OpenShift Container Platform 4.16

安装配置

在安装过程中进行集群范围的配置

Last Updated: 2025-10-14

OpenShift Container Platform 4.16 安装配置

在安装过程中进行集群范围的配置

Legal Notice

Abstract

本文档论述了如何执行初始 OpenShift Container Platform 集群配置。

Table of Contents

第 1 章 自定义节点

OpenShift Container Platform 通过 Ignition 支持集群范围和每机器配置，允许对操作系统进行任意分区和文件内容更改。通常，如果在 Red Hat Enterprise Linux (RHEL) 中记录了配置文件，则支持通过 Ignition 修改它。

部署机器配置更改的方法有两种：

- 创建包含在清单文件中的机器配置，以便在 **openshift-install** 期间启动集群。
- 创建通过 Machine Config Operator 传递给运行的 OpenShift Container Platform 节点的机器配置。

另外，修改引用配置，比如在安装裸机节点时传递给 **coreos-installer** 的 Ignition 配置允许每个机器配置。这些更改目前对 Machine Config Operator 不可见。

以下小节描述了您可能需要以这种方式在节点上配置的功能。

1.1. 使用 BUTANE 创建机器配置

机器配置用于配置 control plane 和 worker 机器，方法是指示机器如何创建用户和文件系统、设置网络、安装 systemd 单元等。

因为修改机器配置可能比较困难，所以您可以使用 Butane 配置为您创建机器配置，从而使节点配置更容易。

1.1.1. 关于 Butane

但ane 是一个命令行实用程序，OpenShift Container Platform 使用它为编写机器配置提供便捷的简写语法，并对机器配置进行额外的验证。Butane 接受的 Butane 配置文件的格式在 OpenShift Butane 配置规格中定义。

1.1.2. 安装 Butane

您可以安装 Butane 工具（**但ane**），以便使用命令行界面创建 OpenShift Container Platform 机器配置。您可以通过下载对应的二进制文件，**在 Linux、Windows 或 macOS 上安装但可** 正常工作。

提示

但ane 版本与旧版本以及 Fedora CoreOS Config Transpiler(FCCT)向后兼容。

流程

1. 导航到位于 https://mirror.openshift.com/pub/openshift-v4/clients/butane/ 的 Butane 映像下载页面。
2. 获取 **withane** 二进制文件：
   a. 对于 Butane 的最新版本，请将最新的 **但ane** 镜像保存到当前目录中：

```
$ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane --output butane
```

b. 可选：对于您要在其中安装的特定类型的架构，如 aarch64 或 ppc64le，代表适当的 URL。例如：

```
$ curl https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane-aarch64 --output butane
```

3. 使下载的二进制文件可执行：

```
$ chmod +x butane
```

4. 将 **-ane** 二进制文件移到 **PATH 上的目录中**。
要查看您的 **PATH**，请打开终端并执行以下命令：

```
$ echo $PATH
```

验证步骤

- 现在，您可以通过运行 butane 命令使用 But **ane** 工具：

```
$ butane <butane_file>
```

1.1.3. 使用 Butane 创建 MachineConfig 对象

您可以使用 Butane 生成 **MachineConfig** 对象，以便在安装时或通过 Machine Config Operator 配置 worker 或 control plane 节点。

先决条件

- 您已安装了 with **ane** 实用程序。

流程

1. 创建一个 Butane 配置文件。以下示例创建一个名为 **99-worker-custom.bu** 的文件，该文件将系统控制台配置为显示内核调试信息并为 chrony 时间服务指定自定义设置：

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-worker-custom
  labels:
    machineconfiguration.openshift.io/role: worker
openshift:
  kernel_arguments:
    - loglevel=7
storage:
  files:
    - path: /etc/chrony.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          pool 0.rhel.pool.ntp.org iburst
          driftfile /var/lib/chrony/drift
```

```
makestep 1.0 3
rtcsync
logdir /var/log/chrony
```

注意

99-worker-custom.bu 文件设置为为 worker 节点创建机器配置。要在 control plane 节点上部署，请将角色从 **worker** 改为 **master**。要进行这两个操作，您可以在两种部署中使用不同的文件名来重复整个过程。

2. 通过提供您在上一步中创建的文件，创建 **MachineConfig** 对象：

```
$ butane 99-worker-custom.bu -o ./99-worker-custom.yaml
```

已为您创建一个 **MachineConfig** 对象 YAML 文件，以完成机器的配置。

3. 如果将来需要更新 **MachineConfig** 对象，请保存 Butane 配置。

4. 如果集群还没有运行，生成清单文件并将 **MachineConfig** 对象 YAML 文件添加到 **openshift** 目录中。如果集群已在运行，按如下所示应用该文件：

```
$ oc create -f 99-worker-custom.yaml
```

其他资源

- 在节点中添加内核模块
- 在安装过程中加密和镜像磁盘

1.2. 添加 DAY-1 内核参数

虽然修改内核参数通常应做为第 2 天的任务，但您可能希望在初始集群安装过程中将内核参数添加到所有 master 节点或 worker 节点中。以下是您可能希望在集群安装过程中添加内核参数以便在系统第一次引导前生效的一些原因：

- 您需要在系统启动前进行一些低级网络配置。
- 您希望禁用某个功能，如 SELinux，因此在系统首次启动时不会受到影响。

警告

不支持在生产环境中禁用 RHCOS 上的 SELinux。在节点上禁用 SELinux 后，必须在生产集群中重新设置前重新置备它。

要在 master 节点或 worker 节点中添加内核参数，您可以创建一个 **MachineConfig** 对象，并将该对象注入 Ignition 在集群设置过程中使用的清单文件集合中。

有关您可以在引导时传递给 RHEL 8 内核的参数列表，请参阅 Kernel.org 内核参数。如果需要参数来完成初始 OpenShift Container Platform 安装，最好使用此流程添加内核参数。

流程

1. 进入包含安装程序的目录，并为集群生成 Kubernetes 清单：

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. 决定您要将内核参数添加到 worker 节点还是 control plane 节点。

3. 在 **openshift** 目录中，创建一个文件（例如： **99-openshift-machineconfig-master-kargs.yaml**）来定义 **MachineConfig** 对象以添加内核设置。这个示例在 control plane 节点中添加了一个 **loglevel=7** 内核参数：

```
$ cat << EOF > 99-openshift-machineconfig-master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
EOF
```

   您可以将 **master** 改为 **worker**，将内核参数添加到 worker 节点。创建单独的 YAML 文件以添加到 master 和 worker 节点。

现在，您可以继续创建集群。

1.3. 在节点中添加内核模块

对于大多数常见硬件，Linux 内核包含了计算机启动时使用该硬件所需的设备驱动程序模块。然而，对于某些硬件，Linux 中无法使用模块。因此，您必须找到一种方法来为每个主机计算机提供这些模块。此流程描述了如何为 OpenShift Container Platform 集群中的节点执行此操作。

当首先按照这些说明部署内核模块时，该模块将提供给当前内核。如果安装了新内核，kmods-via-containers 软件将重建并部署该模块，以便新内核可以使用该模块的兼容版本。

使这个功能能够在每个节点中保持模块最新的方法是：

- 在引导时启动的每个节点中添加 systemd 服务，以检测是否安装了新内核。
- 如果检测到新内核，该服务会重建该模块并将其安装到内核中

有关此流程所需软件的详情，请查看 kmods-via-containers github 站点。

需要记住的几个重要问题：

- 这个过程是技术预览。
- 软件工具和示例还没有官方的 RPM，现只能从非官方的 **github.com** 站点获得。

- 红帽不支持您通过这些步骤添加的第三方内核模块。
- 在此过程中，构建内核模块所需的软件部署在 RHEL 8 容器中。请记住，当节点有新内核时，每个节点上会自动重新构建模块。因此，每个节点都需要访问 **yum** 存储库，该存储库包含重建该模块所需的内核和相关软件包。该内容最好由有效的 RHEL 订阅提供。

1.3.1. 构建和测试内核模块容器

在将内核模块部署到 OpenShift Container Platform 集群之前，您可以在单独的 RHEL 系统上测试该过程。收集内核模块的源代码、KVC 框架和 kmod-via-containers 软件。然后构建并测试模块。要在 RHEL 8 系统中做到这一点，请执行以下操作：

流程

1. 注册 RHEL 8 系统：

