OpenShift Container Platform 4.3

安装

安装并配置 OpenShift Container Platform 集群

Last Updated: 2020-11-30

OpenShift Container Platform 4.3 安装

安装并配置 OpenShift Container Platform 集群

法律通告

摘要

本文档提供有关安装和配置 OpenShift Container Platform 的信息。

目录

第 1 章 收集安装日志

为帮助排查 OpenShift Container Platform 安装失败的问题，您可以从 bootstrap 和 control plane 或 master 机器中收集日志。

1.1. 先决条件

- 已尝试安装 OpenShift Container Platform 集群，但安装失败。
- 您为安装程序提供了 SSH 密钥，且该密钥已加入到正在运行的 **ss-agent** 进程中。

1.2. 从失败安装中收集日志

如果为安装程序提供了 SSH 密钥，则可以收集与失败安装相关的数据。

注意

用于收集失败安装日志的命令与从正在运行的集群收集日志时所用的命令不同。如果需要从正在运行的集群收集日志，请使用 **oc adm must-gather** 命令。

先决条件

- OpenShift Container Platform 安装在 bootstrap 过程完成前失败。bootstrap 节点必须正在运行，并可通过 SSH 访问。
- **ssh-agent** 进程在您的计算机上处于活跃状态，并且为 **ssh-agent** 进程和安装程序提供了相同的 SSH 密钥。
- 如果尝试在您置备的基础架构中安装集群，则一定需要有 control plane 或 master 机器的完全限定域名。

流程

1. 生成从 bootstrap 和 control plane 机器获取安装日志的命令：
   - 如果使用了安装程序置备的基础架构，请运行以下命令：

     ```
     $ ./openshift-install gather bootstrap --dir=<directory> 1
     ```

     1 **installation_directory** 是安装程序所创建的用来保存 OpenShift Container Platform 定义文件的目录。

     对于安装程序置备的基础架构，安装程序会保存有关集群的信息，因此您不用指定主机名或 IP 地址。

   - 如果使用了您置备的基础架构，请运行以下命令：

     ```
     $ ./openshift-install gather bootstrap --dir=<directory> \ 1
       --bootstrap <bootstrap_address> \ 2
       --master <master_1_address> \ 3
       --master <master_2_address> \ 4
       --master <master_3_address>" 5
     ```

1. **installation_directory** 是安装程序所创建的用来保存 OpenShift Container Platform 定义文件的目录。
2. **<bootstrap_address>** 是集群 bootstrap 机器的完全限定域名或 IP 地址。
3. 4. 5. **<master_address>** 是集群中 control plane 或 master 机器的完全限定域名或 IP 地址。

注意

默认集群包含三个 control plane 机器。如所示，列出所有 control plane 机器，无论集群使用了多少个。

命令输出类似以下示例：

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<directory>/log-bundle-<timestamp>.tar.gz"
```

如果需要创建关安装失败的红帽支持问题单，请在问题单中附上压缩日志。

1.3. 通过到主机的 SSH 连接手动收集日志

在 **must-gather** 或自动收集方法无法正常工作的情况下手动收集日志。

先决条件

- 必须有到主机的 SSH 访问权限。

流程

1. 运行以下命令，使用 **journalctl** 命令从 bootstrap 主机收集 **bootkube.service** 服务日志：

```
$ journalctl -b -f -u bootkube.service
```

2. 使用 Podman 的 logs 命令收集 bootstrap 主机的容器日志。以下命令从主机获取所有容器的日志：

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. 或者，通过运行以下命令来使用 **tail** 命令收集主机的容器日志：

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. 运行 **journalctl** 命令从 master 和 worker 主机收集 **kubelet.service** 和 **crio.service** 服务日志：

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. 使用 **tail** 命令收集 master 和 worker 主机容器日志：

```
$ sudo tail -f /var/log/containers/*
```

1.4. 在不使用 SSH 连接到主机的情况下手动收集日志

在 **must-gather** 或自动收集方法无法正常工作的情况下手动收集日志。

如果您无法对节点进行 SSH 访问，则可以通过访问系统日志来调查主机上发生的情况。

先决条件

- OpenShift Container Platform 安装已完成。
- API 服务仍然可以正常工作。
- 有系统管理员特权。

流程

1. 通过运行以下命令访问 **/var/log** 中的 **journald** 单元日志：

