

OpenShift 4.5 WorkShop

基于UPI的离线裸机环境  
OpenShift集群部署手册

REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ver.Type | Ver.Num | Date | Author | Purpose/Change |
| Draft Version | 0.1 | 2020/01/23 | 林文炜 | Temporary version |
| Draft Version | 0.2 | 2020/02/19 | 林文炜 | Temporary version |
| Draft Version | 0.3 | 2020/02/26 | 林文炜 | Temporary version |
| Draft Version | 0.5 | 2020/03/31 | 林文炜 | Temporary version |
| Review version | 0.6 | 2020/04/20 | 林文炜 | 第一阶段送审版本 |
| Fork version | 0.66-1 | 2020/05/04 | 刘晓宇 | 为简化环境下安装OCP 4.3定制的版本 |
| Fork version | 0.66-2 | 2020/05/09 | 刘晓宇 | 文档结构和内容优化版 |
| Fork version | 0.66-3 | 2020/05/12 | 刘晓宇 | 升级到OCP-4.3.18，其它问题修正 |
| Fork version | 0.66-4 | 2020/06/07 | 刘晓宇 | 增加第9章，安装常见问题 |
| Fork version | 0.66-5 | 2020/09/12 | 刘晓宇 | 升级到OCP4.5.8、OCP4.5.9 |

Contents

[1 OpenShift集群说明和部署规划 1](#_Toc42460856)

[1.1 重要声明 1](#_Toc42460857)

[1.2 本文的OpenShift集群安装方式 1](#_Toc42460858)

[1.3 OpenShift集群部署过程和节点角色说明 1](#_Toc42460859)

[1.4 OpenShift集群部署拓扑 1](#_Toc42460860)

[1.5 集群节点角色 2](#_Toc42460861)

[1.6 虚机节点资源配置规划 2](#_Toc42460862)

[1.7 虚机节点网络配置规划 2](#_Toc42460863)

[1.8 DNS域名解析规划 3](#_Toc42460864)

[2 创建Bastion和Support虚机节点 3](#_Toc42460865)

[3 获得离线安装文件 5](#_Toc42460866)

[3.1 获得离线安装文件方法1 5](#_Toc42460867)

[3.2 获得离线安装文件方法2 5](#_Toc42460868)

[3.2.1 获取OCP 4.5的最新版本号 5](#_Toc42460869)

[3.2.2 创建目录 5](#_Toc42460870)

[3.2.3 准备下载环境 5](#_Toc42460871)

[3.2.4 下载离线YUM源 6](#_Toc42460872)

[3.2.5 下载并安装OCP客户端 7](#_Toc42460873)

[3.2.6 下载openshift-install 7](#_Toc42460874)

[3.2.7 下载OCP核心镜像 7](#_Toc42460875)

[3.2.8 下载离线ImageStream镜像包 11](#_Toc42460876)

[3.2.9 下载CoreOS 4.5镜像 16](#_Toc42460877)

[4 配置Support节点的服务 16](#_Toc42460878)

[4.1 上传安装介质 16](#_Toc42460879)

[4.1.1 安装介质文件说明 16](#_Toc42460880)

[4.1.2 上传安装介质 17](#_Toc42460881)

[4.2 设置防火墙和SELinux 17](#_Toc42460882)

[4.2.1 关闭firewalld防火墙 17](#_Toc42460883)

[4.2.2 设置selinux 17](#_Toc42460884)

[4.5 设置环境变量 17](#_Toc42460885)

[4.4 配置本地临时YUM源 18](#_Toc42460886)

[4.4.1 准备YUM源所需的文件 18](#_Toc42460887)

[4.4.2 配置本地临时YUM源 18](#_Toc42460888)

[4.4.5 创建基于HTTP的YUM服务 18](#_Toc42460889)

[4.5 安装配置DNS服务 19](#_Toc42460890)

[4.5.1 安装BIND服务 19](#_Toc42460891)

[4.5.2 设置BIND配置文件 20](#_Toc42460892)

[4.5.3 配置Zone区域 20](#_Toc42460893)

[4.5.4 重启BIND服务 23](#_Toc42460894)

[4.5.5 测试正反向DNS解析 23](#_Toc42460895)

[4.6 配置正式YUM源 24](#_Toc42460896)

[4.6.1 配置Support节点的YUM源 24](#_Toc42460897)

