集群，您必须手动替换 control plane 机器。

其他资源

- Microsoft Azure 供应商规格示例

12.7.3. 恢复降级的 etcd Operator

在某些情况下可能会导致 etcd Operator 降级。

例如，在执行补救时，机器健康检查可能会删除托管 etcd 的 control plane 机器。如果此时无法访问 etcd 成员，etcd Operator 会降级。

当 etcd Operator 降级时，需要人工干预才能强制 Operator 删除失败的成员并恢复集群状态。

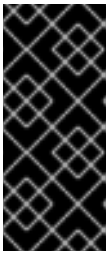
流程

1. 运行以下命令列出集群中的 control plane 机器：

```
$ oc get machines \
  -l machine.openshift.io/cluster-api-machine-role==master \
  -n openshift-machine-api \
  -o wide
```

   以下任何条件都可能表示 control plane 机器失败：

   - **STATE** 值为 **stopped**。
   - **PHASE** 值是 **Failed**。
   - **PHASE** 值为 **Deleting** 超过十分钟。



重要

在继续操作前，请确保集群有两个健康的 control plane 机器。对多个 control plane 机器执行操作可能会丢失 etcd 仲裁，并可能导致数据丢失。

如果您丢失了大多数 control plane 主机，并导致 etcd 仲裁丢失，那么您必须遵循灾难恢复流程 "Restoring to a previous cluster state" 而不是此过程。

2. 运行以下命令，编辑失败的 control plane 机器的机器 CR：

```
$ oc edit machine <control_plane_machine_name>
```

3. 从失败的 control plane 机器中删除 **lifecycleHooks** 参数的内容并保存您的更改。etcd Operator 从集群中移除失败的机器，然后可以安全地添加新的 etcd 成员。

其他资源

- 恢复到一个以前的集群状态

12.7.4. 升级在 RHOSP 上运行的集群

对于使用 OpenShift Container Platform 4.13 或更早版本创建的 Red Hat OpenStack Platform (RHOSP) 运行的集群，您可能需要在使用 control plane 机器集前执行升级后的任务。

12.7.4.1. 升级后配置具有根卷可用区的 RHOSP 集群

对于在升级的 Red Hat OpenStack Platform (RHOSP) 上运行的一些集群，如果以下配置为 true，则必须手动更新机器资源：

- 升级的集群使用 OpenShift Container Platform 4.13 或更早版本创建。

- 集群基础架构是安装程序置备的基础架构 (installer-provisioned)。
- 机器在多个可用区间分布。
- 机器被配置为使用没有定义块存储可用区的 root 卷。

要了解为什么需要这个过程，请参阅解决方案 #7024383。

流程

1. 对于所有 control plane 机器，编辑与环境匹配的所有 control plane 机器的 provider spec。例如，要编辑 **master-0** 机器，请输入以下命令：

```
$ oc edit machine/<cluster_id>-master-0 -n openshift-machine-api
```

其中：

<cluster_id>
指定升级集群的 ID。

2. 在供应商 spec 中，将属性 **rootVolume.availabilityZone** 的值设置为您要使用的可用区的卷。

RHOSP 供应商规格示例

```
providerSpec:
  value:
    apiVersion: machine.openshift.io/v1alpha1
    availabilityZone: az0
      cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: m1.xlarge
    image: rhcos-4.14
    kind: OpenstackProviderSpec
    metadata:
      creationTimestamp: null
    networks:
    - filter: {}
      subnets:
      - filter:
          name: refarch-lv7q9-nodes
          tags: openshiftClusterID=refarch-lv7q9
    rootVolume:
        availabilityZone: nova (1)
        diskSize: 30
        sourceUUID: rhcos-4.12
        volumeType: fast-0
    securityGroups:
    - filter: {}
      name: refarch-lv7q9-master
    serverGroupName: refarch-lv7q9-master
    serverMetadata:
      Name: refarch-lv7q9-master
      openshiftClusterID: refarch-lv7q9
```

```
    tags:
    - openshiftClusterID=refarch-lv7q9
    trunk: true
    userDataSecret:
      name: master-user-data
```

1 将区域名称设置为这个值。

注意

如果在初始集群部署后编辑或重新创建了机器资源，您可能需要为配置调整这些步骤。

在 RHOSP 集群中，为您的机器找到根卷的可用区，并将其用作值。

3. 运行以下命令来检索 control plane 机器集资源的信息：

```
$ oc describe controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

4. 运行以下命令编辑资源：

```
$ oc edit controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

5. 对于该资源，将 **spec.state** 属性的值设置为 **Active** 以激活集群的 control plane 机器集。

您的 control plane 已准备好由 Cluster Control Plane Machine Set Operator 管理。

12.7.4.2. 升级后配置带有可用区的 control plane 机器的 RHOSP 集群

对于在升级的 Red Hat OpenStack Platform (RHOSP) 上运行的一些集群，如果以下配置为 true，则必须手动更新机器资源：

- 升级的集群使用 OpenShift Container Platform 4.13 或更早版本创建。
- 集群基础架构是安装程序置备的基础架构 (installer-provisioned)。
- control plane 机器在多个计算可用区间分布。

要了解为什么需要这个过程，请参阅解决方案 #7013893。

流程

1. 对于 **master-1** 和 **master-2** control plane 机器，打开供应商规格进行编辑。例如，要编辑第一个机器，请输入以下命令：

```
$ oc edit machine/<cluster_id>-master-1 -n openshift-machine-api
```

其中：

<cluster_id>
指定升级集群的 ID。

2. 对于 **master-1** 和 **master-2** control plane 机器，编辑其供应商 specs 中的 **serverGroupName** 属性的值，以匹配机器 **master-0**。

   RHOSP 供应商规格示例

```
providerSpec:
  value:
    apiVersion: machine.openshift.io/v1alpha1
    availabilityZone: az0
      cloudName: openstack
    cloudsSecret:
      name: openstack-cloud-credentials
      namespace: openshift-machine-api
    flavor: m1.xlarge
    image: rhcos-4.16
    kind: OpenstackProviderSpec
    metadata:
      creationTimestamp: null
    networks:
    - filter: {}
      subnets:
      - filter:
          name: refarch-lv7q9-nodes
          tags: openshiftClusterID=refarch-lv7q9
    securityGroups:
    - filter: {}
      name: refarch-lv7q9-master
    serverGroupName: refarch-lv7q9-master-az0 (1)
    serverMetadata:
      Name: refarch-lv7q9-master
      openshiftClusterID: refarch-lv7q9
    tags:
    - openshiftClusterID=refarch-lv7q9
    trunk: true
    userDataSecret:
      name: master-user-data
```

(1) 这个值必须与 **master-0**、**master-1** 和 **master-3** 的机器匹配。

注意

如果在初始集群部署后编辑或重新创建了机器资源，您可能需要为配置调整这些步骤。

在 RHOSP 集群中，找到 control plane 实例所在的服务器组，并将其用作值。

3. 运行以下命令来检索 control plane 机器集资源的信息：

```
$ oc describe controlplanemachineset.machine.openshift.io/cluster --namespace openshift-machine-api
```

