OpenShift Container Platform 4.9

机器管理

添加和维护集群机器

Last Updated: 2023-04-14

OpenShift Container Platform 4.9 机器管理

添加和维护集群机器

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 Amazon Web Services(AWS)、Azure、Google Cloud Platform(GCP)、OpenStack、Red Hat Virtualization(RHV)和 vSphere 等底层基础架构，以管理 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。

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

机器管理作为 自定义资源定义(CRD)来实施。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
- Cluster Autoscaler
- Machine Autoscaler
- 机器健康检查

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 4.9 集群，Machine API 在集群安装完成后执行所有节点主机置备管理操作。由于此系统的缘故，OpenShift Container Platform 4.9 在公有或私有云基础架构之上提供了一种弹性动态置备方法。

两种主要资源分别是：

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，Amazon Web Services (AWS) 上的 worker 节点的机器类型可能会定义特定的机器类型和所需的元数据。

机器集

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

> **警告**
>
> control plane 机器不能由机器集管理。

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

机器健康检查

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

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

1.2. 管理计算机器

作为集群管理员，您可以：

- 在以下位置创建机器集：
  - AWS
  - Azure
  - GCP
  - OpenStack
  - RHV
  - vSphere
- 通过添加或删除机器集来手动扩展机器集。
- 通过 MachineSet YAML 配置文件修改机器集。
- 删除机器。
- 创建基础架构机器集。
- 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. 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。

1.4. 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。

1.5. 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 AWS 上创建机器集

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

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. AWS 上机器设置自定义资源的 YAML 示例

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

在本例中，**<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: awsproviderconfig.openshift.io/v1beta1
    blockDevices:
      - ebs:
          iops: 0
          volumeSize: 120
          volumeType: gp2
    credentialsSecret:
      name: aws-cloud-credentials
    deviceIndex: 0
    iamInstanceProfile:
      id: <infrastructure_id>-worker-profile (11)
    instanceType: m4.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
    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、节点标签和区域。

6 7 9 指定要添加的节点标签。

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

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。

2.1.2. 创建机器集

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
```

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

2.1.3. 将机器部署为 Spot 实例的机器集

您可以通过创建一个在 AWS 上运行的机器集来把机器部署为非保障的 Spot 实例来节约成本。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。为了满足机器集 **replicas** 数量， 机器集会创建一个请求 Spot 实例的机器。

2.1.4. 使用机器集创建 Spot 实例

您可以通过在机器集 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。

2.1.5. 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。

2.1.6. 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. 在 AZURE 上创建机器集

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

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. Azure 上机器设置自定义资源的 YAML 示例

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

在本例中，**<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: azureproviderconfig.openshift.io/v1beta1
        credentialsSecret:
          name: azure-cloud-credentials
          namespace: openshift-machine-api
        image: 12
          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 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。

12 指定机器集的镜像详情。如果要使用 Azure Marketplace 镜像，请参阅"选择 Azure Marketplace 镜像"。

13 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。

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。

2.2.2. 创建机器集

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
```

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

2.2.3. 选择 Azure Marketplace 镜像

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。如果您要使用仅兼容 1 版本的实例类型，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。

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-
   ```

```
ocpworker:4.8.2021122100              4.8.2021122100
rh-ocp-worker  RedHat         rh-ocp-worker-gen1  RedHat:rh-ocp-worker:rh-ocp-worker-
gen1:4.8.2021122100        4.8.2021122100
```

- 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:4.8.2021122100       4.8.2021122100
rh-ocp-worker  redhat-limited  rh-ocp-worker-gen1  redhat-limited:rh-ocp-worker:rh-ocp-
worker-gen1:4.8.2021122100  4.8.2021122100
```

注意

无论您安装的 OpenShift Container Platform 版本是什么，要使用的 Azure Marketplace 镜像的正确版本为 4.8.x。如果需要，在安装过程中，您的虚拟机会自动升级。

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 计算机器的 providerSpec 镜像值示例

```
providerSpec:
  value:
    image:
      offer: rh-ocp-worker
      publisher: redhat
      resourceID: ""
      sku: rh-ocp-worker
      type: MarketplaceWithPlan
      version: 4.8.2021122100
```

2.2.4. 将机器部署为 Spot 虚拟机的机器

您可以通过创建在 Azure 上运行的机器集将机器部署为非保障的 Spot 虚拟机来节约成本。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。为了满足机器集**副本**数量，机器集会创建一个请求 Spot 虚拟机的机器。

2.2.5. 使用机器集创建 Spot 虚拟机

您可以通过在机器设置 YAML 文件中添加 **spotVMOptions**，在 Azure 上启动 Spot VM。

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。

2.2.6. 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。磁盘加密集必须先在 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

- 您可以在 Azure 文档中了解更多有关客户管理的密钥的信息。

2.3. 在 GCP 上创建机器集

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

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. GCP 上机器设置自定义资源的 YAML 示例

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

在本例中，**<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>-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。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. 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。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
```

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-48-x86-64-202210040145**
- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145**

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。

2.3.2. 创建机器集

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
```

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

2.3.3. 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。为了满足机器集副本数量，机器集会创建一个请求抢占虚拟机实例的机器。

2.3.4. 使用机器集创建抢占虚拟机实例

您可以通过在机器设置 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**。

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

```
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：

```
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。

2.4. 在 OPENSTACK 上创建机器集

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

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. RHOSP 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 Red Hat OpenStack Platform（RHOSP）上运行的机器集，并创建带有 **node-role.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: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: <optional_UUID_of_server_group> 11
          kind: OpenstackProviderSpec
```