```
$ oc adm node-logs --role=master -u kubelet
```

2. 通过运行以下命令访问 **/var/log** 中的主机文件路径：

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

第 2 章 支持 FIPS 加密

从版本 4.3 开始，您可以安装一个使用经 FIPS 验证的/Modules in Process 加密库的 OpenShift Container Platform 集群。

对于集群中的 Red Hat Enterprise Linux CoreOS (RHCOS) 机器，当机器根据 **install-config.yaml** 文件中的选项的状态进行部署时，会应用这个更改，该文件管理用户可在集群部署过程中更改的集群选项。在 Red Hat Enterprise Linux 机器中，您必须在计划用作 worker 的机器上安装操作系统时，启用 FIPS 模式。这些配置方法可确保集群满足 FIPS 合规审核的要求：在初始系统引导前，只启用经 FIPS 验证/Modules in Process 加密的软件包。

因为 FIPS 必须在集群首次引导操作系统之前启用，所以您不能在部署集群后启用 FIPS。

2.1. OPENSHIFT CONTAINER PLATFORM 中的 FIPS 验证

OpenShift Container Platform 在 Red Hat Enterprise Linux (RHEL) 和 Red Hat CoreOS (RHCOS) 中使用特定的 FIPS 验证/Modules in Process 模块用于使用它们的操作系统组件。请参阅 RHEL7 core crypto components 中的内容。例如，当用户 SSH 到 OpenShift Container Platform 集群和容器时，这些连接会被正确加密。

OpenShift Container Platform 组件以 Go 编写，并使用红帽的 golang 编译器构建。当您为集群启用 FIPS 模式时，所有需要加密签名的 OpenShift Container Platform 组件调用 RHEL 和 RHCOS 加密程序库。

表 2.1. OpenShift Container Platform 4.3 中的 FIPS 模式属性和限制

属性	限制：
RHEL 7 操作系统支持 FIPS 。	FIPS 实现还没有提供一个单一的计算哈希函数和验证基于该哈希的键的函数。在以后的 OpenShift Container Platform 版本中，将继续评估并改进这个限制。
CRI-O 运行时支持 FIPS。	
OpenShift Container Platform 服务支持 FIPS。	
FIPS 验证的/Modules in Process 的加密模块和算法，它们从 RHEL 7 和 RHCOS 二进制文件和镜像中获得。	
使用 FIPS 兼容 golang 编译器。	对 TLS FIPS 的支持当前还不完整，但计划在将来的 OpenShift Container Platform 版本中被完全支持。

2.2. 集群使用的组件支持 FIPS

尽管 OpenShift Container Platform 集群本身使用 FIPS 验证的/Modules in Process 模块，但请确保支持 OpenShift Container Platform 集群的系统也使用 FIPS 验证的/Modules in Process 模块进行加密。

2.2.1. etcd

要确保存储在 etcd 中的 secret 在进程加密中使用 FIPS 验证的/Modules in Process 模块，请以 FIPS 模式引导节点。在使用 FIPS 模式安装集群后，您可以使用 FIPS 批准的 **aes cbc** 加密算法加密 etcd 数据。

2.2.2. Storage

对于本地存储，使用 RHEL 提供的磁盘加密或者使用 RHEL 提供的磁盘加密的容器原生存储。通过将所有数据存储到使用 RHEL 提供的磁盘加密的卷中，并为您的集群启用 FIPS 模式，静态数据和正在启动的数据或网络数据都受到 FIPS 验证的/Modules in Process 加密的保护。您可以将集群配置为加密每个节点的根文件系统，如自定义节点 中所述。

2.2.3. 运行时

要确保容器知道它们在使用 FIPS 验证的/Modules in Process 加密模块的主机上运行，请使用 CRI-O 管理您的运行时。CRI-O 支持 FIPS-Mode，它将容器配置为知道它们是在 FIPS 模式下运行的。

2.3. 在 FIPS 模式下安装集群

要使用 FIPS 模式安装集群，请按照在相应的基础架构中安装自定义集群的步骤进行。在部署集群前，请确定在 **install-config.yaml** 文件中设置了 **fips: true** 。

- Amazon Web Services
- Microsoft Azure
- 裸机
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

要将 **AES CBC** 加密应用到 etcd 数据存储中，请在安装集群后执行加密 etcd 数据操作。

如果您在集群中添加 RHEL 节点，请确定在机器初始引导前启用 FIPS 模式。请参阅 RHEL 7 文档中的 Adding RHEL compute machines to a OpenShift Container Platform cluster 和 Enabling FIPS Mode 部分。

第 3 章 安装配置

3.1. 不同平台的安装方法

您可以在不同的平台上执行不同类型的安装。

注意

不是所有安装选项都可用于所有平台，如下表所示。

表 3.1. 安装程序置备的基础架构选项

	AWS	Azure	GCP	OpenStack	裸机	vSphere	IBM Z
Default	X	X	X				
Custom	X	X	X	X			
Cluster Network Operator	X	X	X				
私有集群	X	X	X				
现有的虚拟私有网络	X	X	X				

表 3.2. 用户置备的基础架构

	AWS	Azure	GCP	OpenStack	裸机	vSphere	IBM Z
Custom	X	X	X		X	X	
Cluster Network Operator					X	X	
Restricted network	X		X		X	X	

3.2. 自定义节点

虽然不鼓励直接更改 OpenShift Container Platform 节点，但有时在实现低级别安全、网络或性能功能时需要这样做。通过以下方法可以对 OpenShift Container Platform 节点直接进行更改：

- 创建包含在清单文件中的 MachineConfig，以便在 **openshift-install** 期间启动集群。

- 创建通过 Machine Config Operator 传递至运行的 OpenShift Container Platform 节点的 MachineConfig。

以下小节描述了您可能需要以这种方式在节点中配置的功能。

3.2.1. 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。下面是一些您可能需要在集群安装过程中添加内核参数以在系统第一次引导前生效的原因：

- 禁用某个功能（比如 SELinux），以使这个功能不会对系统产生任何影响。
- 您需要在系统启动前进行一些低级网络配置。

要向 master 节点或 worker 节点添加内核参数，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。

如需列出引导时可以传递给 RHEL 8 内核的参数，请参阅 Kernel.org 内核参数。如果需要这些参数来完成初始的 OpenShift Container Platform 安装，最好使用这个流程添加内核参数。

流程

1. 为集群生成 Kubernetes 清单：

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

2. 决定您要向 worker 节点或 master 节点添加的内核参数。
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。这个示例为 master 节点添加了一个 **loglevel=7** 的内核参数：

```
$ 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**，以将内核参数添加到 worker 节点。创建一个单独的 YAML 文件，以添加到 master 节点和 worker 节点中。

现在您可以继续创建集群。

3.2.2. 在节点中添加内核模块

对于大多数常用硬件，在计算机启动时，Linux 内核会包括使用这些硬件所需的设备驱动程序模块。但是对于一些硬件来说，在 Linux 中不提供它们的模块。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。此流程介绍了如何为 OpenShift Container Platform 集群中的节点进行此操作。

当首先按照这些说明部署了一个内核模块后，这个模块就可用于当前内核。如果安装了新内核，kmods-via-containers 软件将被重建并部署该模块，以便使新内核可使用该模块的兼容版本。

使这个功能可以在每个节点中保持模块最新状态的方法是：

- 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。
- 如果检测到一个新的内核，该服务会重建该模块并将其安装到内核中

有关此过程所需软件的详情，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。
- 在本流程中，构建您的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一个 **yum** 存储库，该程序存储库包含重建该模块所需的内核和相关软件包。该内容最好由一个有效的 RHEL 订阅提供。

3.2.2.1. 构建并测试内核模块容器

在将内核模块部署到 OpenShift Container Platform 集群之前，您可以在单独的 RHEL 系统中测试此过程。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. 获取 RHEL 8 系统，然后注册并订阅它：

```
# subscription-manager register
Username: yourname
Password: ***************
# subscription-manager attach --auto
```

2. 安装构建软件和容器所需的软件：

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

3. 克隆 kmod-via-containers 存储库：

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

4. 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：

```
$ cd kmods-via-containers/
$ sudo make install
$ sudo systemctl daemon-reload
```

5. 获取内核模块源代码。通过源代码，可以构建一个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，使用以下步骤将这些内容克隆到您的系统中：

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

6. 编辑示例中的配置文件 **simple-kmod.conf**，将 Dockerfile 的名称改为 **Dockerfile.rhel**，该文件如下所示：

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

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"
```

7. 为您的内核模块创建一个 **kmods-via-containers@.service** 实例（在这个示例中是 **simple-kmod**）并启用它：

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

8. 启用并启动 systemd 服务，然后检查状态：

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

9. 要确认载入了内核模块，请使用 **lsmod** 命令列出模块：