4. 运行以下命令编辑资源：

```
$ oc edit controlplanemachineset.machine.openshift.io/cluster --namespace openshift-
machine-api
```

5. 对于该资源，将 **spec.state** 属性的值设置为 **Active** 以激活集群的 control plane 机器集。

您的 control plane 已准备好由 Cluster Control Plane Machine Set Operator 管理。

12.8. 禁用 CONTROL PLANE 机器集

激活的 **ControlPlaneMachineSet** 自定义资源 (CR) 中的 **.spec.state** 字段无法从 **Active** 改为 **Inactive**。要禁用 control plane 机器集，您必须删除 CR，以便从集群中删除。

删除 CR 时，Control Plane Machine Set Operator 会执行清理操作并禁用 control plane 机器集。然后，Operator 会从集群中删除 CR，并使用默认设置创建一个不活跃的 control plane 机器集。

12.8.1. 删除 control plane 机器集

要停止使用集群中设置了 control plane 机器集管理 control plane 机器，您必须删除 **ControlPlaneMachineSet** 自定义资源 (CR)。

流程

- 运行以下命令来删除 control plane 机器集 CR：

```
$ oc delete controlplanemachineset.machine.openshift.io cluster \
  -n openshift-machine-api
```

验证

- 检查 control plane 机器设置自定义资源状态。**Inactive** 表示删除和替换过程成功。**ControlPlaneMachineSet** CR 存在但没有激活。

12.8.2. 检查 control plane 机器设置自定义资源状态

您可以验证 **ControlPlaneMachineSet** 自定义资源 (CR) 是否存在以及其状态。

流程

- 运行以下命令确定 CR 的状态：

```
$ oc get controlplanemachineset.machine.openshift.io cluster \
  --namespace openshift-machine-api
```

  - **Active** 的结果表示 **ControlPlaneMachineSet** CR 存在并被激活。不需要管理员操作。
  - **Inactive** 表示 **ControlPlaneMachineSet** CR 存在但没有激活。
  - **NotFound** 表示没有现有的 **ControlPlaneMachineSet** CR。

12.8.3. 重新启用 control plane 机器集

要重新启用 control plane 机器集，您必须确保 CR 中的配置正确，用于集群并激活它。

其他资源

- 激活 control plane 机器集自定义资源

第 13 章 使用 CLUSTER API 管理机器

13.1. 关于集群 API

重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

Cluster API 是一个上游项目，被集成到 OpenShift Container Platform 中，作为 Amazon Web Services (AWS)、Google Cloud Platform (GCP) 和 VMware vSphere 的技术预览。

13.1.1. 集群 API 概述

您可以使用 Cluster API 来创建和管理 OpenShift Container Platform 集群中的计算机器和计算机器集。这个功能是使用 Machine API 管理机器的补充或替代功能。

对于 OpenShift Container Platform 4.16 集群，您可以使用 Cluster API 在集群安装完成后执行节点主机置备管理操作。该系统在公有或私有云基础架构之上启用弹性动态置备方法。

使用 Cluster API 技术预览，您可以为支持的供应商在 OpenShift Container Platform 集群中创建计算机器和计算机器集。您还可以探索此实施可能不能使用 Machine API 启用的功能。

13.1.1.1. 集群 API 的优点

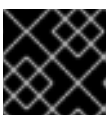
通过使用集群 API，OpenShift Container Platform 用户和开发人员能够实现以下优点：

- 使用上游社区 Cluster API 基础架构提供程序的选项可能不受 Machine API 支持。
- 为基础架构供应商维护机器控制器的第三方合作机会。
- 在 OpenShift Container Platform 中，使用同一组 Kubernetes 工具进行基础架构管理的能力。
- 使用支持 Machine API 不提供的功能的集群 API 创建计算机器集的功能。

13.1.1.2. 集群 API 限制

使用集群 API 管理机器是一个技术预览功能，有以下限制：

- 要使用这个功能，您必须启用 **TechPreviewNoUpgrade** 功能集。

  重要

  启用此功能集无法撤消并阻止次版本更新。

- 只有 Amazon Web Services (AWS)、Google Cloud Platform (GCP) 和 VMware vSphere 集群可以使用 Cluster API。
- 您必须手动创建 Cluster API 需要的主要资源。如需更多信息，请参阅"开始使用集群 API"。

- 您不能使用 Cluster API 管理 control plane 机器。
- 不支持将 Machine API 创建的现有计算机器集迁移到 Cluster API 计算机器集。
- 使用 Machine API 的全功能奇偶校验不可用。
- 对于使用 Cluster API 的集群，OpenShift CLI (**oc**) 命令会优先选择 Cluster API（与 Machine API 对象相比）。此行为会影响任何对集群 API 和 Machine API 中代表的对象执行操作的 **oc** 命令。有关此问题的更多信息和临时解决方案，请参阅故障排除内容中的"使用 CLI 时引用预期的对象"。

其他资源

- 使用功能门启用功能
- Cluster API 入门
- 使用 CLI 引用预期的对象

13.1.2. 集群 API 架构

OpenShift Container Platform 与上游 Cluster API 集成由 Cluster CAPI Operator 实现和管理。Cluster CAPI Operator 及其操作对象在 **openshift-cluster-api** 命名空间中置备，这与使用 **openshift-machine-api** 命名空间中的 Machine API 相反。

13.1.2.1. Cluster CAPI Operator

Cluster CAPI Operator 是一个 OpenShift Container Platform Operator，维护 Cluster API 资源的生命周期。此 Operator 负责在 OpenShift Container Platform 集群中部署 Cluster API 项目的所有管理任务。

如果正确配置了集群以允许使用 Cluster API，Cluster CAPI Operator 会在集群中安装 Cluster API 组件。

如需更多信息，请参阅 *Cluster Operator 参考*内容中的 Cluster CAPI Operator 条目。

其他资源

- Cluster CAPI Operator

13.1.2.2. Cluster API 主资源

集群 API 由以下主要资源组成：对于这个功能的技术预览，您必须在 **openshift-cluster-api** 命名空间中手动创建这些资源。

集群

代表由 Cluster API 管理的集群的基本单元。

基础架构

特定于供应商的资源，用于定义集群中所有计算机器集（如地区和子网）共享的属性。

机器模板

特定于提供程序的模板，用于定义计算机器组创建的机器的属性。

机器集

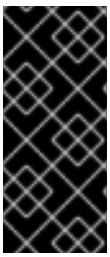
一组机器。
计算机器集适用于机器，副本集则适用于 pod。要添加机器或缩减机器，请更改计算机器设置自定义资源上的 **replicas** 字段来满足您的计算需求。