```
        networks: 12
        - filter: {}
          subnets:
          - filter:
              name: <subnet_name>
              tags: openshiftClusterID=<infrastructure_id> 13
        primarySubnet: <rhosp_subnet_UUID> 14
        securityGroups:
        - filter: {}
          name: <infrastructure_id>-worker 15
        serverMetadata:
          Name: <infrastructure_id>-worker 16
          openshiftClusterID: <infrastructure_id> 17
        tags:
        - openshiftClusterID=<infrastructure_id> 18
        trunk: true
        userDataSecret:
          name: worker-user-data 19
        availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 13 15 16 17 18 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。

2.4.2. 在 RHOSP 上使用 SR-IOV 的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机器集。

此 YAML 示例定义了一个使用 SR-IOV 网络的机器集。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 machine set custom resource on RHOSP"。

使用 SR-IOV 网络的机器集示例

```
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: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          serverGroupID: <optional_UUID_of_server_group>
          kind: OpenstackProviderSpec
          networks:
            - subnets:
              - UUID: <machines_subnet_UUID>
          ports:
            - networkID: <radio_network_UUID> (1)
              nameSuffix: radio
              fixedIPs:
                - subnetID: <radio_subnet_UUID> (2)
              tags:
                - sriov
                - radio
              vnicType: direct (3)
              portSecurity: false (4)
```

```
          - networkID: <uplink_network_UUID> 5
            nameSuffix: uplink
            fixedIPs:
              - subnetID: <uplink_subnet_UUID> 6
            tags:
              - sriov
              - uplink
            vnicType: direct 7
            portSecurity: false 8
        primarySubnet: <machines_subnet_UUID>
        securityGroups:
        - filter: {}
          name: <infrastructure_id>-<node_role>
        serverMetadata:
          Name: <infrastructure_id>-<node_role>
          openshiftClusterID: <infrastructure_id>
        tags:
        - openshiftClusterID=<infrastructure_id>
        trunk: true
        userDataSecret:
          name: <node_role>-user-data
        availabilityZone: <optional_openstack_availability_zone>
        configDrive: true 9
```

1 5 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。

9 **configDrive** 参数的值必须是 **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

- 在支持 SR-IOV 连接的计算机器的 OpenStack 上安装集群

2.4.3. 禁用端口安全性的 SR-IOV 部署的 YAML 示例

要在禁用端口安全的网络上创建单根 I/O 虚拟化（SR-IOV）端口，请定义一个机器集，将端口作为 **spec.template.spec.providerSpec.value.ports** 列表中的项来包含。与标准 SR-IOV 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机器集示例

```yaml
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id>
    machine.openshift.io/cluster-api-machine-role: <node_role>
    machine.openshift.io/cluster-api-machine-type: <node_role>
  name: <infrastructure_id>-<node_role>
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas>
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <node_role>
        machine.openshift.io/cluster-api-machine-type: <node_role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<node_role>
    spec:
      metadata: {}
      providerSpec:
        value:
          apiVersion: openstackproviderconfig.openshift.io/v1alpha1
          cloudName: openstack
          cloudsSecret:
            name: openstack-cloud-credentials
            namespace: openshift-machine-api
          flavor: <nova_flavor>
          image: <glance_image_name_or_location>
          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
    configDrive: True
```

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。

如果您的集群使用 Kuryr，且 RHOSP SR-IOV 网络禁用端口安全，计算机器的主要端口必须具有：

- **spec.template.spec.providerSpec.value.networks.portSecurityEnabled** 参数的值设置为 **false**。
- 对于每个子网，**spec.template.spec.providerSpec.value.networks.subnets.portSecurityEnabled** 参数的值设置为 **false**。
- **spec.template.spec.providerSpec.value.securityGroups** 的值设为空：**[]**。

Kuryr 上使用 SR-IOV 且禁用端口安全性的集群集的示例部分

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

在这种情况下，您可以在创建虚拟机后将计算安全组应用到主虚拟机接口。例如，从命令行：

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

2.4.4. 创建机器集

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
```

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

2.5. 在 RHV 上创建机器集

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

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. RHV 上机器集自定义资源的 YAML 示例

此 YAML 示例定义了一个在 RHV 上运行的机器集，并创建标记为 **node-role.kubernetes.io/<node_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。

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