```
# subscription-manager register
```

2. 为 RHEL 8 系统附加订阅：

```
# subscription-manager attach --auto
```

3. 安装构建软件和容器所需的软件：

```
# yum install podman make git -y
```

4. 克隆 **kmod-via-containers** 存储库：
   a. 为存储库创建一个文件夹：

```
$ mkdir kmods; cd kmods
```

   b. 克隆存储库：

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

5. 在 RHEL 8 构建主机上安装 KVC 框架实例来测试模块。这会添加 **kmods-via-container** systemd 服务并加载它：
   a. 进入 **kmod-via-containers** 目录：

```
$ cd kmods-via-containers/
```

   b. 安装 KVC 框架实例：

```
$ sudo make install
```

   c. 重新载入 systemd Manager 配置：

```
$ sudo systemctl daemon-reload
```

6. 获取内核模块源代码。源代码可用于构建您无法控制但由其他人提供的第三方模块。您需要类似 **kvc-simple-kmod** 示例中显示的内容，该示例可克隆到您的系统中，如下所示：

```
$ cd .. ; git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

7. 编辑本例中的配置文件 **simple-kmod.conf**，并将 Dockerfile 的名称改为 **Dockerfile.rhel** ：

   a. 进入 **kvc-simple-kmod** 目录：

```
$ cd kvc-simple-kmod
```

   b. 重命名 Dockerfile:

```
$ cat simple-kmod.conf
```

   Dockerfile 示例

```
KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-simple-kmod.git"
KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

8. 为您的内核模块创建一个 **kmods-via-containers@.service** 实例，**本例中为 simple-kmod** ：

```
$ sudo make install
```

9. 启用 **kmods-via-containers@.service** 实例：

```
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

10. 启用并启动 systemd 服务：

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service --now
```

   a. 查看服务状态：

```
$ sudo systemctl status kmods-via-containers@simple-kmod.service
```

   输出示例

```
● kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
  Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
        enabled; vendor preset: disabled)
  Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

11. 要确认载入了内核模块，使用 **lsmod** 命令列出模块：

```
$ lsmod | grep simple_
```

输出示例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

12. 可选。使用其他方法检查 **simple-kmod** 是否正常工作：

- 使用 **dmesg** 在内核环缓冲中查找 "Hello world" 信息：

```
$ dmesg | grep 'Hello world'
```

输出示例

```
[ 6420.761332] Hello world from simple_kmod.
```

- 检查 **/proc** 中的 **simple-procfs-kmod** 值：

```
$ sudo cat /proc/simple-procfs-kmod
```

输出示例

```
simple-procfs-kmod number = 0
```

- 运行 **spkut** 命令从模块中获取更多信息：

```
$ sudo spkut 44
```

输出示例

```
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
+ podman run -i --rm --privileged
  simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

下一步，当系统引导此服务时，将检查新内核是否在运行。如果有新的内核，该服务会构建内核模块的新版本，然后载入它。如果已经构建了该模块，它将只加载它。

1.3.2. 为 OpenShift Container Platform 置备内核模块

根据 OpenShift Container Platform 集群首次引导时是否必须存在内核模块，您可以使用以下两种方式之一设置内核模块部署：

- 在 **集群安装时(day-1)置备内核模块** ：您可以通过一个 **MachineConfig** 对象创建内容，并通过包括一组清单文件来将其提供给 **openshift-install**。

- **通过 Machine Config Operator(day-2)置备内核模块：** 如果您可以等到集群启动并运行后再添加内核模块，您可以通过 Machine Config Operator(MCO)部署内核模块软件。

在这两种情况下，每个节点都需要能够在检测到新内核时获取内核软件包和相关软件包。您可以通过几种方法设置每个节点来获取该内容。

- 为每个节点提供 RHEL 权利。

- 从现有 RHEL 主机获取 RHEL 权利，从 **/etc/pki/entitlement** 目录中获取，并将它们复制到与您构建 Ignition 配置时提供的其他文件相同的位置。
- 在 Dockerfile 中，添加指针到包含内核和其他软件包的 **yum** 存储库。这必须包括新内核包，因为它们需要与新安装的内核相匹配。

1.3.2.1. 通过 MachineConfig 对象置备内核模块

通过将内核模块软件与 **MachineConfig** 对象一起打包，您可以在安装时或通过 Machine Config Operator 向 worker 或 control plane 节点提供该软件。

流程

1. 注册 RHEL 8 系统：

```
# subscription-manager register
```

2. 为 RHEL 8 系统附加订阅：

```
# subscription-manager attach --auto
```

3. 安装构建软件所需的软件：

```
# yum install podman make git -y
```

4. 创建托管内核模块和工具的目录：

```
$ mkdir kmods; cd kmods
```

5. 获取 **kmods-via-containers** 软件：

   a. 克隆 **kmods-via-containers** 存储库：

   ```
   $ git clone https://github.com/kmods-via-containers/kmods-via-containers
   ```

   b. 克隆 **kvc-simple-kmod** 存储库：

   ```
   $ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
   ```

6. 获取您的模块软件。本例中使用 **kvc-simple-kmod**。

7. 使用之前克隆的存储库，创建一个 fakeroot 目录，并在其中填充您要通过 Ignition 提供的文件：

   a. 创建目录：

   ```
   $ FAKEROOT=$(mktemp -d)
   ```

   b. 进入 **kmod-via-containers** 目录：

   ```
   $ cd kmods-via-containers
   ```

   c. 安装 KVC 框架实例：

```
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

d. 进入 **kvc-simple-kmod** 目录：

```
$ cd ../kvc-simple-kmod
```

e. 创建实例：

```
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

8. 运行以下命令，克隆 fakeroot 目录，将任何符号链接替换为目标副本：

```
$ cd .. && rm -rf kmod-tree && cp -Lpr ${FAKEROOT} kmod-tree
```

9. 创建一个 Butane 配置文件 **99-simple-kmod.bu**，它嵌入内核模块树并启用 systemd 服务。

注意

如需有关 Butane 的信息，请参阅"使用 Butane 创建机器配置"。

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-simple-kmod
  labels:
    machineconfiguration.openshift.io/role: worker 1
storage:
  trees:
    - local: kmod-tree
systemd:
  units:
    - name: kmods-via-containers@simple-kmod.service
      enabled: true
```

1 要在 control plane 节点上部署，请将 **worker** 改为 **master**。要在 control plane 和 worker 节点上部署，请对每个节点类型执行一次这些指令的其余部分。

10. 使用 Butane 生成机器配置 YAML 文件 **99-simple-kmod.yaml**，其中包含要交付的文件和配置：

```
$ butane 99-simple-kmod.bu --files-dir . -o 99-simple-kmod.yaml
```

11. 如果集群还没有启动，生成清单文件并将该文件添加到 **openshift** 目录中。如果集群已在运行，按如下所示应用该文件：

```
$ oc create -f 99-simple-kmod.yaml
```

您的节点将启动 **kmods-via-containers@simple-kmod.service** 服务，并将载入内核模块。

12. 要确认内核模块已加载，您可以登录到节点（使用 **oc debug node/<openshift-node>**，然后 **chroot /host**）。要列出模块，请使用 **lsmod** 命令：

```
$ lsmod | grep simple_
```