使用 Cluster API 时，计算机器集引用 **Cluster** 对象和特定于供应商的机器模板。

机器

描述节点主机的基本单元。
Cluster API 根据机器模板中的配置创建机器。

13.2. CLUSTER API 入门



重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

对于 Cluster API 技术预览，您必须创建 Cluster API 需要的主要资源。

13.2.1. 创建 Cluster API 主资源

要创建 Cluster API 主资源，您必须获取集群 ID 值，用于集群资源清单中的 **<cluster_name>** 参数。

13.2.1.1. 获取集群 ID 值

您可以使用 OpenShift CLI (**oc**) 来查找集群 ID 值。

先决条件

- 您已部署了 OpenShift Container Platform 集群。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。

流程

- 运行以下命令来获取集群 ID 的值：

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

您可以通过创建 YAML 清单文件并使用 OpenShift CLI (**oc**) 应用它们来手动创建 Cluster API 主资源。

13.2.1.2. 创建 Cluster API 集群资源

您可以通过创建 YAML 清单文件并使用 OpenShift CLI (**oc**) 应用来创建集群资源。

先决条件

- 您已部署了 OpenShift Container Platform 集群。

- 您已启用了 Cluster API 的使用。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 有集群 ID 值。

流程

1. 创建一个类似如下的 YAML 文件：此流程使用 **<cluster_resource_file>.yaml** 作为示例文件名。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: <cluster_name> 1
  namespace: openshift-cluster-api
spec:
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: <infrastructure_kind> 2
    name: <cluster_name>
    namespace: openshift-cluster-api
```

1 指定集群 ID 作为集群的名称。

2 指定集群的基础架构类型。以下值有效：

- **AWSCluster** ：集群在 Amazon Web Services (AWS) 上运行。
- **GCPCluster** ：集群在 Google Cloud Platform (GCP) 上运行。
- **VSphereCluster** ：集群在 VMware vSphere 上运行。

2. 运行以下命令来创建集群 CR：

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

验证

- 运行以下命令确认集群 CR 已存在：

```
$ oc get cluster
```

输**出示例**

```
NAME           PHASE          AGE    VERSION
<cluster_name>   Provisioning   4h6m
```

当 **PHASE** 的值为 **Provisioned** 时，代表集群资源已就绪。

其他资源

- 集群 API 配置

13.2.1.3. 创建 Cluster API 基础架构资源

您可以通过创建 YAML 清单文件并使用 OpenShift CLI (**oc**) 应用它来创建特定于供应商的基础架构资源。

先决条件

- 您已部署了 OpenShift Container Platform 集群。
- 您已启用了 Cluster API 的使用。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 有集群 ID 值。
- 您已创建了并应用集群资源。

流程

1. 创建一个类似如下的 YAML 文件：此流程使用 **<infrastructure_resource_file>.yaml** 作为示例文件名。

```
apiVersion: infrastructure.cluster.x-k8s.io/<version> 1
kind: <infrastructure_kind> 2
metadata:
  name: <cluster_name> 3
  namespace: openshift-cluster-api
spec: 4
```

1. **apiVersion** 因平台而异。如需更多信息，请参阅您的供应商的 Cluster API 基础架构资源 YAML 示例。以下值有效：
   - **infrastructure.cluster.x-k8s.io/v1beta2**: Amazon Web Services (AWS) 集群使用的版本。
   - **infrastructure.cluster.x-k8s.io/v1beta1** ：Google Cloud Platform (GCP) 和 VMware vSphere 集群使用的版本。
2. 指定集群的基础架构类型。这个值必须与您的平台的值匹配。以下值有效：
   - **AWSCluster**：集群在 AWS 上运行。
   - **GCPCluster**：集群在 GCP 上运行。
   - **VSphereCluster**：集群在 vSphere 上运行。
3. 指定集群的名称。
4. 指定您的环境的详情。这些参数特定于具体的供应商。如需更多信息，请参阅您的供应商的 Cluster API 基础架构资源 YAML 示例。

2. 运行以下命令来创建基础架构 CR：

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

验证

- 要确认已创建了基础架构 CR，请运行以下命令：

```
$ oc get <infrastructure_kind>
```

其中 **<infrastructure_kind>** 是与您的平台对应的值。

输**出示例**

```
NAME           CLUSTER        READY
<cluster_name>   <cluster_name>   true
```

注意

此输出可能包含特定于您的云供应商的其他列。

其他资源

- Amazon Web Services 上 Cluster API 基础架构资源的 YAML 示例
- Google Cloud Platform 上 Cluster API 基础架构资源的 YAML 示例
- VMware vSphere 上 Cluster API 基础架构资源的 YAML 示例

13.2.1.4. 创建 Cluster API 机器模板

您可以通过创建 YAML 清单文件并使用 OpenShift CLI (**oc**) 应用它来创建特定于供应商的机器模板资源。

先决条件

- 您已部署了 OpenShift Container Platform 集群。
- 您已启用了 Cluster API 的使用。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您已创建了并应用集群和基础架构资源。

流程

1. 创建一个类似如下的 YAML 文件：此流程使用 **<machine_template_resource_file>.yaml** 作为示例文件名。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: <machine_template_kind> 1
```

```
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
```

1 指定机器模板类型。这个值必须与您的平台的值匹配。以下值有效：

- **AWSMachineTemplate**：集群在 Amazon Web Services (AWS) 上运行。
- **gcpmachinetemplate**：集群在 Google Cloud Platform (GCP) 上运行。
- **VSphereMachineTemplate** ：集群在 VMware vSphere 上运行。

2 为机器模板指定名称。

3 指定您的环境的详情。这些参数特定于具体的供应商。如需更多信息，请参阅您的供应商的 Cluster API 机器模板 YAML 示例。

2. 运行以下命令来创建虚拟机模板 CR：

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

验证

- 运行以下命令确认机器模板 CR 已创建：

```
$ oc get <machine_template_kind>
```

这里的 **<machine_template_kind>** 是与您的平台对应的值。

输出示例

```
NAME              AGE
<template_name>   77m
```

其他资源

- Amazon Web Services 上 Cluster API 机器模板资源的 YAML 示例
- Google Cloud Platform 上的 Cluster API 机器模板资源的 YAML 示例
- VMware vSphere 上 Cluster API 机器模板资源的 YAML 示例

13.2.1.5. 创建 Cluster API 计算机器集

您可以创建使用集群 API 的计算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

- 您已部署了 OpenShift Container Platform 集群。

- 您已启用了 Cluster API 的使用。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您已创建了集群、基础架构和机器模板资源。

流程

1. 创建一个类似如下的 YAML 文件：此流程使用 **<machine_set_resource_file>.yaml** 作为示例文件名。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec: 3
# ...
```

1. 为计算机器设置指定一个名称。
2. 指定集群的名称。
3. 指定您的环境的详情。这些参数特定于具体的供应商。如需更多信息，请参阅您的供应商的 Cluster API 计算机器设置 YAML 示例。

2. 运行以下命令来创建计算机器设置 CR：

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

3. 运行以下命令确认已创建了计算机器设置 CR：

```
$ oc get machineset -n openshift-cluster-api 1
```

1. 指定 **openshift-cluster-api** 命名空间。

输出示例