```
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
      machine.openshift.io/cluster-api-machine-role: <role> 10
      machine.openshift.io/cluster-api-machine-type: <role> 11
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/<role>: "" 13
    providerSpec:
      value:
        apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
        cluster_id: <ovirt_cluster_id> 14
        template_name: <ovirt_template_name> 15
        instance_type_id: <instance_type_id> 16
        cpu: 17
          sockets: <number_of_sockets> 18
          cores: <number_of_cores> 19
          threads: <number_of_threads> 20
        memory_mb: <memory_size> 21
        os_disk: 22
          size_gb: <disk_size> 23
        network_interfaces: 24
          vnic_profile_id:  <vnic_profile_id> 25
        credentialsSecret:
          name: ovirt-credentials 26
        kind: OvirtMachineProviderSpec
        type: <workload_type> 27
        auto_pinning_policy: <auto_pinning_policy> 28
        hugepages: <hugepages> 29
        affinityGroupsNames:
          - compute 30
        userDataSecret:
          name: worker-user-data
```

1 7 9 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 10 11 13 指定要添加的节点标签。

4 8 12 指定基础架构 ID 和节点标签。这两个字符串不能超过 35 个字符。

5 指定要创建的机器数量。

6 机器的选择器。

14 指定此虚拟机实例所属的 RHV 集群的 UUID。

15 指定用于创建机器的 RHV VM 模板。

16 可选：指定虚拟机实例类型。

> **警告**
>
> **instance_type_id** 字段已弃用，并将在以后的发行版本中删除。

如果包含这个参数，则不需要指定包括 CPU 和内存在内的虚拟机的硬件参数，因为这个参数会覆盖所有硬件参数。

17 可选：CPU 字段包含 CPU 配置，包括插槽、内核和线程。

18 可选：指定虚拟机的插槽数量。

19 可选：指定每个插槽的内核数。

20 可选：指定每个内核的线程数量。

21 可选：指定虚拟机的内存大小（MiB）。

22 可选：节点的 Root 磁盘。

23 可选：指定可引导磁盘的大小（GiB）。

24 可选：虚拟机网络接口列表。如果包含此参数，OpenShift Container Platform 会丢弃来自模板中的所有网络接口并创建新接口。

25 可选：指定 vNIC 配置集 ID。

26 指定包含 RHV 凭证的 secret 名称。

27 可选：指定实例优化的工作负载类型。这个值会影响 **RHV VM** 参数。支持的值包括：**desktop**、**server**（默认）、**high_performance**。**high_performance** 提高了虚拟机的性能，但存在一些限制。例如，您无法使用图形控制台访问虚拟机。如需更多信息，请参阅*虚拟机管理指南*中的配置高性能虚拟机、模板和池。

28 可选： AutoPinningPolicy 定义自动设置 CPU 和 NUMA 设置的策略，包括固定到此实例的主机。支持的值： **none**，**resize_and_pin**。如需更多信息，请参阅《*虚拟机管理指南*》中的 设置 NUMA 节点。

29 可选：Helgepages 是以 KiB 为单位的大小，用于在虚拟机中定义巨页。支持的值： **2048** 或 **1048576**。如需更多信息，请参阅《*虚拟机管理指南*》中的 配置大页。

30 可选：应用于虚拟机的关联性组名称列表。oVirt 中必须存在关联性组。

> **注意**
>
> 因为 RHV 在创建虚拟机时使用模板，如果您没有为可选参数指定值，RHV 将使用模板中指定的参数值。

2.5.2. 创建机器集

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
```

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

2.6. 在 VSPHERE 上创建机器集

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

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. vSphere 上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 上运行的机器集，并创建标记为 **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。

```
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。

2.6.2. 机器集管理的最低所需的 vCenter 权限

要在 vCenter 上的 OpenShift Container Platform 集群中管理机器集，您必须使用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小 vCenter 角色和权限。

例 2.1. 机器集管理所需的最小 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**
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**

适用于角色的 vSphere 对象	必要时	所需的权限
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。

重要

在 OpenShift Container Platform 4.9 中，一些 CSI 驱动程序和功能是技术预览。如需更多信息，请参阅 *OpenShift Container Platform 支持的 CSI 驱动程序*。

下表详细介绍了机器集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 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。

其他资源

- 有关 CSI 驱动程序和功能支持的更多信息，请参阅 OpenShift Container Platform 支持的 CSI 驱动程序。

2.6.3. 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

2.6.4. 创建机器集

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
```

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

第 3 章 手动扩展机器集

您可以在机器集中添加或删除机器的实例。

注意

如果您需要在扩展之外修改机器集的各个方面，请参阅修改机器集。

3.1. 先决条件

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

重要

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. 手动扩展机器集

要在机器集中添加或删除机器实例，您可以手动扩展机器集。

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机器集。

先决条件

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

流程

1. 查看集群中的机器集：

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

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

2. 查看集群中的机器：

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

3. 在您要删除的机器上设置注解：

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

4. 进行 cordon 操作，排空您要删除的节点：

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

5. 扩展机器集：

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

或者：

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

提示

您还可以应用以下 YAML 来扩展机器集：

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

您可以扩展或缩减机器集。需要过几分钟以后新机器才可用。

验证

- 验证删除预期的机器：

```
$ oc get machines
```

3.3. 机器集删除策略

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

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

无论删除策略是什么，都可通过在相关机器上添加 **machine.openshift.io/cluster-api-delete-machine=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 机器集扩展为 **0**。

注意

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

第 4 章 修改机器集

您可以对机器集进行更改，例如添加标签、更改实例类型或更改块存储。

在 Red Hat Virtualization(RHV) 中，您还可以更改机器集，以便在不同的存储域中置备新节点。

注意

如果您需要在不进行其他更改的情况下扩展机器集，请参阅手动扩展机器集。

4.1. 修改机器集

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

如果您需要在不进行其他更改的情况下扩展机器集，则不需要删除机器。

注意

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

先决条件

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

流程

1. 编辑机器集：

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

2. 将机器缩减为 **0**:

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

或者：

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

提示

您还可以应用以下 YAML 来扩展机器集：

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

等待机器被删除。

3. 根据需要扩展机器设置：

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

或者：

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

提示

您还可以应用以下 YAML 来扩展机器集：

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

等待机器启动。新机器包含您对机器集所做的更改。

4.2. 将节点迁移到 RHV 上的不同存储域

您可以将 OpenShift Container Platform control plane 和计算节点迁移到 Red Hat Virtualization(RHV)集群中的不同存储域。

4.2.1. 将计算节点迁移到 RHV 中的不同存储域

先决条件

- 已登陆到 Manager。
- 有目标存储域的名称。

流程

1. 确定虚拟机模板：

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

2. 根据您确定的模板在 Manager 中创建新虚拟机。所有其他设置保持不变。详情请参阅 Red Hat Virtualization *虚拟机管理指南*中的基于模板创建虚拟机。