输出示例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

1.4. 在安装过程中加密和镜像磁盘

在 OpenShift Container Platform 安装过程中，您可以在集群节点上启用引导磁盘加密和镜像功能。

1.4.1. 关于磁盘加密

您可以在安装时在 control plane 和计算节点上为引导磁盘启用加密。OpenShift Container Platform 支持 Trusted Platform 模块(TPM)v2 和 Tang 加密模式。

TPM v2

这是首选模式。TPM v2 将密码短语存储在服务器的安全加密处理器中。如果从服务器中删除磁盘，您可以使用此模式来防止在集群节点上解密引导磁盘数据。

tang

Tang 和 Clevis 是启用网络绑定磁盘加密(NBDE)的服务器和客户端组件。您可以将集群节点中的引导磁盘数据绑定到一个或多个 Tang 服务器。这会防止解密数据，除非节点位于可访问 Tang 服务器的安全网络中。Clevis 是一种自动化解密框架，用于在客户端中实施解密。

重要

使用 Tang 加密模式加密磁盘只支持在用户置备的基础架构上安装裸机和 vSphere。

在以前的 Red Hat Enterprise Linux CoreOS(RHCOS)版本中，磁盘加密是通过在 Ignition 配置中指定 **/etc/clevis.json** 来配置的。使用 OpenShift Container Platform 4.7 或更高版本创建的集群不支持该文件。使用以下步骤配置磁盘加密。

启用 TPM v2 或 Tang 加密模式时，RHCOS 引导磁盘将使用 LUKS2 格式进行加密。

这个功能：

- 可用于安装程序置备的基础架构、用户置备的基础架构和辅助安装程序部署
- 对于辅助安装程序部署：
  - 每个集群只能有一个加密方法 Tang 或 TPM
  - 加密可以在某些或所有节点上启用
  - 没有 Tang 阈值 ; 所有服务器都必须有效且可操作
  - 加密只适用于安装磁盘，不适用于工作负载磁盘
- 只在 Red Hat Enterprise Linux CoreOS(RHCOS)系统上支持
- 在清单安装过程中设置磁盘加密，加密写入磁盘的所有数据
- 不需要用户干预来提供密码短语

- 如果启用了 FIPS 模式，则使用 AES-256-XTS 加密或 AES-256-CBC

1.4.1.1. 配置加密阈值

在 OpenShift Container Platform 中，您可以指定多个 Tang 服务器的要求。您还可以同时配置 TPM v2 和 Tang 加密模式。这只有在存在 TPM 安全加密处理器且 Tang 服务器可以通过安全网络访问时启用引导磁盘数据解密。

您可以使用 Butane 配置中的 **threshold** 属性来定义解密所需的 TPM v2 和 Tang 加密条件的最小数量。

通过声明的条件的任意组合达到声明的值时，会满足阈值。如果是离线调配，使用包含的公告访问离线服务器，并且仅在在线服务器数量没有满足集合阈值时使用该广告。

例如，在以下的配置中，**threshold** 的值为 **2**，这在访问带有可用的离线服务器作为备份的两个 Tang 服务器时就达到了；或访问 TPM 安全加密处理器和一个 Tang 服务器达到：

用于磁盘加密的 Butane 配置示例

```
variant: openshift
version: 4.16.0
metadata:
  name: worker-storage
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  layout: x86_64 (1)
  luks:
    tpm2: true (2)
    tang: (3)
      - url: http://tang1.example.com:7500
        thumbprint: jwGN5tRFK-kF6pIX89ssF3khxxX
      - url: http://tang2.example.com:7500
        thumbprint: VCJsvZFjBSIHSldw78rOrq7h2ZF
      - url: http://tang3.example.com:7500
        thumbprint: PLjNyRdGw03zlRoGjQYMahSZGu9
        advertisement: "{\"payload\": \"...\", \"protected\": \"...\", \"signature\": \"...\"}" (4)
    threshold: 2 (5)
openshift:
  fips: true
```

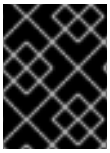
(1) 将此字段设置为集群节点的指令集合架构。一些示例包括、**x86_64**、**aarch64** 或 **ppc64le**。

(2) 如果要使用受信任的平台模块(TPM)加密根文件系统，请包含此字段。

(3) 如果要使用一个或多个 Tang 服务器，请包含此部分。

(4) 可选：包含用于离线置备的此字段。Ignition 将置备 Tang 服务器绑定，而不是在运行时从服务器获取公告。这样，服务器在置备时不可用。

(5) 指定进行解密所需的最小 TPM v2 和 Tang 加密条件。



重要

默认 **阈值** 为 **1**。如果您在配置中包含多个加密条件，但没有指定阈值，则会在满足任何条件时进行解密。

注意

如果需要 TPM v2 *和* Tang 进行，则 **threshold** 属性的值必须等于声明的 Tang 服务器总数再加一。如果阈值较低，可以使用单一加密模式达到阈值。例如，如果您将 **tpm2** 设置为 **true** 并指定两个 Tang 服务器，则可以通过访问两个 Tang 服务器来满足阈值 **2**，即使 TPM 安全加密处理器不可用。

1.4.2. 关于磁盘镜像

在 control plane 和 worker 节点上安装 OpenShift Container Platform 时，您可以将引导和其他磁盘镜像到两个或者多个冗余存储设备。存储设备失败后节点将继续正常工作，提供一个设备仍然可用。

镜像不支持替换失败的磁盘。重新置备节点，将镜像恢复到正常的非降级状态。

注意

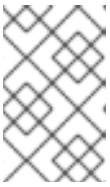
对于用户置备的基础架构部署，镜像只在 RHCOS 系统上可用。使用 BIOS 或 UEFI 和 **ppc64le** 节点上引导的 **x86_64** 节点上支持镜像。

1.4.3. 配置磁盘加密和镜像

您可以在 OpenShift Container Platform 安装过程中启用并配置加密和镜像功能。

先决条件

- 您已在安装节点上下载了 OpenShift Container Platform 安装程序。
- 在安装节点上安装了 Butane。



注意

Butane 是一个命令行实用程序，OpenShift Container Platform 用来为编写和验证机器配置提供方便的简短语法。如需更多信息，请参阅"使用 Butane 创建机器配置"。

- 您可以使用 Red Hat Enterprise Linux(RHEL)8 机器来生成 Tang Exchange 密钥的指纹。

流程

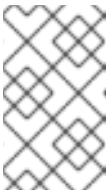
1. 如果要使用 TPM v2 加密集群，请检查每个节点的主机固件中是否需要启用 TPM v2 加密。这在大多数 Dell 系统中是必需的。检查具体系统的手册。
2. 如果要使用 Tang 加密集群，请按照以下步骤操作：
   a. 设置 Tang 服务器或访问现有服务器。具体步骤请查看 网络绑定磁盘加密。
   b. 如果尚未安装，在 RHEL 8 机器上安装 **clevis** 软件包：

```
$ sudo yum install clevis
```

c. 在 RHEL 8 计算机上，运行以下命令来生成交换密钥的指纹。使用 Tang 服务器的 URL 替换 **http://tang1.example.com:7500**

```
$ clevis-encrypt-tang '{"url":"http://tang1.example.com:7500"}' < /dev/null > /dev/null 1
```

1 在本例中，tang **d.socket** 正在侦听 Tang 服务器上的端口 **7500**。



注意

clevis-encrypt-tang 命令生成交换密钥的指纹。此步骤中不会将数据传递给加密命令；**/dev/null** 作为输入而不是纯文本存在。加密的输出也会发送到 **/dev/null**，因为此过程不需要它。

输出示例

```
The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9 1
```

1 Exchange 键的指纹。

当 **Do 您希望信任这些密钥时，显示 [ynYN]** 提示时，键入 **Y**。

d. 可选：对于离线 Tang 置备：

i. 使用 **curl** 命令从服务器获取公告。使用 Tang 服务器的 URL 替换 **http://tang2.example.com:7500** ：

```
$ curl -f http://tang2.example.com:7500/adv > adv.jws && cat adv.jws
```