```
NAME                CLUSTER         REPLICAS   READY   AVAILABLE   AGE   VERSION
<machine_set_name>  <cluster_name>  1          1       1           17m
```

当新的计算机器集可用时，**REPLICAS** 和 **AVAILABLE** 值会匹配。如果 compute 机器集不可用，请等待几分钟，然后再次运行命令。

验证

- 要验证计算机器是否会根据所需配置创建机器，请按照以下步骤查看集群中的机器和节点列表：
  - 查看 Cluster API 机器列表：

```
$ oc get machine -n openshift-cluster-api 1
```

1 指定 **openshift-cluster-api** 命名空间。

输出示例

```
NAME                              CLUSTER        NODENAME                              PROVIDERID
PHASE     AGE     VERSION
<machine_set_name>-<string_id>   <cluster_name>   <ip_address>.
<region>.compute.internal   <provider_id>   Running   8m23s
```

  - 查看节点列表：

```
$ oc get node
```

输出示例

```
NAME                                         STATUS   ROLES    AGE     VERSION
<ip_address_1>.<region>.compute.internal   Ready    worker   5h14m   v1.28.5
<ip_address_2>.<region>.compute.internal   Ready    master   5h19m   v1.28.5
<ip_address_3>.<region>.compute.internal   Ready    worker   7m      v1.28.5
```

其他资源

- Amazon Web Services 上 Cluster API 机器集资源的 YAML 示例
- Google Cloud Platform 上 Cluster API 机器集资源的 YAML 示例
- VMware vSphere 上 Cluster API 机器集资源的 YAML 示例

13.3. 使用 CLUSTER API 管理机器

重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

13.3.1. 修改 Cluster API 机器模板

您可以通过修改 YAML 清单文件并使用 OpenShift CLI (**oc**) 应用它来更新集群的机器模板资源。

先决条件

- 您已部署了使用 Cluster API 的 OpenShift Container Platform 集群。
- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 运行以下命令列出集群的机器模板资源：

```
$ oc get <machine_template_kind> 1
```

1 指定与您的平台对应的值。以下值有效：

- **AWSMachineTemplate**：集群在 Amazon Web Services (AWS) 上运行。
- **gcpmachinetemplate**：集群在 Google Cloud Platform (GCP) 上运行。
- **VSphereMachineTemplate** ：集群在 VMware vSphere 上运行。

输**出示例**

```
NAME              AGE
<template_name>   77m
```

2. 运行以下命令，将集群的机器模板资源写入您可以编辑的文件：

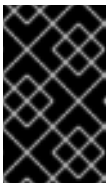
```
$ oc get <template_name> -o yaml > <template_name>.yaml
```

其中 **<template_name>** 是集群的机器模板资源的名称。

3. 使用一个不同的名称生成 **<template_name>.yaml** 文件的副本。此流程使用 **<modified_template_name>.yaml** 作为示例文件名。

4. 使用文本编辑器更改 **<modified_template_name>.yaml** 文件，该文件为集群定义更新的机器模板资源。在编辑机器模板资源时，请观察以下内容：

- **spec** 小节中的参数特定于供应商。如需更多信息，请参阅您的供应商的 Cluster API 机器模板 YAML 示例。
- 您必须为与任何现有值不同的 **metadata.name** 参数使用值。



重要

对于引用此模板的任何 Cluster API 计算机器集，您必须更新 **spec.template.spec.infrastructureRef.name** 参数，以匹配新机器模板资源中的 **metadata.name** 值。

5. 运行以下命令来应用机器模板 CR：

```
$ oc apply -f <modified_template_name>.yaml 1
```

1 使用带有新名称编辑的 YAML 文件。

后续步骤

- 对于引用此模板的任何 Cluster API 计算机器集，更新 **spec.template.spec.infrastructureRef.name** 参数，以匹配新机器模板资源中的 **metadata.name** 值。如需更多信息，请参阅"使用 CLI 修改计算机器集"。

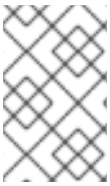
其他资源

- Amazon Web Services 上 Cluster API 机器模板资源的 YAML 示例
- Google Cloud Platform 上的 Cluster API 机器模板资源的 YAML 示例
- VMware vSphere 上 Cluster API 机器模板资源的 YAML 示例
- 使用 CLI 修改计算机器集

13.3.2. 使用 CLI 修改计算机器集

当您修改计算机器集时，您的更改只适用于在保存更新的 **MachineSet** 自定义资源 (CR) 后创建的计算机器。更改不会影响现有的机器。您可以通过扩展计算机器集来将现有机器替换为反映更新的配置的新机器。

如果您需要在不进行其他更改的情况下扩展计算机器，则不需要删除机器。



注意

默认情况下，OpenShift Container Platform 路由器 Pod 部署在计算机器上。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 计算机器集扩展为 **0**。

此流程中的输出示例使用 AWS 集群的值。

先决条件

- OpenShift Container Platform 集群使用 Cluster API。
- 以管理员身份使用 OpenShift CLI (**oc**) 登录集群。

流程

1. 运行以下命令列出集群中的计算机器集：

```
$ oc get machinesets.cluster.x-k8s.io -n openshift-cluster-api
```

输**出示例**

```
NAME                        CLUSTER          REPLICAS   READY   AVAILABLE   AGE
VERSION
<compute_machine_set_name_1>  <cluster_name>     1          1       1          26m
<compute_machine_set_name_2>  <cluster_name>     1          1       1          26m
```

2. 运行以下命令来编辑计算机器集：

```
$ oc edit machinesets.cluster.x-k8s.io <machine_set_name> \
  -n openshift-cluster-api
```

3. 请注意 **spec.replicas** 字段的值，因为在扩展机器集时需要它来应用更改。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name>
  namespace: openshift-cluster-api
spec:
  replicas: 2 (1)
# ...
```

(1) 此流程中的示例显示具有 **replicas** 值 **2** 的计算机器集。

4. 使用您想要的配置选项来更新计算机器设置 CR，并保存您的更改。

5. 运行以下命令，列出由更新的计算机器集管理的机器：

```
$ oc get machines.cluster.x-k8s.io \
  -n openshift-cluster-api \
  -l cluster.x-k8s.io/set-name=<machine_set_name>
```

AWS 集群的输出示例

```
NAME                      CLUSTER        NODENAME                                PROVIDERID
PHASE         AGE     VERSION
<machine_name_original_1>  <cluster_name>  <original_1_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running        4h
<machine_name_original_2>  <cluster_name>  <original_2_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running        4h
```

6. 对于由更新的计算机器集管理的每台机器，请运行以下命令设置 **delete** 注解：

```
$ oc annotate machines.cluster.x-k8s.io/<machine_name_original_1> \
  -n openshift-cluster-api \
  cluster.x-k8s.io/delete-machine="true"
```

7. 要使用新配置创建替换机器，请运行以下命令将计算机器设置为两倍：

```
$ oc scale --replicas=4 \ (1)
  machinesets.cluster.x-k8s.io <machine_set_name> \
  -n openshift-cluster-api
```