提示

您不需要启动新虚拟机。

3. 从新虚拟机创建一个新模板。在目标下指定 **目标** 存储域.详情请参阅 Red Hat Virtualization *虚拟机管理指南*中的创建模板。

4. 使用新模板添加新机器集到 OpenShift Container Platform 集群。

   a. 获取当前机器集的详情：

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

   b. 使用这些详细信息创建机器集。如需更多信息，请参阅 *创建机器集*。
   在 **template_name** 字段中输入新虚拟机模板名称。使用与 Manager 中的**新建模板**对话框中使用的相同模板名称。

   c. 请注意旧机器和新机器集的名称。您需要在后续步骤中引用它们。

5. 迁移工作负载。

   a. 扩展新计算机集。如需有关手动扩展机器集的详细信息，请参阅*手动扩展机器集*。
   当删除旧机器时，OpenShift Container Platform 会将 pod 移到可用的 worker 中。

   b. 缩减旧计算机集。

6. 删除旧的机器集：

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

其他资源

- 创建机器集。
- 手动扩展机器集
- 使用调度程序控制 pod 放置

4.2.2. 将 control plane 节点迁移到 RHV 上的不同存储域

OpenShift Container Platform 不管理 control plane 节点，因此它们比计算节点更容易迁移。就像 Red Hat Virtualization(RHV)上的任何其他虚拟机一样迁移它们。

对每个节点单独执行此步骤。

先决条件

- 已登陆到 Manager。
- 您已识别了 control plane 节点。在 Manager 中，它们被标记为 **master**。

流程

1. 选择标记为 **master** 的虚拟机。
2. 关闭虚拟机。
3. 点 **Disks** 选项卡。
4. 点虚拟机的磁盘。
5. 点 **More Actions** 并选择 **Move**。
6. 选择目标存储域并等待迁移过程完成。
7. 启动虚拟机。
8. 验证 OpenShift Container Platform 集群是否是稳定状态；

   ```
   $ oc get nodes
   ```

   输出应显示状态为 **Ready** 的节点。
9. 为每个 control plane 节点重复此步骤。

第 5 章 删除机器

您可以删除特定的机器。

5.1. 删除一个特定的机器

您可以删除特定的机器。

注意

您无法删除控制平面机器。

先决条件

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

流程

1. 查看集群中的机器，找到要删除的机器：

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

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

2. 删除机器：

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

重要

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

5.2. 其他资源

- 替换不健康的 etcd 成员。

第 6 章 将自动扩展应用到 OPENSHIFT CONTAINER PLATFORM 集群

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

> **重要**
>
> 您只能在机器 API 正常工作的集群中配置集群自动扩展。

6.1. 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：

- 节点上运行的所有 pod 的 CPU 和内存请求总和小于节点上分配资源的 50%。
- 集群自动扩展可以将节点上运行的所有 pod 移到其他节点。
- 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。

6.2. 关于机器自动扩展

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

重要

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

6.3. 配置集群自动扩展

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

注意

由于集群自动扩展的范围仅限于整个集群，因此只能为集群创建一个集群自动扩展。

6.3.1. ClusterAutoscaler 资源定义

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

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

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. 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**。

11 指定集群自动扩展是否可以删除不必要的节点。

12 （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**。

13 指定在最近*删除*节点之后要等待多久才能删除节点。如果不指定值，则使用默认值 **10s**。

14 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**。

15 指定要经过多长时间之后，不需要的节点才符合删除条件。如果不指定值，则使用默认值 **10m**。

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。

6.3.2. 部署集群自动扩展

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

流程

1. 为 **ClusterAutoscaler** 资源创建一个 YAML 文件，其中包含自定义的资源定义。

2. 在集群中创建资源：

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

1 **<filename>** 是您自定义的资源文件的名称。

6.4. 后续步骤

- 配置集群自动扩展后，必须至少配置一台机器自动扩展。

6.5. 配置机器自动扩展

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

重要

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

注意

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

6.5.1. MachineAutoscaler 资源定义

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

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

1. 指定机器自动扩展名称。为了更容易识别此机器自动扩展会扩展哪些机器集，请指定或注明要扩展的机器集的名称。机器集名称采用以下形式：**<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，集群自动扩展会将机器集缩减为零。

   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，提供用于描述要扩展的现有机器集的值。

5. **kind** 参数值始终为 **MachineSet**。

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

6.5.2. 部署机器自动扩展

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

流程

1. 为 **MachineAutoscaler** 资源创建一个 YAML 文件，其中包含自定义的资源定义。

2. 在集群中创建资源：

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

–

1. **<filename>** 是您自定义的资源文件的名称。

6.6. 其他资源

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

第 7 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。OpenShift Logging 和 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。

7.1. OPENSHIFT CONTAINER PLATFORM 基础架构组件

以下基础架构工作负载不会导致 OpenShift Container Platform worker 订阅：

- 在主控机上运行的 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
- 服务代理
- Red Hat Quay
- Red Hat OpenShift Container Storage
- Red Hat Advanced Cluster Manager
- Red Hat Advanced Cluster Security for Kubernetes
- Red Hat OpenShift GitOps
- Red Hat OpenShift Pipelines

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。

7.2. 为生产环境创建基础架构机器集

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。OpenShift Logging 和 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。这样的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。