预期输出

```
{"payload": "eyJrZXlzIjogW3siYWxnIjogIkV", "protected":
"eyJhbGciOiJFUzUxMiIsImN0eSI", "signature": "ADLgk7fZdE3Yt4FyYsm0pHiau7Q"}
```

ii. 为 Clevis 提供广告文件进行加密：

```
$ clevis-encrypt-tang '{"url":"http://tang2.example.com:7500","adv":"adv.jws"}' <
/dev/null > /dev/null
```

e. 如果节点配置了静态 IP 寻址，请运行 **coreos-installer iso custom --dest-karg-append** 或者在安装 RHCOS 节点时使用 **coreos-installer --append-karg** 选项来设置已安装系统的 IP 地址。为您的 **网络附加 ip=** 和其他参数。

重要

有些配置静态 IP 的方法在第一次引导后不会影响 initramfs，且不适用于 Tang 加密。这包括 **coreos-installer --copy-network** 选项、**coreos-installer iso customize --network-keyfile** 选项和 **coreos-installer pxe customize --network-keyfile** 选项，以及在安装过程中在 live ISO 或 PXE 镜像的内核命令行中添加 **ip=** 参数。静态 IP 配置不正确会导致节点第二次引导失败。

3. 在安装节点上，切换到包含安装程序的目录，并为集群生成 Kubernetes 清单：

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 将 **<installation_directory> 替换为您要存储安装文件的目录的路径。**

4. 创建一个 Butane 配置来配置磁盘加密、镜像或两者。例如，若要为计算节点配置存储，请创建一个 **$HOME/clusterconfig/worker-storage.bu** 文件。

引导设备的ane 配置示例

```
variant: openshift
version: 4.16.0
metadata:
  name: worker-storage 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
boot_device:
  layout: x86_64 3
  luks: 4
    tpm2: true 5
    tang: 6
      - url: http://tang1.example.com:7500 7
        thumbprint: PLjNyRdGw03zlRoGjQYMahSZGu9 8
      - url: http://tang2.example.com:7500
        thumbprint: VCJsvZFjBSIHSldw78rOrq7h2ZF
        advertisement: "{"payload": "eyJrZXlzIjogW3siYWxnIjogIkV", "protected":
"eyJhbGciOiJFUzUxMiIsImN0eSI", "signature": "ADLgk7fZdE3Yt4FyYsm0pHiau7Q"}" 9
    threshold: 1 10
  mirror: 11
    devices: 12
      - /dev/sda
      - /dev/sdb
openshift:
  fips: true 13
```

1 2 对于 control plane 配置，在这两个位置中将 **worker** 替换为 **master**。

3 将此字段设置为集群节点的指令集合架构。一些示例包括、**x86_64**、**aarch64** 或 **ppc64le**。

4 如果要加密 root 文件系统，请包含此部分。如需了解更多详细信息，请参阅"关于磁盘加密"。

5 如果要使用受信任的平台模块(TPM)加密根文件系统，请包含此字段。

6 如果要使用一个或多个 Tang 服务器，请包含此部分。

7 指定 Tang 服务器的 URL。在本例中，tang **d.socket** 正在侦听 Tang 服务器上的端口 **7500**。

8 指定上一步中生成的 Exchange key thumbprint。

9 可选：以有效 JSON 格式指定离线 Tang 服务器的公告。

10 指定进行解密时必须满足的最小 TPM v2 和 Tang 加密条件。默认值为 **1**。有关此主题的更多信息，请参阅"配置加密阈值"。

11 如果要镜像引导磁盘，请包含此部分。如需了解更多详细信息，请参阅"关于磁盘镜像"。

12 列出引导磁盘镜像中包含的所有磁盘设备，包括 RHCOS 将安装到的磁盘。

13 包含此指令以在集群中启用 FIPS 模式。

重要

要为集群启用 FIPS 模式，您必须从配置为以 FIPS 模式操作的 Red Hat Enterprise Linux (RHEL) 计算机运行安装程序。有关在 RHEL 中配置 FIPS 模式的更多信息，请参阅在 FIPS 模式中安装该系统。如果要将节点配置为同时使用磁盘加密和镜像，则必须在同一 Butane 配置文件中配置这两个功能。如果要在启用了 FIPS 模式的节点上配置磁盘加密，则必须在同一 Butane 配置文件中包含 **fips** 指令，即使在单独的清单中也启用了 FIPS 模式。

5. 从对应的 Butane 配置文件创建一个 control plane 或计算节点清单，并将它保存到 **<installation_directory>/openshift** 目录。例如，要为计算节点创建清单，请运行以下命令：

```
$ butane $HOME/clusterconfig/worker-storage.bu -o <installation_directory>/openshift/99-worker-storage.yaml
```

对需要磁盘加密或镜像的每种节点类型重复此步骤。

6. 保存 Butane 配置文件，以防将来需要更新清单。

7. 继续进行 OpenShift Container Platform 安装的其余部分。

提示

您可以在安装过程中监控 RHCOS 节点上的控制台日志，以了解与磁盘加密或镜像相关的错误消息。

重要

如果您配置附加数据分区，除非明确请求加密，否则不会加密它们。

验证

安装 OpenShift Container Platform 后，您可以验证是否在集群节点上启用了引导磁盘加密或镜像功能。

1. 在安装主机上，使用 debug pod 访问集群节点：

   a. 为节点启动 debug pod，例如：

   ```
   $ oc debug node/compute-1
   ```

   b. 将 **/host** 设置为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载节点的根文件系统。通过将根目录改为 **/host**，您可以运行节点上可执行路径中包含的二进制文件：

   ```
   # chroot /host
   ```

   注意

   运行 Red Hat Enterprise Linux CoreOS(RHCOS)的 OpenShift Container Platform 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点。但是，如果 OpenShift Container Platform API 不可用，或者 **kubelet** 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

2. 如果配置了引导磁盘加密，请验证是否启用它：

   a. 在 debug shell 中查看节点上 root 映射的状态：

   ```
   # cryptsetup status root
   ```

   输出示例

   ```
   /dev/mapper/root is active and is in use.
     type:    LUKS2 1
     cipher:  aes-xts-plain64 2
     keysize: 512 bits
     key location: keyring
     device:  /dev/sda4 3
     sector size:  512
     offset:  32768 sectors
     size:    15683456 sectors
     mode:    read/write
   ```

   1 加密格式。启用 TPM v2 或 Tang 加密模式时，RHCOS 引导磁盘将使用 LUKS2 格式进行加密。

   2 用于加密 LUKS2 卷的加密算法。如果启用了 FIPS 模式，则使用 **aes-cbc-essiv:sha256** 密码。

   3 包含加密 LUKS2 卷的设备。如果启用了镜像，该值将表示软件镜像设备，如 **/dev/md126**。

   b. 列出绑定到加密设备的 Clevis 插件：

   ```
   # clevis luks list -d /dev/sda4 1
   ```

1 指定上一步中输出中 **device** 字段中列出的设备。

输出示例

```
1: sss '{"t":1,"pins":{"tang":[{"url":"http://tang.example.com:7500"}]}}' 1
```

1 在示例输出中，Tang 插件由 **/dev/sda4** 设备的 Shamir 的 Secret 共享 SSS) Clevis 插件使用。

3. 如果配置了镜像(mirror)，请验证是否启用它：

a. 在 debug shell 中列出节点上的软件 RAID 设备：

```
# cat /proc/mdstat
```

输出示例

```
Personalities : [raid1]
md126 : active raid1 sdb3[1] sda3[0] 1
  393152 blocks super 1.0 [2/2] [UU]

md127 : active raid1 sda4[0] sdb4[1] 2
  51869632 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```

1 **/dev/md126** 软件 RAID 镜像设备使用集群节点中的 **/dev/sda3** 和 **/dev/sdb3** 磁盘设备。

2 **/dev/md127** 软件 RAID 镜像设备使用集群节点中的 **/dev/sda4** 和 **/dev/sdb4** 磁盘设备。

b. 查看上一命令输出中列出的每个软件 RAID 设备的详细信息。以下示例列出了 **/dev/md126** 设备详情：