(1) 原始示例值 **2** 加倍到 **4**。

8. 运行以下命令，列出由更新的计算机器集管理的机器：

```
$ oc get machines.cluster.x-k8s.io \
  -n openshift-cluster-api \
  -l cluster.x-k8s.io/set-name=<machine_set_name>
```

AWS 集群的输出示例

```
NAME                      CLUSTER         NODENAME                                  PROVIDERID
PHASE         AGE     VERSION
<machine_name_original_1>  <cluster_name>  <original_1_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running       4h
<machine_name_original_2>  <cluster_name>  <original_2_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running       4h
<machine_name_updated_1>   <cluster_name>  <updated_1_ip>.
<region>.compute.internal   aws:///us-east-2a/i-04e7b2cbd61fd2075  Provisioned    55s
<machine_name_updated_2>   <cluster_name>  <updated_2_ip>.
<region>.compute.internal   aws:///us-east-2a/i-04e7b2cbd61fd2075  Provisioning   55s
```

当新机器处于 **Running** 阶段时，您可以将计算机器设置为原始副本数。

9. 要删除使用旧配置创建的机器，请运行以下命令将计算机器设置为原始副本数：

```
$ oc scale --replicas=2 \ (1)
  machinesets.cluster.x-k8s.io <machine_set_name> \
  -n openshift-cluster-api
```

(1) 原始示例值 **2**。

验证

- 要验证更新机器集创建的机器是否具有正确的配置，请运行以下命令检查 CR 中的相关字段是否有新机器：

```
$ oc describe machines.cluster.x-k8s.io <machine_name_updated_1> \
  -n openshift-cluster-api
```

- 要验证没有更新的配置的计算机器，请运行以下命令列出由更新的计算机器集管理的机器：

```
$ oc get machines.cluster.x-k8s.io \
  -n openshift-cluster-api \
  cluster.x-k8s.io/set-name=<machine_set_name>
```

删除时的输出示例是 AWS 集群的进度

```
NAME                      CLUSTER         NODENAME                                  PROVIDERID
PHASE     AGE     VERSION
<machine_name_original_1>  <cluster_name>  <original_1_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running   18m
<machine_name_original_2>  <cluster_name>  <original_2_ip>.<region>.compute.internal
aws:///us-east-2a/i-04e7b2cbd61fd2075  Running   18m
<machine_name_updated_1>   <cluster_name>  <updated_1_ip>.
```

```
<region>.compute.internal    aws:///us-east-2a/i-04e7b2cbd61fd2075   Running   18m
<machine_name_updated_2>   <cluster_name>   <updated_2_ip>.
<region>.compute.internal    aws:///us-east-2a/i-04e7b2cbd61fd2075   Running   18m
```

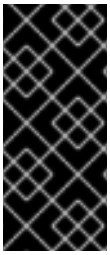
为 **AWS 集群完成删除时的输出示例**

```
NAME                 CLUSTER        NODENAME                                PROVIDERID
PHASE     AGE     VERSION
<machine_name_updated_1>   <cluster_name>   <updated_1_ip>.
<region>.compute.internal    aws:///us-east-2a/i-04e7b2cbd61fd2075   Running   18m
<machine_name_updated_2>   <cluster_name>   <updated_2_ip>.
<region>.compute.internal    aws:///us-east-2a/i-04e7b2cbd61fd2075   Running   18m
```

其他资源

- Amazon Web Services 上 Cluster API 机器集资源的 YAML 示例
- Google Cloud Platform 上 Cluster API 机器集资源的 YAML 示例
- VMware vSphere 上 Cluster API 机器集资源的 YAML 示例

13.4. 集群 API 配置



重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

以下示例 YAML 文件演示了如何使 Cluster API 主资源协同工作，并为它们创建适合您环境的机器配置设置。

13.4.1. Cluster API 集群资源的 YAML 示例

集群资源定义集群的名称和基础架构供应商，并由 Cluster API 管理。此资源对所有提供程序具有相同的结构。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: <cluster_name> 1
  namespace: openshift-cluster-api
spec:
  controlPlaneEndpoint: 2
    host: <control_plane_endpoint_address>
    port: 6443
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: <infrastructure_kind> 3
    name: <cluster_name>
    namespace: openshift-cluster-api
```

-

1. 指定集群的名称。
2. 指定 control plane 端点的 IP 地址以及用于访问它的端口。
3. 指定集群的基础架构类型。有效值为：
    - **AWSCluster** ：集群在 Amazon Web Services 上运行。
    - **GCPCluster** ：集群在 Google Cloud Platform 上运行。
    - **VSphereCluster** ：集群在 VMware vSphere 上运行。

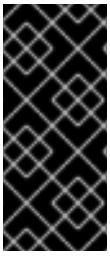
13.4.2. 特定于供应商的配置选项

剩余的 Cluster API 资源是特定于供应商的。有关集群的特定于供应商的配置选项，请参阅以下资源：

- Amazon Web Services 的集群 API 配置选项
- Google Cloud Platform 的集群 API 配置选项
- VMware vSphere 的集群 API 配置选项

13.5. CLUSTER API 机器的配置选项

13.5.1. Amazon Web Services 的集群 API 配置选项



重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

您可以通过更新 Cluster API 自定义资源清单中的值来更改 Amazon Web Services (AWS) 集群 API 机器的配置。

13.5.1.1. 用于配置 Amazon Web Services 集群的 YAML 示例

以下示例 YAML 文件显示了 Amazon Web Services 集群的配置。

13.5.1.1.1. Amazon Web Services 上 Cluster API 基础架构资源的 YAML 示例

基础架构资源是特定于提供程序的，定义由集群中所有计算机器集（如地区和子网）共享的属性。计算机器集在创建机器时引用此资源。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: AWSCluster 1
metadata:
  name: <cluster_name> 2
  namespace: openshift-cluster-api
```

```
spec:
  controlPlaneEndpoint: 3
    host: <control_plane_endpoint_address>
    port: 6443
  region: <region> 4
```

1 指定集群的基础架构类型。这个值必须与您的平台的值匹配。

2 指定集群 ID 作为集群的名称。

3 指定 control plane 端点的地址以及用于访问它的端口。

4 指定 AWS 区域。

13.5.1.1.2. Amazon Web Services 上 Cluster API 机器模板资源的 YAML 示例

机器模板资源是特定于提供程序的，定义计算机器创建的机器的基本属性。计算机器设置在创建机器时引用此模板。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: AWSMachineTemplate 1
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      uncompressedUserData: true
      iamInstanceProfile: # ...
      instanceType: m5.large
      ignition:
        storageType: UnencryptedUserData
        version: "3.2"
      ami:
        id: # ...
      subnet:
        filters:
        - name: tag:Name
          values:
          - # ...
      additionalSecurityGroups:
      - filters:
        - name: tag:Name
          values:
          - # ...
```

1 指定机器模板类型。这个值必须与您的平台的值匹配。

2 为机器模板指定名称。

3 指定您的环境的详情。这里的值为示例。

13.5.1.1.3. Amazon Web Services 上 Cluster API 机器集资源的 YAML 示例

计算机器设置资源定义它所创建的机器的额外属性。计算机器集也会在创建机器时引用基础架构资源和机器模板。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: AWSMachineTemplate 4
        name: <template_name> 5
```