7.2.1. 为不同云创建机器集

使用云的示例机器集。

7.2.1.1. AWS 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 **us-east-1a** Amazon Web Services（AWS）区域中运行的机器集，并创建通过 **node-role.kubernetes.io/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
      taints: 10
        - key: node-role.kubernetes.io/infra
          effect: NoSchedule
      providerSpec:
        value:
          ami:
```

```
        id: ami-046fe691f52a953f9 (11)
      apiVersion: awsproviderconfig.openshift.io/v1beta1
      blockDevices:
        - ebs:
            iops: 0
            volumeSize: 120
            volumeType: gp2
      credentialsSecret:
        name: aws-cloud-credentials
      deviceIndex: 0
      iamInstanceProfile:
        id: <infrastructure_id>-worker-profile (12)
      instanceType: m4.large
      kind: AWSMachineProviderConfig
      placement:
        availabilityZone: <zone> (13)
        region: <region> (14)
      securityGroups:
        - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-worker-sg (15)
      subnet:
        filters:
          - name: tag:Name
            values:
              - <infrastructure_id>-private-<zone> (16)
      tags:
        - name: kubernetes.io/cluster/<infrastructure_id> (17)
          value: owned
      userDataSecret:
        name: worker-user-data
```

1 3 5 12 15 17 指定基于置备集群时所设置的集群 ID 的基础架构 ID。如果已安装 OpenShift CLI，您可以通过运行以下命令来获取基础架构 ID：

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

2 4 8 指定基础架构 ID、**<infra>** 节点标签和区域。

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。

11 为您的 OpenShift Container Platform 节点的 AWS 区域指定有效的 Red Hat Enterprise Linux CoreOS (RHCOS) 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>-worker-<zone>
```

13 指定区域，如 **us-east-1a**。

14 指定区域，如 **us-east-1**。

16 指定基础架构 ID 和区域。

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。通过在 **MachineSet** YAML 文件中添加 **SpotMarketOptions** 来配置 Spot 实例。

7.2.1.2. Azure 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在区域(region)的 **1** Microsoft Azure 区域(zone)中运行的机器集，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
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image: 12
            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
    taints: 22
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
```

1 5 7 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 **<infra>** 节点标签。

4 6 10 指定基础架构 ID、**<infra>** 节点标签和地区。

12. 指定机器集的镜像详情。如果要使用 Azure Marketplace 镜像，请参阅"选择 Azure Marketplace 镜像"。

13. 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。

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。

22. 指定一个污点，以防止将用户工作负载调度到 infra 节点上。

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。您可以通过在 **MachineSet** YAML 文件中添加 **spotVMOptions** 来配置 Spot 虚拟机。

其他资源

- 选择 Azure Marketplace 镜像

7.2.1.3. GCP 上机器设置自定义资源的 YAML 示例

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

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-48-x86-64-202210040145**
- OpenShift Platform Plus: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-48-x86-64-202206140145**
- OpenShift Kubernetes Engine: **https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-48-x86-64-202206140145**

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。

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。您可以通过在 **MachineSet** YAML 文件中添加 **preemptible** 来配置抢占虚拟机实例。

7.2.1.4. RHOSP 上机器设置自定义资源的 YAML 示例

此 YAML 示例定义了一个在 Red Hat OpenStack Platform（RHOSP）上运行的机器集，并创建带有 **node-role.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: openstackproviderconfig.openshift.io/v1alpha1
        cloudName: openstack
        cloudsSecret:
          name: openstack-cloud-credentials
          namespace: openshift-machine-api
        flavor: <nova_flavor>
        image: <glance_image_name_or_location>
        serverGroupID: <optional_UUID_of_server_group> 12
        kind: OpenstackProviderSpec
        networks: 13
        - filter: {}
          subnets:
          - filter:
              name: <subnet_name>
              tags: openshiftClusterID=<infrastructure_id> 14
        primarySubnet: <rhosp_subnet_UUID> 15
        securityGroups:
        - filter: {}
          name: <infrastructure_id>-worker 16
        serverMetadata:
          Name: <infrastructure_id>-worker 17
          openshiftClusterID: <infrastructure_id> 18
        tags:
        - openshiftClusterID=<infrastructure_id> 19
        trunk: true
        userDataSecret:
          name: worker-user-data 20
        availabilityZone: <optional_openstack_availability_zone>
```

1 5 7 14 16 17 18 19 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。

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。

7.2.1.5. RHV 上机器集自定义资源的 YAML 示例

此 YAML 示例定义了一个在 RHV 上运行的机器集，并创建标记为 **node-role.kubernetes.io/<node_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。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    machine.openshift.io/cluster-api-machine-role: <role> 2
    machine.openshift.io/cluster-api-machine-type: <role> 3
  name: <infrastructure_id>-<role> 4
  namespace: openshift-machine-api
spec:
  replicas: <number_of_replicas> 5
  selector: 6
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 7
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 8
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 9
        machine.openshift.io/cluster-api-machine-role: <role> 10
        machine.openshift.io/cluster-api-machine-type: <role> 11
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role> 12
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/<role>: "" 13
      providerSpec:
        value:
          apiVersion: ovirtproviderconfig.machine.openshift.io/v1beta1
          cluster_id: <ovirt_cluster_id> 14
          template_name: <ovirt_template_name> 15
          instance_type_id: <instance_type_id> 16
          cpu: 17
            sockets: <number_of_sockets> 18
            cores: <number_of_cores> 19
            threads: <number_of_threads> 20
          memory_mb: <memory_size> 21
          os_disk: 22
            size_gb: <disk_size> 23
          network_interfaces: 24
            vnic_profile_id:  <vnic_profile_id> 25
          credentialsSecret:
            name: ovirt-credentials 26
          kind: OvirtMachineProviderSpec
```