```
$ lsmod | grep simple_
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

10. simple-kmod 示例还有几个其它方法来测试它是否可以正常工作。使用 **dmesg** 在内核环缓冲中查找 "Hello world" 信息：

```
$ dmesg | grep 'Hello world'
[ 6420.761332] Hello world from simple_kmod.
```

在 /**proc** 中检查 **simple-procfs-kmod** 的值：

```
$ sudo cat /proc/simple-procfs-kmod
simple-procfs-kmod number = 0
```

运行 spkut 命令从模块中获取更多信息：

```
$ sudo spkut 44
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，当系统引导这个服务时，会检查新内核是否在运行。如果有一个新内核，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，它将只载入该模块。

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

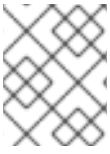
在这两种情况下，每个节点都需要在检测到新内核时可以获得内核软件包及相关软件包。您可以通过以下几种方法设置每个节点来获取该内容。

- 为每个节点提供 RHEL 权利。
- 从现有的 RHEL 主机获取 RHEL 权利。把 **/etc/pki/entitlement** 目录中的 RHEL 授权复制到与您在构建 Ignition 配置时提供的其他文件相同的位置。
- 在 Dockerfile 中，添加包括了内核和其他软件包的 **yum** 存储库。这必须包括新内核软件包，因为它们需要与新安装的内核相匹配。

3.2.2.2.1. 通过 MachineConfig 置备内核模块

通过将内核模块软件与 MachineConfig 一起打包，您可以在安装时或通过 Machine Config Operator 向 worker 或 master 节点发送该软件。

首先创建您要使用的基本 Ignition 配置。在安装时，Ignition 配置需要包含添加到集群中的 **core** 用户的 **authorized_keys** 文件中的 ssh 公钥。如果在以后通过 MCO 添加 MachineConfig，则不需要 ssh 公钥。对于这两种方式，simple-kmod 服务示例都会创建一个 systemd 单元文件，它需要一个 **kmods-via-containers@simple-kmod.service**



注意

systemd 单元是 上游程序错误的一个临时解决方案。确保 **kmods-via-containers@simple-kmod.service** 在引导时启动：

1. 获取 RHEL 8 系统，然后注册并订阅它：

```
# subscription-manager register
Username: yourname
Password: ***************
# subscription-manager attach --auto
```

2. 安装构建软件所需的软件：

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

3. 创建 Ignition 配置文件以创建 systemd 单元文件：

```
$ mkdir kmods; cd kmods
$ cat <<EOF > ./baseconfig.ign
{
  "ignition": { "version": "2.2.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "groups": ["sudo"],
        "sshAuthorizedKeys": [
          "ssh-rsa AAAA"
        ]
      }
    ]
  },
  "systemd": {
    "units": [{
      "name": "require-kvc-simple-kmod.service",
      "enabled": true,
      "contents": "[Unit]\nRequires=kmods-via-containers@simple-
kmod.service\n[Service]\nType=oneshot\nExecStart=/usr/bin/true\n\n[Install]\nWantedBy=multi-
user.target"
    }]
  }
}
EOF
```

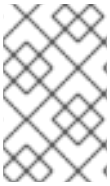
注意

您必须将公共 SSH 密钥添加到 **baseconfig.ign** 文件中，以便 **openshift-install** 使用该文件。如果通过 MCO 创建 MachineConfig，则不需要使用公共 SSH 密钥。

4. 创建如下所示的基础 MCO YAML 片断：

```
$ cat <<EOF > mc-base.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 10-kvc-simple-kmod
spec:
  config:
EOF
```

+



注意

Mc-base.yaml 设置为在 **worker** 节点上部署内核模块 。要部署到 master 节点上，请将角色从 **worker** 更改为 **master**。要进行这两个操作，您可以通过为不同类型的部署使用不同的文件名来重复整个过程。

1. 获得 **kmods-via-containers** 软件：

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

2. 获取您的模块软件。在这个示例中使用 **kvc-simple-kmod**：

3. 使用之前克隆的存储库，创建一个 fakeroot 目录，并将您要通过 Ignition 传递的文件放置到这个目录：

```
$ FAKEROOT=$(mktemp -d)
$ cd kmods-via-containers
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
$ cd ../kvc-simple-kmod
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

4. 获取名为 **filetranspiler** 的工具程序及相关的依赖软件：

```
$ cd ..
$ sudo yum install -y python3
git clone https://github.com/ashcrow/filetranspiler.git
```

5. 生成最终的 MachineConfig YAML（**mc.yaml**），它包含基础 Ignition 配置、基础 MachineConfig，以及包括您要提供的文件的 fakeroot 目录：

```
$ ./filetranspiler/filetranspile -i ./baseconfig.ign \
    -f ${FAKEROOT} --format=yaml --dereference-symlinks \
    | sed 's/^/     /' | (cat mc-base.yaml -) > 99_simple-kmod.yaml
```

6. 如果集群还没有启用，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，按照以下方法应用该文件：

```
$ 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。

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

```
$ lsmod | grep simple_
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

3.2.3. 在安装过程中加密磁盘

在 OpenShift Container Platform 安装过程中，您可以在所有 master 和 worker 节点上启用磁盘加密。这个功能：

- 可用于安装程序置备的基础架构和用户置备的基础架构部署
- 只在 Red Hat Enterprise Linux CoreOS (RHCOS) 系统上被支持
- 在清单安装阶段设置磁盘加密，以便加密所有写入磁盘的数据（从第一次引导开始）
- 只加密 root 文件系统中的数据（/**dev/mapper/coreos-luks-root** on /）
- 不需要用户干预就可以提供密码短语
- 使用 AES-256-CBC 加密
- 应该为集群启用来支持 FIPS。

支持的加密模式有两种：

- TPM v2：这是首选模式。TPM v2 在安全加密处理器中存储密码短语。要实现 TPM v2 磁盘加密，请创建 Ignition 配置文件，如下所述。
- Tang：要使用 Tang 来加密集群，您需要使用一个 Tang 服务器。Clevis 在客户端实现解密。只有在裸机上的安装支持 Tang 加密模式。

按照以下两个流程之一为集群中的节点启用磁盘加密。

3.2.3.1. 启用 TPM v2 磁盘加密

使用这个流程在 OpenShift Container Platform 部署过程中启用 TPM v2 模式磁盘加密。

流程

1. 检查每个节点的 BIOS 是否需要启用 TPM v2 加密。这在大多数 Dell 系统中是必需的。请参阅您的计算机的相关手册。
2. 为集群生成 Kubernetes 清单：