```
# mdadm --detail /dev/md126
```

输出示例

```
/dev/md126:
           Version : 1.0
     Creation Time : Wed Jul  7 11:07:36 2021
        Raid Level : raid1 1
        Array Size : 393152 (383.94 MiB 402.59 MB)
     Used Dev Size : 393152 (383.94 MiB 402.59 MB)
      Raid Devices : 2
     Total Devices : 2
       Persistence : Superblock is persistent

       Update Time : Wed Jul  7 11:18:24 2021
```

```
          State : clean (2)
    Active Devices : 2 (3)
   Working Devices : 2 (4)
    Failed Devices : 0 (5)
     Spare Devices : 0

Consistency Policy : resync

           Name : any:md-boot (6)
           UUID : ccfa3801:c520e0b5:2bee2755:69043055
         Events : 19

    Number   Major   Minor   RaidDevice State
       0     252       3        0      active sync   /dev/sda3 (7)
       1     252      19        1      active sync   /dev/sdb3 (8)
```

1. 指定设备的 RAID 级别。**raid1** 表示 RAID 1磁盘镜像。
2. 指定 RAID 设备的状态。
3. 4. 指出活跃且正常工作的底层磁盘设备数量。
5. 说明处于故障状态的底层磁盘设备数量。
6. 软件 RAID 设备的名称。
7. 8. 提供有关软件 RAID 设备使用的底层磁盘设备的信息。

c. 列出软件 RAID 设备中挂载的文件系统：

```
# mount | grep /dev/md
```

输出示例

```
/dev/md127 on / type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /etc type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /usr type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /sysroot type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/containers/storage/overlay type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/1 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/2 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/3 type xfs
```

```
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/4 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md127 on /var/lib/kubelet/pods/e5054ed5-f882-4d14-b599-99c050d4e0c0/volume-
subpaths/etc/tuned/5 type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,prjquota)
/dev/md126 on /boot type ext4 (rw,relatime,seclabel)
```

在示例输出中，**/boot** 文件系统挂载到 **/dev/md126 软件** RAID 设备上，root 文件系统挂载到 **/dev/md127**。

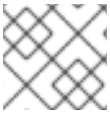
4. 对每种 OpenShift Container Platform 节点类型重复验证步骤。

其他资源

- 有关 TPM v2 和 Tang 加密模式的更多信息，请参阅使用 基于策略的解密配置加密卷的自动解锁。

1.4.4. 配置启用了 RAID 的数据卷

您可以启用软件 RAID 分区以提供外部数据卷。OpenShift Container Platform 支持 RAID 0、RAID 1、RAID 4、RAID 5、RAID 6 和 RAID 10 用于数据保护和容错。如需了解更多详细信息，请参阅"关于磁盘镜像"。



注意

OpenShift Container Platform 4.16 不支持安装驱动器的软件 RAID。

先决条件

- 您已在安装节点上下载了 OpenShift Container Platform 安装程序。
- 您已在安装节点上安装了 Butane。

注意

但ane 是一个命令行实用程序，OpenShift Container Platform 使用它为编写机器配置提供便捷的简写语法，并对机器配置进行额外的验证。如需更多信息，*请参阅使用 Butane 创建机器配置* 部分。

流程

1. 创建一个 Butane 配置，以使用软件 RAID 配置数据卷。
   - 要在用于镜像引导磁盘的相同磁盘上配置带有 RAID 1 的数据卷，请创建一个 **$HOME/clusterconfig/raid1-storage.bu** 文件，例如：

     镜像引导磁盘上的 RAID 1

```
variant: openshift
version: 4.16.0
metadata:
  name: raid1-storage
```

```
  labels:
    machineconfiguration.openshift.io/role: worker
boot_device:
  mirror:
    devices:
      - /dev/disk/by-id/scsi-3600508b400105e210000900000490000
      - /dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6
storage:
  disks:
    - device: /dev/disk/by-id/scsi-3600508b400105e210000900000490000
      partitions:
        - label: root-1
          size_mib: 25000 1
        - label: var-1
    - device: /dev/disk/by-id/scsi-SSEAGATE_ST373453LW_3HW1RHM6
      partitions:
        - label: root-2
          size_mib: 25000 2
        - label: var-2
  raid:
    - name: md-var
      level: raid1
      devices:
        - /dev/disk/by-partlabel/var-1
        - /dev/disk/by-partlabel/var-2
  filesystems:
    - device: /dev/md/md-var
      path: /var
      format: xfs
      wipe_filesystem: true
      with_mount_unit: true
```

1 2 当在引导磁盘中添加数据分区时，推荐最少使用 25000 MB。如果没有指定值，或者指定的值小于推荐的最小值，则生成的 root 文件系统会太小，而在以后进行的 RHCOS 重新安装可能会覆盖数据分区的开始部分。

- 要在辅助磁盘上配置带有 RAID 1 的数据卷，请创建一个 **$HOME/clusterconfig/raid1-alt-storage.bu** 文件，例如：

  辅助磁盘上的 RAID 1

```
variant: openshift
version: 4.16.0
metadata:
  name: raid1-alt-storage
  labels:
    machineconfiguration.openshift.io/role: worker
storage:
  disks:
    - device: /dev/sdc
      wipe_table: true
      partitions:
        - label: data-1
    - device: /dev/sdd
      wipe_table: true
```

```
    partitions:
      - label: data-2
raid:
  - name: md-var-lib-containers
    level: raid1
    devices:
      - /dev/disk/by-partlabel/data-1
      - /dev/disk/by-partlabel/data-2
filesystems:
  - device: /dev/md/md-var-lib-containers
    path: /var/lib/containers
    format: xfs
    wipe_filesystem: true
    with_mount_unit: true
```

2. 从您在上一步中创建的 Butane 配置创建 RAID 清单，并将它保存到 **<installation_directory>/openshift** 目录中。例如，要为计算节点创建清单，请运行以下命令：

```
$ butane $HOME/clusterconfig/<butane_config>.bu -o
<installation_directory>/openshift/<manifest_name>.yaml 1
```

1 将 **<butane_config>** 和 **<manifest_name>** 替换为上一步中的文件名。例如，对于辅助磁盘，**raid1-alt-storage.bu** 和 **raid1-alt-storage.yaml**。

3. 保存 Butane 配置，以防将来需要更新清单。

4. 继续进行 OpenShift Container Platform 安装的其余部分。

1.4.5. 在 CPU (VROC) 数据卷中配置 Intel® 虚拟 RAID

Intel® VROC 是混合 RAID 类型，其中有些维护被卸载到硬件，但显示为软件 RAID 到操作系统。

以下流程配置启用了 Intel® VROC 的 RAID1。

先决条件

- 您有一个启用了 Intel® Volume Management Device (VMD) 的系统。

流程

1. 运行以下命令来创建 Intel® Matrix Storage Manager (IMSM) RAID 容器：

```
$ mdadm -CR /dev/md/imsm0 -e \
 imsm -n2 /dev/nvme0n1 /dev/nvme1n1 1
```

1 RAID 设备名称。本例中列出了两个设备。如果提供多个设备名称，您需要调整 **-n** 标志。例如，列出三个使用 **-n3** 的设备。

2. 在容器内创建 RAID1 存储：

   a. 运行以下命令，在实际 RAID1 卷前面创建一个 dummy RAID0 卷：