```
    type: <workload_type> 27
    auto_pinning_policy: <auto_pinning_policy> 28
    hugepages: <hugepages> 29
    affinityGroupsNames:
      - compute 30
    userDataSecret:
      name: worker-user-data
```

1 7 9 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 10 11 13 指定要添加的节点标签。

4 8 12 指定基础架构 ID 和节点标签。这两个字符串不能超过 35 个字符。

5 指定要创建的机器数量。

6 机器的选择器。

14 指定此虚拟机实例所属的 RHV 集群的 UUID。

15 指定用于创建机器的 RHV VM 模板。

16 可选：指定虚拟机实例类型。

> **警告**
>
> **instance_type_id** 字段已弃用，并将在以后的发行版本中删除。

如果包含这个参数，则不需要指定包括 CPU 和内存在内的虚拟机的硬件参数，因为这个参数会覆盖所有硬件参数。

17 可选：CPU 字段包含 CPU 配置，包括插槽、内核和线程。

18 可选：指定虚拟机的插槽数量。

19 可选：指定每个插槽的内核数。

20 可选：指定每个内核的线程数量。

21 可选：指定虚拟机的内存大小（MiB）。

22 可选：节点的 Root 磁盘。

23 可选：指定可引导磁盘的大小（GiB）。

24 可选：虚拟机网络接口列表。如果包含此参数，OpenShift Container Platform 会丢弃来自模板中的所有网络接口并创建新接口。

25 可选：指定 vNIC 配置集 ID。

26 指定包含 RHV 凭证的 secret 名称。

27 可选：指定实例优化的工作负载类型。这个值会影响 **RHV VM** 参数。支持的值包括：**desktop**、**server**（默认）、**high_performance**。**high_performance** 提高了虚拟机的性能，但存在一些限制。例如，您无法使用图形控制台访问虚拟机。如需更多信息，请参阅*虚拟机管理指南*中的配置高性能虚拟机、模板和池。

28 可选： AutoPinningPolicy 定义自动设置 CPU 和 NUMA 设置的策略，包括固定到此实例的主机。支持的值： **none**，**resize_and_pin**。如需更多信息，请参阅《*虚拟机管理指南*》中的 设置 NUMA 节点。

29 可选：Helgepages 是以 KiB 为单位的大小，用于在虚拟机中定义巨页。支持的值： **2048** 或 **1048576**。如需更多信息，请参阅《*虚拟机管理指南*》中的 配置大页。

30 可选：应用于虚拟机的关联性组名称列表。oVirt 中必须存在关联性组。

注意

因为 RHV 在创建虚拟机时使用模板，如果您没有为可选参数指定值，RHV 将使用模板中指定的参数值。

7.2.1.6. vSphere 上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。

11 指定要将机器集部署到的 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 指定要将机器集部署到的 vCenter Datacenter。

14 指定要部署机器集的 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。

7.2.2. 创建机器集

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
```

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

7.2.3. 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: topology.kubernetes.io/region=us-east-1 1
...
```

1 默认情况下，此节点选择器示例将容器集部署到 **us-east-1** 区域的节点。

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

7.2.4. 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。

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。

7.3. 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。

7.3.1. 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: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 **NoSchedule**。具有 **NoSchedule effect** 的节点仅调度容许该污点的 pod，但允许现有 pod 继续调度到该节点上。

注意

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

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

```
tolerations:
  - effect: NoExecute 1
    key: node-role.kubernetes.io/infra 2
    operator: Exists 3
    value: reserved 4
```

1 指定添加到节点的效果。

2 指定添加到节点的键。

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

4 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。

7.4. 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。

7.4.1. 移动路由器

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

先决条件

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

流程

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

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

命令输出类似于以下文本：

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

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

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

```
  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   v1.22.1
```

由于角色列表包含 **infra**，因此 Pod 在正确的节点上运行。

7.4.2. 移动默认 registry

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

先决条件

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

流程

1. 查看 **config/instance** 对象：

```
$ oc get 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。

7.4.3. 移动监控解决方案

监控堆栈包含多个组件，包括 Prometheus、Grafana 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
    grafana:
      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
k8sPrometheusAdapter:
  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
```

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。

7.4.4. 移动 OpenShift Logging 资源

您可以配置 Cluster Logging Operator，以将用于日志记录子系统组件的 Pod（如 Elasticsearch 和 Kibana）部署到不同的节点上。您无法将 Cluster Logging Operator Pod 从其安装位置移走。

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

先决条件

- 必须安装 Red Hat OpenShift Logging 和 Elasticsearch Operator。默认情况下没有安装这些功能。

流程

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

```
$ oc edit ClusterLogging instance
```

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

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  logStore:
    elasticsearch:
      nodeCount: 3
      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
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: (2)
        node-role.kubernetes.io/infra: ''
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

...
```

1 2 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。

验证

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

例如：

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

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

输出示例

```
NAME                      READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
kibana-5b8bdf44f9-ccpq9   2/2     Running   0          27s   10.129.2.18   ip-10-0-147-79.us-
east-2.compute.internal   <none>           <none>
```

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

```
$ oc get nodes
```

输出示例

```
NAME                                       STATUS   ROLES      AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master       60m   v1.22.1
ip-10-0-139-146.us-east-2.compute.internal   Ready    master       60m   v1.22.1
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker       51m   v1.22.1
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker       51m   v1.22.1
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker       51m   v1.22.1
ip-10-0-152-241.us-east-2.compute.internal   Ready    master       60m   v1.22.1
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra       51m   v1.22.1
```