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

3. 在 **openshift** 目录中，创建一个 master 和/或 worker 文件来为这些节点加密磁盘。以下是这两个文件的例子：

```
$ cat << EOF > ./99_openshift-worker-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tpm
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
```

```
        mode: 420
        path: /etc/clevis.json
EOF
```

```
$ cat << EOF > ./99_openshift-master-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tpm
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

4. 生成 YAML 文件的备份副本。在创建集群时会删除该文件，所以您应该进行备份。

5. 继续进行 OpenShift Container Platform 部署的剩余部分。

3.2.3.2. 启用 Tang 磁盘加密

使用这个流程在 OpenShift Container Platform 部署过程中启用 Tang 模式磁盘加密。

流程

1. 使用一个 Red Hat Enterprise Linux 服务器配置加密设置，并运行 **openshift-install** 来安装一个集群及 **oc**。

2. 设置或访问现有的 Tang 服务器。具体步骤请查看网络绑定磁盘加密 。另请参阅 Securing Automated Decryption New Cryptography and Techni。

3. 添加内核参数来配置集群在安装 Red Hat Enterprise Linux CoreOS（RHCOS）时的网络。例如：在内核命令行中添加参数来配置 DHCP 网络，指定 **ip=dhcp**，或者设置静态网络。对于 DHCP 和静态网络，您必须提供 **rd.neednet=1** 内核参数。

重要

跳过这一步会导致第二次引导失败。

4. 生成 thumbprint。安装 clevis 软件包（如果还没有安装），并从 Tang 服务器生成一个 thumbprint。使用 Tang 服务器 URL 替换 **url** 的值：

```
$ sudo yum install clevis -y
$ echo nifty random wordwords \
    | clevis-encrypt-tang \
```

```
      '{"url":"https://tang.example.org"}'

The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9

Do you wish to trust these keys? [ynYN] y
eyJhbmc3SlRyMXpPenc3ajhEQ01tZVJiTi1oM...
```

5. 创建 Base64 编码文件，使用新生成的 Tang server 替换 **url**，thumbprint 替换**thp**：

```
$ (cat <<EOM
{
 "url": "https://tang.example.com",
 "thp": "PLjNyRdGw03zlRoGjQYMahSZGu9"
}
EOM
) | base64 -w0

ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRocCI6ICJaUk1leTFjR3cw
N3psVExxHYlhuUWFoUzBHdTAiCn0K
```

6. 将 TPM2 示例中的 “source” 替换为 worker 和/或 master 节点的其中一个或两个示例的 Base64 编码文件：

重要

您必须添加 **rd.neednet=1** 内核参数。

```
$ cat << EOF > ./99-openshift-worker-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tang
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRoc
CI6ICJaUk1leTFjR3cwN3psVExxHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
  kernelArguments:
    - rd.neednet=1 <.>
EOF
```

必需。

```
$ cat << EOF > ./99_openshift-master-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tang
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRoc
CI6ICJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
  kernelArguments:
    - rd.neednet=1 <.>
EOF
```

必需。

7. 继续进行 OpenShift Container Platform 部署的剩余部分。

3.2.4. 配置 chrony 时间服务

您可以通过修改 **chrony.conf** 文件的内容来设置 chrony 时间服务 (chronyd) 使用的时间服务器和相关设置，并通过一个 MachineConfig 将这些内容传递给节点。

流程

1. 创建 **chrony.conf** 文件的内容并对其进行 base64 编码。例如：

```
$ cat << EOF | base64
    server clock.redhat.com iburst
    driftfile /var/lib/chrony/drift
    makestep 1.0 3
    rtcsync
    logdir /var/log/chrony
    EOF

ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXIvbGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2
RpciAv
dmFyL2xvZy9jaHJvbnkK
```

2. 创建 MachineConfig 文件，将 base64 字符串替换为您刚刚创建的字符串。本例将文件添加到 **master** 节点。您可以将其更改为 **worker**，或为 **worker** 角色创建额外的 MachineConfig：

```
$ cat << EOF > ./99_masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,c2VydmVyIGNsb2NrLnJlZGhhdC5jb20gaWJ1cnN0CmRyaWZ0ZmlsZSAvdmFyL2xp
Yi9jaHJvbnkvZHJpZnQKbWFrZXN0ZXAgMS4wIDMKcnRjc3luYwpsb2dkaXIgL3Zhci9sb2cvY2h
yb255Cg==
          verification: {}
        filesystem: root
        mode: 420
        path: /etc/chrony.conf
  osImageURL: ""
EOF
```

3. 对配置文件做一个副本备份。

4. 如果集群还没有启动，生成清单文件，将此文件添加到 **openshift** 目录中，然后继续创建集群。

5. 如果集群已在运行，按照以下方法应用该文件：

```
$ oc apply -f ./masters-chrony-configuration.yaml
```

3.2.5. 其他资源

如需更多与 FIPS 相关的信息，请参阅 Support for FIPS cryptography。

3.3. 创建用于在受限网络中安装的镜像 REGISTRY

在受限网络中置备的基础架构上安装集群前，您必须将所需的容器镜像镜像(mirror)到那个环境中。在受限网络中安装仅支持您置备的基础架构，不支持安装程序置备的基础架构。您也可以在不受限制的网络中使用此流程来确保集群只使用满足您机构对外部内容控制的容器镜像。

重要

您必须可以访问互联网来获取所需的容器镜像。在这一流程中，您要将镜像 registry 放在可访问您的网络以及互联网的镜像（mirror）主机上。如果您无法访问镜像主机，请使用断开连接的步骤将镜像复制到可跨网络界限的设备中。

3.3.1. 关于镜像 registry

您可以镜像 OpenShift Container Platform 安装和后续的产品更新镜像(mirror)镜像(mirror)。这些步骤使用同样的过程。发行版本镜像（包含内容描述）及其引用的镜像都被镜像(mirror)。此外，Operator 目录源镜像及其引用的镜像必须针对您使用的每个 Operator 进行镜像(mirror)。镜像内容后，您要将每个集群配置为从镜像 registry 中检索此内容。

镜像 registry 可以是支持最新容器镜像 API（称为 **schema2** ）的任何容器 registry。所有主要的云供应 registry，以及 Red Hat Quay、artifactory 和开源 Docker 发行 registry 都被支持。使用其中一个 registry 可确保 OpenShift Container Platform 可在断开连接的环境中检查各个镜像的完整性。

镜像 registry 必须可以被您置备的集群中的每台机器访问。如果 registry 无法访问，更新或常规操作（如工作负载重新定位）可能会失败。因此，您必须以高度可用的方式运行镜像 registry，镜像 registry 至少必须与 OpenShift Container Platform 集群的生产环境可用性相匹配。

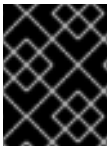
使用 OpenShift Container Platform 镜像填充镜像 registry 时，可以采用以下两种情况。如果您的主机可以同时访问互联网和您的镜像 registry，而不能访问您的集群节点，您可以直接从该机器中镜像该内容。这个过程被称为 *连接的镜像(mirror)*。如果没有这样的主机，则必须将该镜像文件镜像到文件系统中，然后将该主机或者可移动介质放入受限环境中。这个过程被称为 *断开连接的镜像*。

3.3.2. 准备您的镜像主机

执行镜像步骤前，必须准备主机以检索内容并将其推送到远程位置。

3.3.2.1. 通过下载二进制文件安装 CLI

您需要安装 CLI（**oc**） 来使用命令行界面与 OpenShift Container Platform 进行交互。您可在 Linux、Windows 或 macOS 上安装 **oc**。



重要

如果安装了旧版本的 **oc**，则无法使用 OpenShift Container Platform 4.3 中的所有命令。下载并安装新版本的 **oc**。

3.3.2.1.1. 在 Linux 上安装 CLI

您可以按照以下流程在 Linux 上安装 OpenShift CLI（**oc**）二进制文件。

流程

1. 访问 Red Hat OpenShift Cluster Manager 网站的 Infrastructure Provider 页面。
2. 选择您的基础架构供应商及安装类型。
3. 在 **Command-line interface** 部分，从下拉菜单中选择 **Linux**，并点 **Download command-line tools**。
4. 解包存档：