1. 为计算机器设置指定一个名称。
2. 指定集群 ID 作为集群的名称。
3. 对于 Cluster API 技术预览，Operator 可以使用 **openshift-machine-api** 命名空间中的 worker 用户数据 secret。
4. 指定机器模板类型。这个值必须与您的平台的值匹配。
5. 指定机器模板名称。

13.5.2. Google Cloud Platform 的集群 API 配置选项

重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

您可以通过更新 Cluster API 自定义资源清单中的值来更改 Google Cloud Platform (GCP) Cluster API 机器的配置。

13.5.2.1. 用于配置 Google Cloud Platform 集群的 YAML 示例

以下示例 YAML 文件显示了 Google Cloud Platform 集群的配置。

13.5.2.1.1. Google Cloud Platform 上 Cluster API 基础架构资源的 YAML 示例

基础架构资源是特定于提供程序的，定义由集群中所有计算机器集（如地区和子网）共享的属性。计算机器集在创建机器时引用此资源。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPCluster (1)
metadata:
  name: <cluster_name> (2)
spec:
  controlPlaneEndpoint: (3)
    host: <control_plane_endpoint_address>
    port: 6443
  network:
    name: <cluster_name>-network
  project: <project> (4)
  region: <region> (5)
```

1. 指定集群的基础架构类型。这个值必须与您的平台的值匹配。
2. 指定集群 ID 作为集群的名称。
3. 指定 control plane 端点的 IP 地址以及用于访问它的端口。
4. 指定 GCP 项目名称。
5. 指定 GCP 区域。

13.5.2.1.2. Google Cloud Platform 上的 Cluster API 机器模板资源的 YAML 示例

机器模板资源是特定于提供程序的，定义计算机器创建的机器的基本属性。计算机器设置在创建机器时引用此模板。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPMachineTemplate (1)
metadata:
  name: <template_name> (2)
  namespace: openshift-cluster-api
spec:
  template:
    spec: (3)
      rootDeviceType: pd-ssd
      rootDeviceSize: 128
      instanceType: n1-standard-4
      image: projects/rhcos-cloud/global/images/rhcos-411-85-202203181601-0-gcp-x86-64
      subnet: <cluster_name>-worker-subnet
      serviceAccounts:
        email: <service_account_email_address>
        scopes:
          - https://www.googleapis.com/auth/cloud-platform
      additionalLabels:
```

```
    kubernetes-io-cluster-<cluster_name>: owned
  additionalNetworkTags:
   - <cluster_name>-worker
  ipForwarding: Disabled
```

1. 指定机器模板类型。这个值必须与您的平台的值匹配。
2. 为机器模板指定名称。
3. 指定您的环境的详情。这里的值为示例。

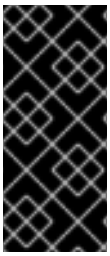
13.5.2.1.3. Google Cloud Platform 上 Cluster API 机器集资源的 YAML 示例

计算机器设置资源定义它所创建的机器的额外属性。计算机器集也会在创建机器时引用基础架构资源和机器模板。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: GCPMachineTemplate 4
        name: <template_name> 5
      failureDomain: <failure_domain> 6
```

1. 为计算机器设置指定一个名称。
2. 指定集群 ID 作为集群的名称。
3. 对于 Cluster API 技术预览，Operator 可以使用 **openshift-machine-api** 命名空间中的 worker 用户数据 secret。
4. 指定机器模板类型。这个值必须与您的平台的值匹配。
5. 指定机器模板名称。
6. 指定 GCP 区域中的故障域。

13.5.3. VMware vSphere 的集群 API 配置选项



重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

您可以通过更新 Cluster API 自定义资源清单中的值来更改 VMware vSphere Cluster API 机器的配置。

13.5.3.1. 用于配置 VMware vSphere 集群的 YAML 示例

以下示例 YAML 文件显示 VMware vSphere 集群的配置。

13.5.3.1.1. VMware vSphere 上 Cluster API 基础架构资源的 YAML 示例

基础架构资源是特定于提供程序的，定义由集群中所有计算机器集（如地区和子网）共享的属性。计算机器集在创建机器时引用此资源。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: VSphereCluster 1
metadata:
  name: <cluster_name> 2
spec:
  controlPlaneEndpoint: 3
    host: <control_plane_endpoint_address>
    port: 6443
  identityRef:
    kind: Secret
    name: <cluster_name>
  server: <vsphere_server> 4
```

1. 指定集群的基础架构类型。这个值必须与您的平台的值匹配。
2. 指定集群 ID 作为集群的名称。
3. 指定 control plane 端点的 IP 地址以及用于访问它的端口。
4. 指定集群的 vSphere 服务器。您可以运行以下命令来在现有 vSphere 集群上找到这个值：

```
$ oc get infrastructure cluster \
  -o jsonpath="{.spec.platformSpec.vsphere.vcenters[0].server}"
```

13.5.3.1.2. VMware vSphere 上 Cluster API 机器模板资源的 YAML 示例

机器模板资源是特定于提供程序的，定义计算机器创建的机器的基本属性。计算机器设置在创建机器时引用此模板。

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: VSphereMachineTemplate 1
```

```
metadata:
  name: <template_name> 2
  namespace: openshift-cluster-api
spec:
  template:
    spec: 3
      template: <vm_template_name> 4
      server: <vcenter_server_ip> 5
      diskGiB: 128
      cloneMode: linkedClone 6
      datacenter: <vcenter_datacenter_name> 7
      datastore: <vcenter_datastore_name> 8
      folder: <vcenter_vm_folder_path> 9
      resourcePool: <vsphere_resource_pool> 10
      numCPUs: 4
      memoryMiB: 16384
      network:
        devices:
        - dhcp4: true
          networkName: "<vm_network_name>" 11
```

1. 指定机器模板类型。这个值必须与您的平台的值匹配。
2. 为机器模板指定名称。
3. 指定您的环境的详情。这里的值为示例。
4. 指定要使用的 vSphere 虚拟机模板，如 **user-5ddjd-rhcos**。
5. 指定 vCenter 服务器 IP 或完全限定域名。
6. 指定要使用的虚拟机克隆类型。以下值有效：
   - **fullClone**
   - **linkedClone**

   使用 **linkedClone** 类型时，磁盘大小与克隆源匹配，而不使用 **diskGiB** 值。如需更多信息，请参阅有关虚拟机克隆类型的 vSphere 文档。
7. 指定要将计算机器设置为的 vCenter Datacenter。
8. 指定要部署计算机器的 vCenter Datastore。
9. 指定 vCenter 中 vSphere 虚拟机文件夹的路径，如 **/dc1/vm/user-inst-5ddjd**。
10. 指定虚拟机的 vSphere 资源池。
11. 指定要将计算机器设置为的 vSphere VM 网络。此虚拟机网络必须是集群中其他计算机器所处的位置。

13.5.3.1.3. VMware vSphere 上 Cluster API 机器集资源的 YAML 示例

计算机器设置资源定义它所创建的机器的额外属性。计算机器集也会在创建机器时引用基础架构资源和机器模板。

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineSet
metadata:
  name: <machine_set_name> 1
  namespace: openshift-cluster-api
spec:
  clusterName: <cluster_name> 2
  replicas: 1
  selector:
    matchLabels:
      test: example
  template:
    metadata:
      labels:
        test: example
    spec:
      bootstrap:
        dataSecretName: worker-user-data 3
      clusterName: <cluster_name>
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: VSphereMachineTemplate 4
        name: <template_name> 5
      failureDomain: 6
        - name: <failure_domain_name>
          region: <region_a>
          zone: <zone_a>
          server: <vcenter_server_name>
          topology:
            datacenter: <region_a_datacenter>
            computeCluster: "</region_a_datacenter/host/zone_a_cluster>"
            resourcePool: "</region_a_datacenter/host/zone_a_cluster/Resources/resource_pool>"
            datastore: "</region_a_datacenter/datastore/datastore_a>"
            networks:
            - port-group
```