请注意，该节点具有 **node-role.kubernetes.io/infra: ''** label:

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

输出示例

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: ''
...
```

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

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

...

spec:

...

  visualization:
    kibana:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1. 添加节点选择器以匹配节点规格中的 label。

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

```
$ oc get pods
```

输出示例

```
NAME                                           READY   STATUS        RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7        1/1     Running        0          29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running        0          28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running        0          28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running        0          28m
fluentd-42dzz                                   1/1     Running        0          28m
fluentd-d74rq                                   1/1     Running        0          28m
fluentd-m5vr9                                   1/1     Running        0          28m
fluentd-nkxl7                                   1/1     Running        0          28m
fluentd-pdvqb                                   1/1     Running        0          28m
fluentd-tflh6                                   1/1     Running        0          28m
kibana-5b8bdf44f9-ccpq9                         2/2     Terminating    0          4m11s
kibana-7d85dcffc8-bfpfp                         2/2     Running        0          33s
```

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

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
```

输出示例

```
NAME                      READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
kibana-7d85dcffc8-bfpfp   2/2     Running   0          43s   10.131.0.22   ip-10-0-139-48.us-
east-2.compute.internal   <none>           <none>
```

- 片刻后，原始 Kibana Pod 将被删除。

```
$ oc get pods
```

输出示例

```
NAME                                           READY   STATUS    RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7        1/1     Running   0          30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running   0          29m
fluentd-42dzz                                   1/1     Running   0          29m
fluentd-d74rq                                   1/1     Running   0          29m
fluentd-m5vr9                                   1/1     Running   0          29m
fluentd-nkxl7                                   1/1     Running   0          29m
fluentd-pdvqb                                   1/1     Running   0          29m
fluentd-tflh6                                   1/1     Running   0          29m
kibana-7d85dcffc8-bfpfp                         2/2     Running   0          62s
```

其他资源

- 有关移动 OpenShift Container Platform 组件的常规说明，请参阅监控文档。

第 8 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）计算或 worker 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。计算机器上只能使用 RHEL 作为操作系统。

8.1. 关于在集群中添加 RHEL 计算节点

在 OpenShift Container Platform 4.9 中，如果使用用户置备的基础架构安装，您可以选择将 Red Hat Enterprise Linux (RHEL) 机器用作集群中的计算机器（也称为 worker）。集群中的 control plane 或主控机器（master）必须使用 Red Hat Enterprise Linux CoreOS (RHCOS)。

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

重要

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

重要

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

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

8.2. RHEL 计算节点的系统要求

OpenShift Container Platform 环境中的 Red Hat Enterprise Linux (RHEL) 计算或 worker 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 7.9 或 RHEL 7.9 到 8.7。

重要

在 OpenShift Container Platform 集群中添加 RHEL 7 计算机器已被弃用。弃用的功能仍然包含在 OpenShift Container Platform 中，并将继续被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

另外，您无法将 RHEL 7 计算机器升级到 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 7 文档中的启用 FIPS 模式 。

重要

只有在 **x86_64** 架构中的 OpenShift Container Platform 部署支持 FIPS 验证的/Modules in Process 加密库。

- 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

8.2.1. 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。

8.3. 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。

8.3.1. 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。

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。

8.4. 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 4.9 集群之前，必须准备一个 RHEL 7 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 7 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 4.9 所需的存储库：

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms" \
    --enable="rhel-7-server-ose-4.9-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。

8.5. 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 4.9 需要的存储库。

   a. 对于 RHEL 7 节点，您必须启用以下软件仓库：

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-optional-rpms" \
    --enable="rhel-7-server-ose-4.9-rpms"
```

注意

RHEL 7 节点的使用已弃用，并计划在以后的 OpenShift Container Platform 4 发行版本中删除。

   b. 对于 RHEL 8 节点，您必须启用以下软件仓库：

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.9-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。

8.6. 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 角色的 AWS 权限。

8.7. 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。

8.8. 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 4.9 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。

8.9. 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  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

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  v1.22.1
   master-1  Ready     master  73m  v1.22.1
   master-2  Ready     master  74m  v1.22.1
   worker-0  Ready     worker  11m  v1.22.1
   worker-1  Ready     worker  11m  v1.22.1
   ```

   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。

8.10. 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**。

参数	描述	值
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。

8.10.1. 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 9 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。

9.1. 关于在集群中添加 RHEL 计算节点

在 OpenShift Container Platform 4.9 中，如果使用用户置备的基础架构安装，您可以选择将 Red Hat Enterprise Linux (RHEL) 机器用作集群中的计算机器（也称为 worker）。集群中的 control plane 或主控机器（master）必须使用 Red Hat Enterprise Linux CoreOS (RHCOS)。

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

重要

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

重要

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

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

9.2. RHEL 计算节点的系统要求

OpenShift Container Platform 环境中的 Red Hat Enterprise Linux (RHEL) 计算或 worker 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 7.9 或 RHEL 7.9 到 8.7。

重要

在 OpenShift Container Platform 集群中添加 RHEL 7 计算机器已被弃用。弃用的功能仍然包含在 OpenShift Container Platform 中，并将继续被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

另外，您无法将 RHEL 7 计算机器升级到 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 7 文档中的启用 FIPS 模式 。

重要

只有在 **x86_64** 架构中的 OpenShift Container Platform 部署支持 FIPS 验证的/Modules in Process 加密库。

- 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。您必须列出 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。