```
$ tar xvzf <file>
```

5. 把 **oc** 二进制代码放到 **PATH** 中的目录中。
   执行以下命令可以查看当前的 **PATH** 设置：

```
$ echo $PATH
```

安装 CLI 后，就可以使用**oc**命令：

```
$ oc <command>
```

3.3.2.1.2. 在 Windows 上安装 CLI

您可以按照以下流程在 Windows 上安装 OpenShift CLI（**oc**）二进制代码。

流程

1. 访问 Red Hat OpenShift Cluster Manager 网站的 Infrastructure Provider 页面。
2. 选择您的基础架构供应商及安装类型。
3. 在 **Command-line interface** 部分，从下拉菜单中选择 **Windows**，点 **Download command-line tools**。
4. 使用 ZIP 程序解压存档。
5. 把 **oc** 二进制代码放到 **PATH** 中的目录中。
   要查看您的 **PATH**，请打开命令提示窗口并执行以下命令：

```
C:\> path
```

安装 CLI 后，就可以使用**oc**命令：

```
C:\> oc <command>
```

3.3.2.1.3. 在 macOS 上安装 CLI

您可以按照以下流程在 macOS 上安装 OpenShift CLI（**oc**）二进制代码。

流程

1. 访问 Red Hat OpenShift Cluster Manager 网站的 Infrastructure Provider 页面。
2. 选择您的基础架构供应商及安装类型。
3. 在 **Command-line interface** 部分，从下拉菜单中选择 **MacOS**，并点 **Download command-line tools**。
4. 解包和解压存档。
5. 将 **oc** 二进制文件移到 PATH 的目录中。
   要查看您的 **PATH**，打开一个终端窗口并执行以下命令：

```
$ echo $PATH
```

安装 CLI 后，就可以使用**oc**命令：

```
$ oc <command>
```

3.3.3. 配置允许对容器镜像进行镜像的凭证

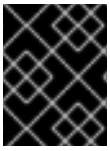
创建容器镜像 registry 凭证文件，允许将红帽的镜像镜像到您的镜像环境中。

警告

安装集群时不要使用此镜像 registry 凭据文件作为 pull secret。如果在安装集群时提供此文件，集群中的所有机器都将具有镜像 registry 的写入权限。

警告

此过程需要您可以对镜像 registry 上的容器镜像 registry 进行写操作，并将凭证添加到 registry pull secret。



重要

安装集群时不要使用此镜像 registry 凭据文件作为 pull secret。如果在安装集群时提供此文件，集群中的所有机器都将具有镜像 registry 的写入权限。

先决条件

- 配置了一个镜像（mirror） registry 在受限网络中使用。
- 您在镜像 registry 中标识了镜像仓库的位置，以将容器镜像镜像(mirror)到这个位置。
- 您置备了一个镜像 registry 帐户，允许将镜像上传到该镜像仓库。

流程

在安装主机上完成以下步骤：

1. 从 Red Hat OpenShift Cluster Manager 站点的 Pull Secret 页面下载 **registry.redhat.io** 的 pull secret，将它保存为一个 **.json** 文件。
2. 为您的镜像 registry 生成 base64 编码的用户名和密码或令牌：

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

1 通过 **<user_name>** 和 **<password>** 指定 registry 的用户名和密码。

3. 以 JSON 格式创建您的 pull secret 副本：

```
$ cat ./pull-secret.text | jq .  > <path>/<pull-secret-file> 1
```

1 指定到存储 pull secret 的文件夹的路径，以及您创建的 JSON 文件的名称。

该文件类似于以下示例：

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

4. 编辑新文件并添加描述 registry 的部分：

```
  "auths": {
    "<mirror_registry>": { 1
      "auth": "<credentials>", 2
      "email": "you@example.com"
    },
```

1 对于 **<mirror_registry>**，指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。例如： **registry.example.com** 或 **registry.example.com:5000**

2 使用 **<credentials>** 为您的镜像 registry 指定 base64 编码的用户名和密码。

该文件类似于以下示例：

```
{
  "auths": {
    "<mirror_registry>": {
      "auth": "<credentials>",
      "email": "you@example.com"
    },
```

```
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

1. 编辑新文件并添加描述 registry 的部分：

```
  "auths": {
...
    "<mirror_registry>": { 1
      "auth": "<credentials>", 2
      "email": "you@example.com"
  },
...
```

1 对于 **<mirror_registry>**，指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。例如： **registry.example.com** 或 **registry.example.com:5000**

2 使用 **<credentials>** 为您的镜像 registry 指定 base64 编码的用户名和密码。

该文件类似于以下示例：

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "<mirror_registry>": {
      "auth": "<credentials>",
      "email": "you@example.com"
    },
```

```
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

3.3.4. 镜像 OpenShift Container Platform 镜像存储库

镜像要在集群安装或升级过程中使用的 OpenShift Container Platform 镜像仓库。

先决条件

- 您的镜像主机可访问互联网。
- 您已将镜像 registry 配置为在受限网络中使用，并可访问您配置的证书和凭证。
- 您已从 Red Hat OpenShift Cluster Manager 站点的 Pull Secret 页面下载了 pull secret，并已修改为包含镜像存储库身份验证信息。