1. 为计算机器设置指定一个名称。
2. 指定集群 ID 作为集群的名称。
3. 对于 Cluster API 技术预览，Operator 可以使用 **openshift-machine-api** 命名空间中的 worker 用户数据 secret。
4. 指定机器模板类型。这个值必须与您的平台的值匹配。
5. 指定机器模板名称。
6. 指定故障域配置详情。

注意

在使用 Cluster API 的 vSphere 集群上使用多个区域（region）和区（zone）是没有经过验证的配置。

13.6. 对使用 CLUSTER API 的集群进行故障排除

重要

使用集群 API 管理机器只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

使用本节中的信息从您可能会遇到的问题了解和恢复。通常，对集群 API 问题进行故障排除的步骤与 Machine API 存在问题的步骤类似。

Cluster CAPI Operator 及其操作对象在 **openshift-cluster-api** 命名空间中置备，而 Machine API 使用 **openshift-machine-api** 命名空间。使用引用命名空间的 **oc** 命令时，请务必引用正确的命名空间。

13.6.1. 使用 CLI 引用预期的对象

对于使用 Cluster API 的集群，OpenShift CLI (**oc**) 命令会优先选择 Cluster API（与 Machine API 对象相比）。

此行为会影响任何对集群 API 和 Machine API 中代表的对象执行操作的 **oc** 命令。本说明使用 **oc delete machine** 命令，该命令删除机器，作为示例。

原因
运行 **oc** 命令时，**oc** 与 Kube API 服务器通信以确定要采取哪些对象。Kube API 服务器使用第一个安装的自定义资源定义 (CRD)，它在运行 **oc** 命令时按字母顺序进行。

Cluster API 对象的 CRD 位于 **cluster.x-k8s.io** 组中，而 Machine API 对象的 CRD 位于 **machine.openshift.io** 组中。因为字母 **c** 在字母 **m** 前面，所以 Kube API 服务器在 Cluster API 对象 CRD 上匹配。因此，**oc** 命令会操作 Cluster API 对象。

结果
由于此行为，使用 Cluster API 的集群中可能会出现以下意外结果：

- 对于包含两种类型的对象的命名空间，**oc get machine** 等命令只返回 Cluster API 对象。
- 对于仅包含 Machine API 对象的命名空间，**oc get machine** 等命令不会返回任何结果。

临时解决方案
您可以使用对应的完全限定名称来确保 **oc** 命令对您想要的对象类型进行操作。

先决条件

- 您可以使用具有 **cluster-admin** 权限的账户访问集群。
- 已安装 OpenShift CLI（**oc**）。

流程

- 要删除 Machine API 机器，在运行 **oc delete machine** 命令时使用完全限定名称 **machine.machine.openshift.io** ：

```
$ oc delete machine.machine.openshift.io <machine_name>
```

- 要删除 Cluster API 机器，在运行 **oc delete machine** 命令时使用完全限定名称 **machine.cluster.x-k8s.io** ：

```
$ oc delete machine.cluster.x-k8s.io <machine_name>
```

第 14 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

您只能在 Machine API 操作的集群中使用高级机器管理和扩展功能。具有用户置备的基础架构的集群需要额外的验证和配置才能使用 Machine API。

具有基础架构平台类型 **none** 的集群无法使用 Machine API。即使附加到集群的计算机器安装在支持该功能的平台上，也会应用这个限制。在安装后无法更改此参数。

要查看集群的平台类型，请运行以下命令：

```
$ oc get infrastructure cluster -o jsonpath='{.status.platform}'
```

14.1. 关于机器健康检查

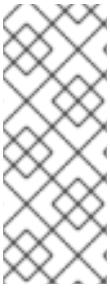
注意

您只能对由计算机器集或 control plane 机器集管理的机器应用机器健康检查。

要监控机器的健康状况，创建资源来定义控制器的配置。设置要检查的条件（例如，处于 **NotReady** 状态达到五分钟或 node-problem-detector 中显示了持久性状况），以及用于要监控的机器集合的标签。

监控 **MachineHealthCheck** 资源的控制器会检查定义的条件。如果机器无法进行健康检查，则会自动删除机器并创建一个机器来代替它。删除机器之后，您会看到**机器被删除**事件。

为限制删除机器造成的破坏性影响，控制器一次仅清空并删除一个节点。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，则补救会停止，需要启用手动干预。



注意

请根据工作负载和要求仔细考虑超时。

- 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。
- 超时时间太短可能会导致补救循环。例如，检查 **NotReady** 状态的超时时间必须足够长，以便机器能够完成启动过程。

要停止检查，请删除资源。

14.1.1. 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。
- 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

- 关于列出集群中的所有节点
- 短路机器健康检查补救
- 关于 Control Plane Machine Set Operator

14.2. MACHINEHEALTHCHECK 资源示例

所有基于云的安装类型的 **MachineHealthCheck** 资源，以及裸机以外的资源，类似以下 YAML 文件：

```
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. 指定目标池中允许同时修复的机器数量。这可设为一个百分比或一个整数。如果不健康的机器数量超过 **maxUnhealthy** 设定的限制，则不会执行补救。
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。

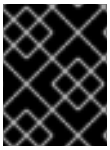
注意

matchLabels 只是示例; 您必须根据具体需要映射您的机器组。

14.2.1. 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。

适当的 **maxUnhealthy** 值取决于您部署的集群规模以及 **MachineHealthCheck** 覆盖的机器数量。例如，您可以使用 **maxUnhealthy** 值覆盖多个可用区间的多个计算机器集，以便在丢失整个区时，**maxUnhealthy** 设置可以在集群中防止进一步补救。在没有多个可用区的全局 Azure 区域，您可以使用可用性集来确保高可用性。

重要

如果您为 control plane 配置 **MachineHealthCheck** 资源，请将 **maxUnhealthy** 的值设置为 **1**。

此配置可确保当多个 control plane 机器显示为不健康时，机器健康检查不会采取任何操作。多个不健康的 control plane 机器可能会表示 etcd 集群已降级或扩展操作来替换失败的机器。

如果 etcd 集群降级，可能需要手动干预。如果扩展操作正在进行，机器健康检查应该允许它完成。

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。

14.2.1.1. 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。

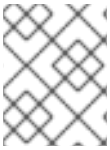
14.2.1.2. 使用百分比设置 maxUnhealthy

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。

14.3. 创建机器健康检查资源

您可以为集群中的机器集创建 **MachineHealthCheck** 资源。

注意

您只能对由计算机器集或 control plane 机器集管理的机器应用机器健康检查。

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
```