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 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 4.9 需要的存储库。

   a. 对于 RHEL 7 节点，您必须启用以下软件仓库：

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-optional-rpms" \
    --enable="rhel-7-server-ose-4.9-rpms"
```

   注意

   RHEL 7 节点的使用已弃用，并计划在以后的 OpenShift Container Platform 4 发行版本中删除。

   b. 对于 RHEL 8 节点，您必须启用以下软件仓库：

```
# subscription-manager repos \
    --enable="rhel-8-for-x86_64-baseos-rpms" \
    --enable="rhel-8-for-x86_64-appstream-rpms" \
    --enable="rhocp-4.9-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。

9.5. 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 角色的 AWS 权限。

9.6. 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。

9.7. 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 4.9 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。

9.8. 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  v1.22.1
master-1  Ready     master  63m  v1.22.1
master-2  Ready     master  64m  v1.22.1
```

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  v1.22.1
   master-1  Ready     master  73m  v1.22.1
   master-2  Ready     master  74m  v1.22.1
   worker-0  Ready     worker  11m  v1.22.1
   worker-1  Ready     worker  11m  v1.22.1
   ```

   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。

9.9. 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 10 章 手动管理用户置备基础架构

10.1. 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。

10.1.1. 将计算机器添加到 Amazon Web Services

要将更多计算机器添加到 Amazon Web Services (AWS) 上的 OpenShift Container Platform 集群，请参阅使用 CloudFormation 模板将计算机器添加到 AWS。

10.1.2. 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。

10.1.3. 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。

10.1.4. 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。

10.1.5. 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。

10.1.6. 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，请参阅在裸机中添加计算机器。

10.2. 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。

10.2.1. 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。

10.2.2. 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。

10.2.3. 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  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

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  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

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。

10.3. 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。

10.3.1. 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，需要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。

10.3.2. 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。

先决条件

- 获取计算机器的 base64 编码 Ignition 文件。
- 您可以访问您为集群创建的 vSphere 模板。

流程

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

10.3.3. 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  v1.22.1
```

```
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

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  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

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。

10.4. 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。

10.4.1. 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，需要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。

10.4.2. 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。

10.4.2.1. 使用 ISO 镜像创建更多 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。

流程

1. 使用 ISO 文件在更多计算机器上安装 RHCOS。在安装集群前，使用创建机器时使用的相同方法：
   - 将 ISO 镜像刻录到磁盘并直接启动。

- 在 LOM 接口中使用 ISO 重定向。

2. 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。

3. 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
```

4. 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。

5. 继续为集群创建更多计算机器。

10.4.2.2. 通过 PXE 或 iPXE 启动来创建更多 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。

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：

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.inst.install_dev=/dev/sda coreos.inst.ignition_url=http://<HTTP_server>/worker.ign
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.<architecture>.img
(1)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
(2)
```

(1) 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.inst.ignition_url** 参数值是 worker Ignition 配置文件的位置，**coreos.live.rootfs_url** 参数值则是 live **rootfs** 文件的位置。**coreos.inst.ignition_url** 和 **coreos.live.rootfs_url** 参数仅支持 HTTP 和 HTTPS。

2 指定上传到 HTTP 服务器的 **initramfs** 文件的位置。

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 **内核参数** 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？

1. 使用 PXE 或 iPXE 基础架构为集群创建所需的计算机器。

10.4.3. 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  v1.22.1
master-1  Ready    master  63m  v1.22.1
master-2  Ready    master  64m  v1.22.1
```

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  v1.22.1
master-1  Ready    master  73m  v1.22.1
master-2  Ready    master  74m  v1.22.1
worker-0  Ready    worker  11m  v1.22.1
worker-1  Ready    worker  11m  v1.22.1
```

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 11 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}'
```

11.1. 关于机器健康检查

机器健康检查自动修复特定机器池中不健康的机器。

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。

注意

您不能对具有 master 角色的机器进行机器健康检查。

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。

11.1.1. 部署机器健康检查时的限制

部署机器健康检查前需要考虑以下限制：

- 只有机器集拥有的机器才可以由机器健康检查修复。
- 目前不支持 control plane 机器，如果不健康，则不会被修复。
- 如果机器的节点从集群中移除，机器健康检查会认为机器不健康，并立即修复机器。

- 如果机器对应的节点在 **nodeStartupTimeout** 之后没有加入集群，则会修复机器。
- 如果 **Machine** 资源阶段为 **Failed**，则会立即修复机器。

其他资源

- 有关您可以在 **MachineHealthCheck** CR 中定义的节点状况的更多信息，请参阅关于列出集群中的所有节点。
- 有关短路的更多信息，请参阅 Short-circuiting 机器健康检查补救。

11.2. 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。

11.2.1. 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机器集，以便在丢失整个区时，**maxUnhealthy** 设置可以在集群中阻止进一步补救。

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。

11.2.1.1. 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。

11.2.1.2. 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。

11.3. 创建 MACHINEHEALTHCHECK 资源

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

先决条件

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。

11.4. 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
- 在超聚合环境中减少了数据丢失的风险
- 减少了因为恢复物理机器造成的停机时间

11.4.1. 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。要将默认补救过程从机器删除到主机的节能周期，请使用 **machine.openshift.io/remediation-strategy: external-baremetal** 注解来注解 **MachineHealthCheck** 资源。

设置注解后，不健康的机器会使用 BMC 凭证进行节能。

11.4.2. 了解补救过程

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。

11.4.3. 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。
- 访问基板管理控制器（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

```
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

3 4 为要检查的机器池指定一个标签。

5 以 **<cluster_name>-<label>-<zone>** 格式 指定要跟踪的机器集。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。

<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>