流程

在镜像主机上完成以下步骤：

1. 查看 OpenShift Container Platform 下载页面，以确定您要安装的 OpenShift Container Platform 版本，并决定 Repository Tags 页中的相应标签（tag）。
2. 设置所需的环境变量：

```
$ export OCP_RELEASE=<release_version> 1
$ export LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>' 2
$ export LOCAL_REPOSITORY='<local_repository_name>' 3
$ export PRODUCT_REPO='openshift-release-dev' 4
$ export LOCAL_SECRET_JSON='<path_to_pull_secret>' 5
$ export RELEASE_NAME="ocp-release" 6
$ export ARCHITECTURE=<server_architecture> 7
$ REMOVABLE_MEDIA_PATH=<path> 8
```

1. 对于 **<release_version>**，请指定与 OpenShift Container Platform 版本对应的标签，用于您的架构，如 **4.3.0**。
2. 对于 **<local_registry_host_name>**，请指定镜像存储库的 registry 域名；对于 **<local_registry_host_port>**，请指定用于提供内容的端口。
3. 对于 **<local_repository_name>**，请指定要在 registry 中创建的仓库名称，如 **ocp4/openshift4**。
4. 要镜像的存储库。对于生产环境版本，必须指定 **openshift-release-dev**。
5. 对于 **<path_to_pull_secret>**，请指定您创建的镜像 registry 的 pull secret 的绝对路径和文件名。
6. 发行版本镜像。对于生产环境版本，您必须指定 **ocp-release**。
7. 对于 **server_architecture**，指定服务器的构架，如 **x86_64**。

8 对于 **<path>**，指定托管镜像的目录的路径。

3. 将版本镜像镜像(mirror)到内部容器 registry：

- 如果您的镜像主机无法访问互联网，请执行以下操作：

i. 将可移动介质连接到连接到互联网的系统。

ii. 查看要镜像的镜像和配置清单：

```
$ oc adm -a ${LOCAL_SECRET_JSON} release mirror
--from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
--to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}
--to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE} --run-dry
```

iii. 记录上一命令输出中的 **imageContentSources** 部分。您的镜像信息与您的镜像存储库相对应，您必须在安装过程中将 **imageContentSources** 部分添加到 **install-config.yaml** 文件中。

iv. 将镜像镜像到可移动介质的目录中：

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```

v. 将介质上传到受限网络环境中，并将镜像上传到本地容器 registry。

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}
```

- 如果本地容器 registry 连接到镜像主机，请执行以下操作：

i. 使用以下命令直接将发行版镜像推送到本地 registry:

```
$ oc adm -a ${LOCAL_SECRET_JSON} release mirror \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
```

该命令将发行信息提取为摘要，其输出包括安装集群时所需的 **imageContentSources** 数据。

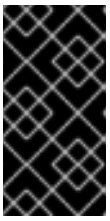
ii. 记录上一命令输出中的 **imageContentSources** 部分。您的镜像信息与您的镜像存储库相对应，您必须在安装过程中将 **imageContentSources** 部分添加到 **install-config.yaml** 文件中。

注意

镜像名称在镜像过程中被修补到 Quay.io， podman 镜像将在 bootstrap 虚拟机的 registry 中显示 Quay.io。

1. 要创建基于您镜像内容的安装程序，请提取内容并将其固定到发行版中：

```
$ oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}"
```



重要

要确保将正确的镜像用于您选择的 OpenShift Container Platform 版本，您必须从镜像内容中提取安装程序。

您必须在有活跃互联网连接的机器上执行这个步骤。

3.3.5. 准备集群以收集支持数据

使用受限网络的集群必须导入默认的 must-gather 镜像，以便为红帽支持收集调试数据。在默认情况下，must-gather 镜像不会被导入，受限网络中的集群无法访问互联网来从远程存储库拉取最新的镜像。

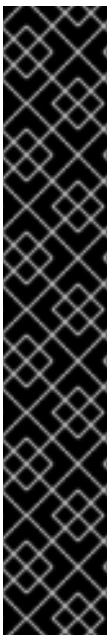
流程

1. 从您的安装有效负载中导入默认 must-gather 镜像：

```
$ oc import-image is/must-gather -n openshift
```

3.3.6. 使用带有备用或经过镜像的 registry 的 Samples Operator 镜像流

OpenShift 命名空间中大多数由 Samples Operator 管理的镜像流指向位于 registry.redhat.io 上红帽 registry 中的镜像。镜像功能不适用于这些镜像流。



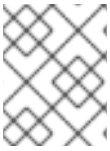
重要

jenkins、**jenkins-agent-maven** 和 **jenkins-agent-nodejs** 镜像流的确来自安装有效负载，并由 Samples Operator 管理，因此这些镜像流不需要进一步的镜像操作。

将 Sample Operator 配置文件中的 **samplesRegistry** 字段设置为 registry.redhat.io 有很多冗余，因为它已经定向到 registry.redhat.io，只用于 Jenkins 镜像和镜像流。它还会破坏 Jenkins 镜像流的安装有效负载。

Samples Operator 禁止将以下 registry 用于 Jenkins 镜像流：

- docker.io
- registry.redhat.io
- registry.access.redhat.com
- *.quay.io。



注意

cli、**installer**、**must-gather** 和 **test** 镜像流虽然属于安装有效负载的一部分，但不由 Samples Operator 管理。此流程中不涉及这些镜像流。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 为您的镜像 registry 创建 pull secret。

流程

1. 访问被镜像（mirror）的特定镜像流的镜像，例如：

```
$ oc get is <imagestream> -n openshift -o json | jq .spec.tags[].from.name | grep
registry.redhat.io
```

2. 将 registry.redhat.io 中与您在受限网络环境中需要的任何镜像流关联的镜像，镜像 (mirror) 成一个定义的镜像 (mirror)：

```
$ oc image mirror registry.redhat.io/rhscl/ruby-25-rhel7:latest ${MIRROR_ADDR}/rhscl/ruby-
25-rhel7:latest
```

3. 在集群的镜像配置对象中，为镜像添加所需的可信 CA：

```
$ oc create configmap registry-config --from-
file=${MIRROR_ADDR_HOSTNAME}..5000=$path/ca.crt -n openshift-config
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"registry-config"}}}' --type=merge
```

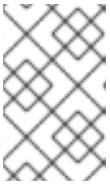
4. 更新 Samples Operator 配置对象中的 **samplesRegistry** 字段，使其包含镜像配置中定义的镜像位置的 **hostname** 部分：