您可以配置和部署机器健康检查，以检测并修复不健康的裸机节点。

14.4. 关于裸机的基于电源的补救

在裸机集群中，修复节点对于确保集群的整体健康状况至关重要。以物理方式修复集群可能会有一定难度，且在使机器进入安全或操作状态时出现任何延迟，这会增加集群处于降级状态的时间，以及后续故障可能会导致集群离线的风险。基于电源的补救可帮助解决此类问题。

基于电源的补救不重新置备节点，而是使用电源控制器关闭不可操作的节点。这种类型的补救也称为电源隔离。

OpenShift Container Platform 使用 **MachineHealthCheck** 控制器来检测出现故障的裸机节点。基于电源的补救速度会较快，它只重启有问题的节点，而不是从集群中移除。

基于电源的补救提供以下功能：

- 允许恢复 control plane 节点
- 在超融合环境中降低数据丢失的风险
- 减少了因为恢复物理机器造成的停机时间

14.4.1. 裸机上的 MachineHealthCheck

在裸机集群上删除机器会触发重新置备裸机主机。通常，裸机重新置备是一个需要较长时间的过程，在这个过程中，集群缺少计算资源，应用程序可能会中断。

将默认补救过程从机器删除改为主机电源周期的方法有两种：

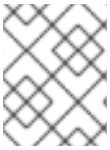
1. 使用 **machine.openshift.io/remediation-strategy: external-baremetal** 注解来注解 **MachineHealthCheck** 资源。
2. 创建 **Metal3RemediationTemplate** 资源，并在 **MachineHealthCheck** 的 **spec.remediationTemplate** 中引用它。

使用其中一种方法后，不健康的机器会使用 Baseboard Management Controller (BMC) 凭证进行节能。

14.4.2. 了解基于注解的补救过程

补救过程按如下方式运行：

1. MachineHealthCheck（MHC）控制器检测到节点不健康。
2. MHC 通知裸机控制器，它请求关闭不健康的节点。
3. 关闭电源后，节点会被删除，这允许集群将受影响的工作负载重新调度到其他节点上。
4. 裸机机器控制器请求启动节点。
5. 节点启动后，节点会重新注册到集群，从而会创建新节点。
6. 重新创建节点后，裸机控制器会在删除前恢复不健康节点上存在的注解和标签。



注意

如果电源操作未完成，裸机控制器会触发不健康节点的重新置备，除非这是 control plane 节点或外部置备的节点。

14.4.3. 了解基于 metal3 的补救过程

补救过程按如下方式运行：

1. MachineHealthCheck（MHC）控制器检测到节点不健康。
2. MHC 为 metal3 补救控制器创建一个 metal3 补救自定义资源，它请求关闭不健康的节点。
3. 关闭电源后，节点会被删除，这允许集群将受影响的工作负载重新调度到其他节点上。
4. metal3 补救控制器请求，以便在节点上电源。
5. 节点启动后，节点会重新注册到集群，从而会创建新节点。
6. 重新创建节点后，metal3 补救控制器会在删除前恢复不健康节点上存在的注解和标签。

注意

如果电源操作没有完成，metal3 补救控制器会触发不健康节点的重新置备，除非这是 control plane 节点或外部置备的节点。

14.4.4. 为裸机创建 MachineHealthCheck 资源

先决条件

- OpenShift Container Platform 使用安装程序置备的基础架构（IPI）安装。

- 访问 BMC 凭证（或 BMC 访问每个节点）。
- 网络访问不健康节点的 BMC 接口。

流程

1. 创建一个 **healthcheck.yaml** 文件，其中包含您的机器健康检查的定义。
2. 使用以下命令将 **healthcheck.yaml** 文件应用到集群：

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

裸机的 MachineHealthCheck 资源示例，基于注解的补救

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example (1)
  namespace: openshift-machine-api
  annotations:
    machine.openshift.io/remediation-strategy: external-baremetal (2)
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role> (3)
      machine.openshift.io/cluster-api-machine-type: <role> (4)
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone> (5)
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s" (6)
    status: "False"
  - type:    "Ready"
    timeout: "300s" (7)
    status: "Unknown"
  maxUnhealthy: "40%" (8)
  nodeStartupTimeout: "10m" (9)
```

(1) 指定要部署的机器健康检查的名称。

(2) 对于裸机集群，您必须在 **annotations** 部分中包含 **machine.openshift.io/remediation-strategy: external-baremetal** 注解来启用电源周期补救。采用这种补救策略时，不健康的主机会被重启，而不是从集群中删除。

(3) (4) 为要检查的机器池指定一个标签。

(5) 以**<cluster_name>-<label>-<zone>** 格式指定要跟踪的计算机器。例如，**prod-node-us-east-1a**。

(6) (7) 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。

(8) 指定目标池中允许同时修复的机器数量。这可设为一个百分比或一个整数。如果不健康的机器数量超过 **maxUnhealthy** 设定的限制，则不会执行补救。

(9) 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

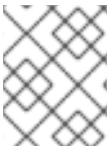
matchLabels 只是示例; 您必须根据具体需要映射您的机器组。

裸机的 MachineHealthCheck 资源示例，基于 metal3 的补救

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-machine-role: <role>
      machine.openshift.io/cluster-api-machine-type: <role>
      machine.openshift.io/cluster-api-machineset: <cluster_name>-<label>-<zone>
  remediationTemplate:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: Metal3RemediationTemplate
    name: metal3-remediation-template
    namespace: openshift-machine-api
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s"
```

裸机的 Metal3RemediationTemplate 资源示例，基于 metal3 的补救

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3RemediationTemplate
metadata:
  name: metal3-remediation-template
  namespace: openshift-machine-api
spec:
  template:
    spec:
      strategy:
        type: Reboot
        retryLimit: 1
        timeout: 5m0s
```



注意

matchLabels 只是示例; 您必须根据具体需要映射您的机器组。**annotations** 部分不适用于基于 metal3 的补救。基于注解的补救和基于 metal3 的补救是互斥的。

<mgmt-troubleshooting-issue-power-remediation_deploying-machine-health-checks><title>基于电源补救的故障排除</title>
要排除基于电源补救的问题，请验证以下内容：

- 您可以访问 BMC。
- BMC 连接到负责运行补救任务的 control plane 节点。

</mgmt-troubleshooting-issue-power-remediation_deploying-machine-health-checks>