```
$ mdadm -CR /dev/md/dummy -l0 -n2 /dev/md/imsm0 -z10M --assume-clean
```

b. 运行以下命令来创建实际的 RAID1 阵列：

```
$ mdadm -CR /dev/md/coreos -l1 -n2 /dev/md/imsm0
```

c. 停止 RAID0 和 RAID1 成员阵列，使用以下命令删除 dummy RAID0 阵列：

```
$ mdadm -S /dev/md/dummy \
  mdadm -S /dev/md/coreos \
  mdadm --kill-subarray=0 /dev/md/imsm0
```

d. 运行以下命令重启 RAID1 阵列：

```
$ mdadm -A /dev/md/coreos /dev/md/imsm0
```

3. 在 RAID1 设备上安装 RHCOS：

a. 运行以下命令，获取 IMSM 容器的 UUID：

```
$ mdadm --detail --export /dev/md/imsm0
```

b. 运行以下命令安装 RHCOS 并包含 **rd.md.uuid** 内核参数：

```
$ coreos-installer install /dev/md/coreos \
  --append-karg rd.md.uuid=<md_UUID> 1
  ...
```

1 IMSM 容器的 UUID。

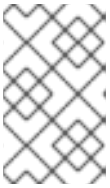
包括安装 RHCOS 所需的任何其他 **coreos-installer** 参数。

1.5. 配置 CHRONY 时间服务

您可以通过修改 chrony **.conf 文件的内容，并将这些内容作为机器配置传递给节点，从而设置 chrony** 时间服务(**chronyd**)使用的时间服务器和相关设置。

流程

1. 创建一个 Butane 配置，包括 **chrony.conf** 文件的内容。例如，要在 worker 节点上配置 chrony，请创建一个 **99-worker-chrony.bu** 文件。



注意

您在配置文件中指定的 Butane 版本应与 OpenShift Container Platform 版本匹配，并且始终以 **0** 结尾。例如：**4.16.0**。有关 Butane 的信息，请参阅"使用 Butane 创建机器配置"。

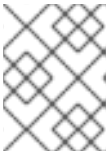
```
variant: openshift
version: 4.16.0
metadata:
```

```
  name: 99-worker-chrony 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 3
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst 4
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

1 2 在 control plane 节点上，在这两个位置中将 **master** 替换为 **worker**。

3 为机器配置文件的 **mode** 字段指定数值模式。在创建文件并应用更改后，**模式** 将转换为十进制值。您可以使用 **oc get mc <mc-name> -o yaml** 命令来检查 YAML 文件。

4 指定任何有效的、可访问的时间源，如 DHCP 服务器提供的源。



注意

对于全机器与全机器的通信，UDP 上的网络时间协议(NTP)是端口 **123**。如果配置了外部 NTP 时间服务器，需要打开 UDP 端口 **123**。

或者，您可以指定以下 NTP 服务器：**1.rhel.pool.ntp.org**, **2.rhel.pool.ntp.org**, 或 **3.rhel.pool.ntp.org**。当您将 NTP 与 DHCP 服务器搭配使用时，您必须在 **chrony.conf** 文件中设置 **sourcedir /run/chrony-dhcp** 参数。

2. 使用 Butane 生成 **MachineConfig** 对象文件 **99-worker-chrony.yaml**，其中包含要交付至节点的配置：

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. 使用以下两种方式之一应用配置：

- 如果集群还没有运行，在生成清单文件后，将 **MachineConfig** 对象文件添加到 **<installation_directory>/openshift** 目录中，然后继续创建集群。
- 如果集群已在运行，请应用该文件：

```
$ oc apply -f ./99-worker-chrony.yaml
```

1.6. 其他资源

- 有关 Butane 的详情，请参考使用 Butane 创建机器配置。
- 有关 FIPS 支持的详情，请参考对 FIPS 加密的支持。

第 2 章 配置防火墙

如果使用防火墙，您必须进行配置，以便 OpenShift Container Platform 可以访问正常工作所需的站点。您必须始终授予某些站点的访问权限，如果使用 Red Hat Insights、Telemetry 服务、托管集群的云以及某些构建策略，则还要授予更多站点的访问权限。

2.1. 为 OPENSHIFT CONTAINER PLATFORM 配置防火墙

在安装 OpenShift Container Platform 前，您必须配置防火墙，以授予 OpenShift Container Platform 所需站点的访问权限。在使用防火墙时，为防火墙提供额外的配置，以便 OpenShift Container Platform 可以访问正常工作所需的站点。

与 worker 节点相比，仅在控制器节点上运行的服务没有特殊的配置注意事项。

注意

如果您的环境在 OpenShift Container Platform 集群前面有一个专用的负载均衡器，请查看防火墙和负载均衡器之间的允许列表，以防止对集群造成不必要的网络限制。

流程

1. 为您的防火墙的允许列表设置以下 registry URL：

URL	port	功能
registry.redhat.io	443	提供核心容器镜像
access.redhat.com	443	托管签名存储，容器客户端需要验证从 **registry.access.redhat.com** 中拉取的镜像。在防火墙环境中，确保此资源位于允许列表中。
registry.access.redhat.com	443	托管存储在 Red Hat Ecosytem Catalog 中的所有容器镜像，包括核心容器镜像。
quay.io	443	提供核心容器镜像
cdn.quay.io	443	提供核心容器镜像
cdn01.quay.io	443	提供核心容器镜像
cdn02.quay.io	443	提供核心容器镜像
cdn03.quay.io	443	提供核心容器镜像
cdn04.quay.io	443	提供核心容器镜像
cdn05.quay.io	443	提供核心容器镜像
cdn06.quay.io	443	提供核心容器镜像

URL	port	功能
sso.redhat.com	443	**https://console.redhat.com** 站点使用来自 **sso.redhat.com** 的身份验证
icr.io	443	提供 IBM Cloud Pak 容器镜像。只有在使用 IBM Cloud Paks 时，才需要这个域。
cp.icr.io	443	提供 IBM Cloud Pak 容器镜像。只有在使用 IBM Cloud Paks 时，才需要这个域。

- 您可以在 allowlist 中使用通配符 **lfquay.io** 和 **lfopenshiftapps.com** 而不是 **cdn.quay.io** 和 **cdn0[1-6].quay.io**。
- 您可以使用通配符 ***.access.redhat.com** 来简化配置，并确保所有子域（包括 **registry.access.redhat.com** ）都被允许。
- 在 allowlist 中添加站点（如 **quay.io** ）时，不要向 denylist 添加通配符条目，如 ***.quay.io**。在大多数情况下，镜像 registry 使用内容交付网络（CDN）来提供镜像。如果防火墙阻止访问，则初始下载请求重定向到一个主机名（如 **cdn01.quay.io** ）时，镜像下载将被拒绝。

2. 将防火墙的允许列表设置为包含为构建所需的语言或框架提供资源的任何站点。
3. 如果不禁用 Telemetry，您必须授予对以下 URL 的访问权限，以访问 Red Hat Insights：

URL	port	功能
cert-api.access.redhat.com	443	Telemetry 所需
api.access.redhat.com	443	Telemetry 所需
infogw.api.openshift.com	443	Telemetry 所需
console.redhat.com	443	Telemetry 和 **insights-operator**需要

4. 如果使用 Alibaba Cloud、Amazon Web Services (AWS)、Microsoft Azure 或 Google Cloud Platform (GCP) 来托管您的集群，您必须授予对为该云提供云供应商 API 和 DNS 的 URL 的访问权限：

云	URL	port	功能
Alibaba	***.aliyuncs.com**	443	需要此项以访问 Alibaba Cloud 服务和资源。查看 Alibaba endpoint_config.go 文件，以查找您使用的区域所允许的确切端点。