```
$ oc edit configs.samples.operator.openshift.io -n openshift-cluster-samples-operator
```

注意

这是必要的，因为镜像流导入过程在此刻不使用镜像（mirro）或搜索机制。

5. 将所有未镜像的镜像流添加到 Samples Operator 配置对象的 **skippedImagestreams** 字段。或者，如果您不想支持任何示例镜像流，请在 Samples Operator 配置对象中将 Samples Operator 设置为 **Removed**。



注意

镜像流导入失败两小时后，任何没有跳过的未镜像的镜像流，或者如果 Samples Operator 没有更改为 **Removed**，都会导致 Samples Operator 报告 **Degraded** 状态。

OpenShift 命名空间中的多个模板都引用镜像流。因此，使用 **Removed** 清除镜像流和模板，将避免在因为缺少镜像流而导致镜像流和模板无法正常工作时使用它们。

3.3.7. 后续步骤

- 在您在受限网络中置备的基础架构上安装集群，如 VMware vSphere、裸机或 Amazon Web Services。

3.4. 可用的集群自定义

大多数集群配置和自定义在 OpenShift Container Platform 集群部署后完成。有若干*配置资源*可用。

您可以修改配置资源来配置集群的主要功能，如镜像 registry、网络配置、镜像构建操作以及用户身份供应商。

如需设置这些资源的当前信息，请使用 **oc explain** 命令，如 **oc explain builds --api-version=config.openshift.io/v1**

3.4.1. 集群配置资源

所有集群配置资源都作用于全局范围（而非命名空间），且命名为 **cluster**。

资源名称	描述
apiserver.config.openshift.io	提供 api-server 配置，如证书和证书颁发机构。
authentication.config.openshift.io	控制集群的用户身份供应商和验证配置。
build.config.openshift.io	控制集群中所有构建的默认和强制配置。
console.config.openshift.io	配置 Web 控制台界面的行为，包括注销行为。
featuregate.config.openshift.io	启用 FeatureGates，以便您能使用技术预览功能。
image.config.openshift.io	配置应如何对待特定的镜像 registry（允许、禁用、不安全、CA 详情）。
ingress.config.openshift.io	与路由相关的配置详情，如路由的默认域。
oauth.config.openshift.io	配置用户身份供应商，以及与内部 OAuth 服务器流程相关的其他行为。
project.config.openshift.io	配置项目的创建方式，包括项目模板。
proxy.config.openshift.io	定义需要外部网络访问的组件要使用的代理。注意：目前不是所有组件都会消耗这个值。

资源名称	描述
scheduler.config.openshift.io	配置调度程序行为，如策略和默认节点选择器。

3.4.2. Operator 配置资源

这些配置资源是集群范围的实例，即 **cluster**，控制归特定 Operator 所有的特定组件的行为。

资源名称	描述
console.operator.openshift.io	控制控制台外观，如品牌定制
config.imageregistry.operator.openshift.io	配置内部镜像 registry 设置，如公共路由、日志级别、代理设置、资源约束、副本数和存储类型。
config.samples.operator.openshift.io	配置 Samples Operator，以控制在集群上安装哪些镜像流和模板示例。

3.4.3. 其他配置资源

这些配置资源代表一个特定组件的单一实例。在有些情况下，您可以通过创建多个资源实例来请求多个实例。在其他情况下，Operator 只消耗指定命名空间中的特定资源实例名称。如需有关如何和何时创建其他资源实例的详情，请参考具体组件的文档。

资源名称	实例名称	命名空间	描述
alertmanager.monitoring.coreos.com	main	openshift-monitoring	控制 alertmanager 部署参数。
ingresscontroller.operator.openshift.io	default	openshift-ingress-operator	配置 Ingress Operator 行为，如域、副本数、证书和控制器放置。

3.4.4. 信息资源

可以使用这些资源检索集群信息。请不要直接编辑这些资源。

资源名称	实例名称	描述
clusterversion.config.openshift.io	version	在 OpenShift Container Platform 4.3 中，不得自定义生产集群的 ClusterVersion 资源，而应遵循相关流程来更新集群。
dns.config.openshift.io	cluster	无法修改集群的 DNS 设置。您可以查看 DNS Operator 状态。
infrastructure.config.openshift.io	cluster	允许集群与其云供应商交互的配置详情。
network.config.openshift.io	cluster	无法在安装后修改集群网络。要自定义您的网络，请遵循相关的流程在安装过程中自定义联网。

3.5. 配置防火墙

如果使用防火墙，您必须进行配置，以便 OpenShift Container Platform 能访问正常运作所需要的网站。您必须始终授予一些站点的访问权限，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。

3.5.1. 为 OpenShift Container Platform 配置防火墙

在安装 OpenShift Container Platform 之前，您必须配置防火墙，以授予 OpenShift Container Platform 所需站点的访问权限。

和在 worker 节点上运行的服务相比，运行在控制器节点中的服务没有特殊的配置注意事项。

流程

1. 允许以下 registry URL：

URL	功能
registry.redhat.io	提供核心容器镜像
quay.io	提供核心容器镜像
sso.redhat.com	**https://cloud.redhat.com/openshift** 站点使用 **sso.redhat.com** 提供的身份验证
openshift.org	提供 Red Hat Enterprise Linux CoreOS（RHCOS）镜像

2. 将提供构建所需语言或框架的资源的任何站点列入允许列表。
3. 如果不禁用 Telemetry，您必须授予对以下 URL 的访问权限，以便能访问 Red Hat Insights：

URL	功能
cert-api.access.redhat.com	Telemetry 所需
api.access.redhat.com	Telemetry 所需
infogw.api.openshift.com	Telemetry 所需
https://cloud.redhat.com/api/ingress	Telemetry 和 **insights-operator** 需要

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

云	URL	功能
AWS	***.amazonaws.com**	需要此项以访问 AWS 服务和资源。请参阅 AWS 文档中 AWS Service Endpoints，以确定您使用的区域所允许的具体端点。
GCP	***.googleapis.com**	需要此项以访问 GCP 服务和资源。请参阅 GCP 文档中的 Cloud Endpoints，以确定您的 API 所允许的端点。
	accounts.google.com	需要此项以访问您的 GCP 帐户。
Azure	**management.azure.com**	需要此项以访问 Azure 服务和资源。请参阅 Azure 文档中的Azure REST API 参考，以确定您的 API 所允许的端点。

5. 将以下 URL 列入允许列表：

URL	功能
mirror.openshift.com	需要此项以访问镜像安装内容和镜像。此站点也是发行镜像签名的来源，但 Cluster Version Operator 只需要一个可正常工作的源。
storage.googleapis.com/openshift-release	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。
***.apps.<cluster_name>.<base_domain>**	需要此项以访问默认的集群路由，除非您在安装过程中设置了入口通配符。
quay-registry.s3.amazonaws.com	需要此项以访问 AWS 中的 Quay 镜像内容。
api.openshift.com	需要此项以检查集群是否有可用的更新。

URL	功能
art-rhcos-ci.s3.amazonaws.com	需要此项以下载 Red Hat Enterprise Linux CoreOS（RHCOS）镜像。
api.openshift.com	集群令牌所需
cloud.redhat.com/openshift	集群令牌所需

操作员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>**，请允许这些路由：

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**，或者，在 **consoles.operator/cluster** 对象的 **spec.route.hostname** 字段中指定的主机名（如果此字段非空）。

3.6. 配置私有集群

安装 OpenShift Container Platform 版本 4.3 集群后，您可以将其某些核心组件设置为私有。

重要

您只能对使用您置备的云供应商的基础架构的集群配置进行这个更改。。

3.6.1. 关于私有集群

默认情况下，OpenShift Container Platform 被置备为使用可公开访问的 DNS 和端点。在部署集群后，您可以将 DNS、Ingress Controller 和 API 服务器设置为私有。

DNS
如果在安装程序置备的基础架构上安装 OpenShift Container Platform，安装程序会在预先存在的公共区中创建记录，并在可能的情况下为集群自己的 DNS 解析创建一个私有区。在公共区和私有区中，安装程序或集群为 ***.apps** 和 Ingress 创建 DNS，为 API 服务器创建 **api**。

公共和私有区中的 ***.apps** 记录是相同的，因此当您删除公有区时，私有区为集群无缝地提供所有 DNS 解析。

Ingress Controller
因为默认 Ingress 对象是作为公共对象创建的，所以负载均衡器是面向互联网的，它在公共子网中。您可以将默认 Ingress Controller 替换为内部控制器。

API Server
默认情况下，安装程序为 API 服务器创建适当的网络负载均衡器，供内部和外部流量使用。

在 Amazon Web Services（AWS）上，会分别创建独立的公共和私有负载均衡器。负载均衡器是基本相同的，唯一不同是带有一个额外的、用于在集群内部使用的端口。虽然安装程序根据 API 服务器要求自动创建或销毁负载均衡器，但集群并不管理或维护它们。只要保留集群对 API 服务器的访问，您可以手动修改

或移动负载均衡器。对于公共负载均衡器，需要打开端口 6443，并根据 **/readyz** 路径配置 HTTPS 用于健康检查。

在 Google Cloud Platform 上，会创建一个负载均衡器来管理内部和外部 API 流量，因此您无需修改负载均衡器。

在 Microsoft Azure 上，会创建公共和私有负载均衡器。但是，由于当前实施的限制，您刚刚在私有集群中保留两个负载均衡器。

3.6.2. 将 DNS 设置为私有

部署集群后，您可以修改其 DNS 使其只使用私有区。

流程

1. 查看集群的 DNS 自定义资源：

```
$ oc get dnses.config.openshift.io/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>: owned
  publicZone:
    id: Z2XXXXXXXXXXA4
status: {}
```

请注意，**spec** 部分包含一个私有区和一个公共区。

2. 修补 DNS 自定义资源以删除公共区：

```
$ oc patch dnses.config.openshift.io/cluster --type=merge --patch='{"spec": {"publicZone": null}}'
dns.config.openshift.io/cluster patched
```

因为 Ingress Controller 在创建 Ingress 对象时会参考 DNS 定义，因此当您创建或修改 Ingress 对象时，只会创建私有记录。

重要

在删除公共区时，现有 Ingress 对象的 DNS 记录不会修改。

3. 可选：查看集群的 DNS 自定义资源，并确认已删除公共区：

```
$ oc get dnses.config.openshift.io/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>-wfpg4: owned
status: {}
```

3.6.3. 将 Ingress Controller 设置为私有

部署集群后，您可以修改其 Ingress Controller 使其只使用私有区。

流程

1. 修改默认 Ingress Controller，使其仅使用内部端点：

```
$ oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: Internal
EOF
ingresscontroller.operator.openshift.io "default" deleted
ingresscontroller.operator.openshift.io/default replaced
```

删除公共 DNS 条目，并更新私有区条目。

3.6.4. 将 API 服务器限制为私有

将集群部署到 Amazon Web Services（AWS）或 Microsoft Azure 后，可以重新配置 API 服务器，使其只使用私有区。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 使用具有 **admin** 权限的用户登陆到 web 控制台。

流程

1. 在 AWS 或 Azure 的 web 门户或控制台中，执行以下操作：
   a. 找到并删除相关的负载均衡器组件。
      - 对于 AWS，删除外部负载均衡器。私有区的 API DNS 条目已指向内部负载均衡器，它使用相同的配置，因此您无需修改内部负载均衡器。
      - 对于 Azure，删除负载均衡器的 **api-internal** 规则。
   b. 在公共区中删除 **api.$clustername.$yourdomain** DNS 条目。
2. 在终端中列出集群机器：

```
$ oc get machine -n openshift-machine-api
NAME                      STATE    TYPE       REGION     ZONE         AGE
lk4pj-master-0            running  m4.xlarge  us-east-1  us-east-1a   17m
lk4pj-master-1            running  m4.xlarge  us-east-1  us-east-1b   17m
lk4pj-master-2            running  m4.xlarge  us-east-1  us-east-1a   17m
lk4pj-worker-us-east-1a-5fzfj  running  m4.xlarge  us-east-1  us-east-1a  15m
lk4pj-worker-us-east-1a-vbghs  running  m4.xlarge  us-east-1  us-east-1a  15m
lk4pj-worker-us-east-1b-zgpzg  running  m4.xlarge  us-east-1  us-east-1b  15m
```

   在以下步骤中，修改 control plane 机器（名称中包含 **master** 的机器）。

3. 从每台 control plane 机器移除外部负载均衡器。
   a. 编辑 **master** Machine 对象，删除对外部负载均衡器的引用。

```
$ oc edit machines -n openshift-machine-api <master_name> 1
```

   1 指定要修改的 control plane 或 master 机器的名称。

   b. 删除描述外部负载均衡器的行（在以下示例中已被标记），保存并退出对象规格：

```
...
spec:
  providerSpec:
    value:
      ...
      loadBalancers:
      - name: lk4pj-ext 1
        type: network 2
      - name: lk4pj-int
        type: network
```

   1 2 删除这一行。

   c. 对名称中包含 **master** 的每个机器重复这个过程。