云	URL	port	功能
AWS	**aws.amazon.com**	443	用于在 AWS 环境中安装和管理集群。
	***.amazonaws.com** 另外，如果您选择不对 AWS API 使用通配符，则必须在允许列表中包含以下 URL：	443	需要此项以访问 AWS 服务和资源。请参阅 AWS 文档中的 AWS Service Endpoints，以查找您使用的区域所允许的确切端点。
	ec2.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	events.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	iam.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	route53.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	***.s3.amazonaws.com**	443	用于在 AWS 环境中安装和管理集群。
	***.s3. <aws_region>.amazonaws.com**	443	用于在 AWS 环境中安装和管理集群。
	***.s3.dualstack. <aws_region>.amazonaws.com**	443	用于在 AWS 环境中安装和管理集群。
	sts.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	sts. <aws_region>.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	tagging.us-east-1.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。此端点始终为 **us-east-1**，无论集群要部署到的区域。
	ec2. <aws_region>.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	elasticloadbalancing. <aws_region>.amazonaws.com	443	用于在 AWS 环境中安装和管理集群。
	servicequotas. <aws_region>.amazonaws.com	443	必需。用于确认用于部署该服务的配额。
	tagging. <aws_region>.amazonaws.com	443	允许以标签的形式分配 AWS 资源的元数据。

云	URL	port	功能
	***.cloudfront.net**	443	用于提供对 CloudFront 的访问。如果使用 AWS 安全令牌服务(STS)和私有 S3 存储桶，您必须提供对 CloudFront 的访问。
GCP	***.googleapis.com**	443	需要此项以访问 GCP 服务和资源。请参阅 GCP 文档中的 Cloud Endpoints，以查找您的 API 允许的端点。
	accounts.google.com	443	需要此项以访问您的 GCP 帐户。
Microsoft Azure	**management.azure.com**	443	需要此项以访问 Microsoft Azure 服务和资源。请参阅 Microsoft Azure 文档中的 Microsoft Azure REST API 参考，以查找允许您 API 的端点。
	***.blob.core.windows.net**	443	需要下载 Ignition 文件。
	login.microsoftonline.com	443	需要此项以访问 Microsoft Azure 服务和资源。请参阅 Microsoft Azure 文档中的 Azure REST API 参考，以查找您的 API 允许的端点。

5. 将以下 URL 列入允许列表：

URL	port	功能
***.apps.<cluster_name>.<base_domain>**	443	需要此项以访问默认集群路由，除非您在安装过程中设置了入口通配符。
api.openshift.com	443	集群令牌需要，并检查集群是否有可用的更新。
console.redhat.com	443	集群令牌所需。
mirror.openshift.com	443	需要此项以访问镜像安装内容和镜像。此站点也是发行版本镜像签名的来源，但 Cluster Version Operator 只需要一个可正常工作的源。
quayio-production-s3.s3.amazonaws.com	443	需要此项以访问 AWS 中的 Quay 镜像内容。

URL	port	功能
rhcos.mirror.openshift.com	443	需要此项以下载 Red Hat Enterprise Linux CoreOS(RHCOS)镜像。
sso.redhat.com	443	**https://console.redhat.com** 站点使用来自 **sso.redhat.com** 的身份验证
storage.googleapis.com/openshift-release	443	发行版本镜像签名源，但 Cluster Version Operator 只需要一个可正常工作的源。

Operator 需要路由访问权限来执行健康检查。特别是，身份验证和 Web 控制台 Operator 会连接到两个路由，以验证路由是否正常工作。如果您是集群管理员，且不想允许 ***.apps.<cluster_name>.<base_domain>**，则允许这些路由：

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **canary-openshift-ingress-canary.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**，或在 **consoles.operator/cluster** 对象中的 **spec.route.hostname** 项指定的主机名（如果这个项不为空）。

6. 将以下 URL 列入允许的可选第三方内容：

URL	port	功能
registry.connect.redhat.com	443	所有第三方镜像和认证操作器必填.
rhc4tp-prod-z8cxf-image-registry-us-east-1-evenkyleffocxqvofrk.s3.dualstack.us-east-1.amazonaws.com	443	提供对托管在 **registry.connect.redhat.com**上的容器镜像的访问
oso-rhc4tp-docker-registry.s3-us-west-2.amazonaws.com	443	对于 Sonatype Nexus, F5 Big IP operator 是必需的。

7. 如果您使用默认的红帽网络时间协议(NTP)服务器允许以下 URL：
   - **1.rhel.pool.ntp.org**
   - **2.rhel.pool.ntp.org**
   - **3.rhel.pool.ntp.org**

注意

如果您不使用默认的 Red Hat NTP 服务器，请验证您的平台的 NTP 服务器并在防火墙中允许它。

其他资源

- AWS STS 的 OpenID Connect 要求

2.2. OPENSHIFT CONTAINER PLATFORM 网络流列表

网络流列表描述了 OpenShift Container Platform 服务的入站流。对于裸机和云环境，列表中的网络信息是准确的。使用网络流列表中的信息来帮助管理入口流量。您可以将入口流量限制为基本流以提高网络安全性。

要查看或下载原始 CSV 内容，请参阅此资源。

另外，在管理入口流量时请考虑以下动态端口范围：

- **9000-9999**: 主机级别的服务
- **30000-32767**: Kubernetes 节点端口
- **49152-65535**: 动态或专用端口

注意

网络流列表描述了基本 OpenShift Container Platform 安装的入口流量流。它没有描述其他组件的网络流，如 Red Hat Marketplace 提供的可选 Operator。列表不适用于托管 control plane、红帽构建的 MicroShift 或独立集群。

表 2.1. 网络流列表

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	22	主机系统服务	sshd			master	TRUE
入口	TCP	53	openshift-dns	dns-default	dnf-default	dns	master	FALSE
入口	TCP	80	openshift-ingress	router-default	router-default	路由器	master	FALSE
入口	TCP	111	主机系统服务	rpcbind			master	TRUE
入口	TCP	443	openshift-ingress	router-default	router-default	路由器	master	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	1936	openshift-ingress	router-default	router-default	路由器	master	FALSE
入口	TCP	2379	openshift-etcd	etcd	etcd	etcdctl	master	FALSE
入口	TCP	2380	openshift-etcd	healthz	etcd	etcd	master	FALSE
入口	TCP	5050	openshift-machine-api		ironic-proxy	ironic-proxy	master	FALSE
入口	TCP	5051	openshift-machine-api	metal3-state	metal3	metal3-httpd	master	FALSE
入口	TCP	6080	openshift-kube-apiserver		kube-apiserver	kube-apiserver-insecure-readyz	master	FALSE
入口	TCP	6180	openshift-machine-api	metal3-state	metal3	metal3-httpd	master	FALSE
入口	TCP	6183	openshift-machine-api	metal3-state	metal3	metal3-httpd	master	FALSE
入口	TCP	6385	openshift-machine-api		ironic-proxy	ironic-proxy	master	FALSE
入口	TCP	6388	openshift-machine-api	metal3-state	metal3	metal3-httpd	master	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	6443	openshift-kube-apiserver	APIServer	kube-apiserver	kube-apiserver	master	FALSE
入口	TCP	8080	openshift-network-operator		network-operator	network-operator	master	FALSE
入口	TCP	8798	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	machine-config-daemon	master	FALSE
入口	TCP	9001	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	kube-rbac-proxy	master	FALSE
入口	TCP	9099	openshift-cluster-version	cluster-version-operator	cluster-version-operator	cluster-version-operator	master	FALSE
入口	TCP	9100	openshift-monitoring	node-exporter	node-exporter	kube-rbac-proxy	master	FALSE
入口	TCP	9103	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-node	master	FALSE
入口	TCP	9104	openshift-network-operator	metrics	network-operator	network-operator	master	FALSE
入口	TCP	9105	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-ovn-metrics	master	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	9107	openshift-ovn-kubernetes	egressip-node-healthcheck	ovnkube-node	ovnkube-controller	master	FALSE
入口	TCP	9108	openshift-ovn-kubernetes	ovn-kubernetes-control-plane	ovnkube-control-plane	kube-rbac-proxy	master	FALSE
入口	TCP	9192	openshift-cluster-machine-approver	machine-approver	machine-approver	kube-rbac-proxy	master	FALSE
入口	TCP	9258	openshift-cloud-controller-manager-operator	machine-approver	cluster-cloud-controller-manager	cluster-cloud-controller-manager	master	FALSE
入口	TCP	9444	openshift-kni-infra		hapoxy	hapoxy	master	FALSE
入口	TCP	9445	openshift-kni-infra		hapoxy	hapoxy	master	FALSE
入口	TCP	9447	openshift-machine-api		metal3-baremetal-operator		master	FALSE
入口	TCP	9537	主机系统服务	crio-metrics			master	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	9637	openshift-machine-config-operator	kube-rbac-proxy-crio	kube-rbac-proxy-crio	kube-rbac-proxy-crio	master	FALSE
入口	TCP	9978	openshift-etcd	etcd	etcd	etcd-metrics	master	FALSE
入口	TCP	9979	openshift-etcd	etcd	etcd	etcd-metrics	master	FALSE
入口	TCP	9980	openshift-etcd	etcd	etcd	etcd	master	FALSE
入口	TCP	10250	主机系统服务	kubelet			master	FALSE
入口	TCP	10256	openshift-ovn-kubernetes	ovnkube	ovnkube	ovnkube-controller	master	FALSE
入口	TCP	10257	openshift-kube-controller-manager	kube-controller-manager	kube-controller-manager	kube-controller-manager	master	FALSE
入口	TCP	10258	openshift-cloud-controller-manager-operator	cloud-controller	cloud-controller-manager	cloud-controller-manager	master	FALSE
入口	TCP	10259	openshift-kube-scheduler	scheduler	openshift-kube-scheduler	kube-scheduler	master	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	10260	openshift-cloud-controller-manager-operator	cloud-controller	cloud-controller-manager	cloud-controller-manager	master	FALSE
入口	TCP	10300	openshift-cluster-csi-drivers	csi-livenessprobe	csi-driver-node	csi-driver	master	FALSE
入口	TCP	10309	openshift-cluster-csi-drivers	csi-node-driver	csi-driver-node	csi-node-driver-registrar	master	FALSE
入口	TCP	10357	openshift-kube-apiserver	openshift-kube-apiserver-healthz	kube-apiserver	kube-apiserver-check-endpoints	master	FALSE
入口	TCP	17697	openshift-kube-apiserver	openshift-kube-apiserver-healthz	kube-apiserver	kube-apiserver-check-endpoints	master	FALSE
入口	TCP	18080	openshift-kni-infra		coredns	coredns	master	FALSE
入口	TCP	22623	openshift-machine-config-operator	machine-config-server	machine-config-server	machine-config-server	master	FALSE
入口	TCP	22624	openshift-machine-config-operator	machine-config-server	machine-config-server	machine-config-server	master	FALSE

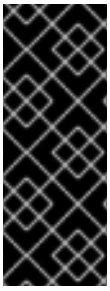
方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	UDP	53	openshift-dns	dns-default	dnf-default	dns	master	FALSE
入口	UDP	111	主机系统服务	rpcbind			master	TRUE
入口	UDP	6081	openshift-ovn-kubernetes	ovn-kubernetes geneve			master	FALSE
入口	TCP	22	主机系统服务	sshd			worker	TRUE
入口	TCP	53	openshift-dns	dns-default	dnf-default	dns	worker	FALSE
入口	TCP	80	openshift-ingress	router-default	router-default	路由器	worker	FALSE
入口	TCP	111	主机系统服务	rpcbind			worker	TRUE
入口	TCP	443	openshift-ingress	router-default	router-default	路由器	worker	FALSE
入口	TCP	1936	openshift-ingress	router-default	router-default	路由器	worker	FALSE
入口	TCP	8798	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	machine-config-daemon	worker	FALSE
入口	TCP	9001	openshift-machine-config-operator	machine-config-daemon	machine-config-daemon	kube-rbac-proxy	worker	FALSE
入口	TCP	9100	openshift-monitoring	node-exporter	node-exporter	kube-rbac-proxy	worker	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	9103	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-node	worker	FALSE
入口	TCP	9105	openshift-ovn-kubernetes	ovn-kubernetes-node	ovnkube-node	kube-rbac-proxy-ovn-metrics	worker	FALSE
入口	TCP	9107	openshift-ovn-kubernetes	egressip-node-healthcheck	ovnkube-node	ovnkube-controller	worker	FALSE
入口	TCP	9537	主机系统服务	crio-metrics			worker	FALSE
入口	TCP	9637	openshift-machine-config-operator	kube-rbac-proxy-crio	kube-rbac-proxy-crio	kube-rbac-proxy-crio	worker	FALSE
入口	TCP	10250	主机系统服务	kubelet			worker	FALSE
入口	TCP	10256	openshift-ovn-kubernetes	ovnkube	ovnkube	ovnkube-controller	worker	TRUE
入口	TCP	10300	openshift-cluster-csi-drivers	csi-liveness probe	csi-driver-node	csi-driver	worker	FALSE
入口	TCP	10309	openshift-cluster-csi-drivers	csi-node-driver	csi-driver-node	csi-node-driver-registrar	worker	FALSE

方向	协议	port	Namespace	service	Pod	Container	节点角色	选填
入口	TCP	18080	openshift-kni-infra		coredns	coredns	worker	FALSE
入口	UDP	53	openshift-dns	dns-default	dnf-default	dns	worker	FALSE
入口	UDP	111	主机系统服务	rpcbind			worker	TRUE
入口	UDP	6081	openshift-ovn-kubernetes	ovn-kubernetes geneve			worker	FALSE

第 3 章 启用 LINUX 控制组版本 1 (CGROUP V1)

自 OpenShift Container Platform 4.14 起，OpenShift Container Platform 在集群中使用 Linux 控制组版本 2 (cgroup v2)。如果您在 OpenShift Container Platform 4.13 或更早版本中使用 cgroup v1，迁移到 OpenShift Container Platform 4.16 不会自动将 cgroup 配置更新至版本 2。全新安装 OpenShift Container Platform 4.14 或更高版本默认使用 cgroup v2。但是，您可以在安装时启用 Linux 控制组群版本 1 (cgroup v1)。在 OpenShift Container Platform 中启用 cgroup v1 禁用集群中的所有 cgroup v2 控制器和层次结构。



重要

cgroup v1 是一个已弃用的功能。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。

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。

cgroup v2 是 Linux cgroup API 的当前版本。cgroup v2 比 cgroup v1 提供多种改进，包括统一层次结构、更安全的子树委派、新功能，如 Pressure Stall Information，以及增强的资源管理和隔离。但是，cgroup v2 与 cgroup v1 具有不同的 CPU、内存和 I/O 管理特征。因此，在运行 cgroup v2 的集群上，一些工作负载可能会遇到内存或 CPU 用量差异。

您可以通过编辑 **node.config** 对象在 cgroup v1 和 cgroup v2 间切换。如需更多信息，请参阅本节"添加资源"中的"在节点上配置 Linux cgroup"。

3.1. 在安装过程中启用 LINUX CGROUP V1

您可以通过创建安装清单来安装集群时启用 Linux 控制组群版本 1 (cgroup v1)。

重要

cgroup v1 是一个已弃用的功能。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。

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。

流程

1. 创建或编辑 **node.config** 对象以指定 **v1** cgroup：

```
apiVersion: config.openshift.io/v1
kind: Node
metadata:
  name: cluster
spec:
  cgroupMode: "v2"
```

2. 照常继续安装。

其他资源

- OpenShift Container Platform 安装概述
- 在节点上配置 Linux cgroup