[4.6.2 配置Bastion节点的YUM源 25](#_Toc42460898)

[4.6.3 安装基础软件包，验证YUM源 25](#_Toc42460899)

[4.7 再次验证DNS服务 26](#_Toc42460900)

[4.8 部署NTP服务 27](#_Toc42460901)

[4.8.1 设置正确的时区 27](#_Toc42460902)

[4.8.2 安装chrony 27](#_Toc42460903)

[4.8.3 配置chrony 27](#_Toc42460904)

[4.8.4 检查chrony服务端启动 28](#_Toc42460905)

[4.8.5 验证chrony服务同步 28](#_Toc42460906)

[4.9 部署本地Docker Registry 29](#_Toc42460907)

[4.9.1 创建Docker Registry相关目录 29](#_Toc42460908)

[4.9.2 创建访问Docker Registry的证书 29](#_Toc42460909)

[4.9.3 安装Docker Registry 29](#_Toc42460910)

[4.9.4 从本地访问Docker Registry 30](#_Toc42460911)

[4.9.5 从远程访问Docker Registry 30](#_Toc42460912)

[4.10 部署HAProxy负载均衡服务 31](#_Toc42460913)

[5 定制化安装文件 33](#_Toc42460914)

[5.1 设置基础环境变量 33](#_Toc42460915)

[5.2 准备OpenShift核心镜像 33](#_Toc42460916)

[5.2.1 安装oc客户端 33](#_Toc42460917)

[5.2.2 安装podman等镜像操作工具 34](#_Toc42460918)

[5.2.3 向Docker Registry导入OpenShift核心镜像 34](#_Toc42460919)

[5.3 准备CoreOS启动文件及其HTTP访问服务 34](#_Toc42460920)

[5.3.1 准备CoreOS启动文件的HTTP访问服务 34](#_Toc42460921)

[5.3.2 准备CoreOS Metal Raw文件 34](#_Toc42460922)

[5.3.3 准备CoreOS ISO文件 35](#_Toc42460923)

[5.3.4 验证可访问到CorsOS的ISO和RWS启动文件 36](#_Toc42460924)

[5.4 准备Ignition引导文件 37](#_Toc42460925)

[5.4.1 安装openshift-install 37](#_Toc42460926)

[5.4.2 准备install-config.yaml文件 37](#_Toc42460927)

[5.4.5 创建manifest文件 40](#_Toc42460928)

[5.4.4 创建Ignition引导文件 40](#_Toc42460929)

[5.4.5 创建Ignition引导文件下载目录 41](#_Toc42460930)

[6 创建Bootstrap、Master、Worker虚拟机节点 41](#_Toc42460931)

[7 安装OCP集群 41](#_Toc42460932)

[7.1 第一阶段：部署bootstrap阶段 42](#_Toc42460933)

[7.2 第二阶段：部署master阶段 43](#_Toc42460934)

[7.3 第三阶段：配置master阶段 49](#_Toc42460935)

[7.4 第四阶段：部署worker阶段 49](#_Toc42460936)

[8 集群初始化和功能验证 52](#_Toc42460937)

[8.1 用户管理 52](#_Toc42460938)

[8.1.1 新建集群管理员 52](#_Toc42460939)

[8.1.2 新建普通用户 53](#_Toc42460940)

[8.1.3 删除kubeadmin 53](#_Toc42460941)

[8.2 部署BusyBox应用 53](#_Toc42460942)

[8.2.1 导入BusyBox的应用镜像 53](#_Toc42460943)

[8.2.2 部署BusyBox应用 54](#_Toc42460944)

[8.3 部署NFS服务 54](#_Toc42460945)

[8.3.1 安装NFS服务 54](#_Toc42460946)

[8.3.2 验证NFS共享目录 55](#_Toc42460947)

[8.4 配置OpenShift内部镜像库的存储 55](#_Toc42460948)

[8.4.1 创建内部镜像库使用的NFS目录 55](#_Toc42460949)

[8.4.2 创建PV 55](#_Toc42460950)

[8.4.5 创建PVC 56](#_Toc42460951)

[8.4.4 指定内部镜像库使用PVC 56](#_Toc42460952)

[8.5 配置NFS StorageClass 56](#_Toc42460953)

[8.5.1 创建NFS 目录 56](#_Toc42460954)

[8.5.2 创建NFS StorageClass部署配置 57](#_Toc42460955)

[8.5.3 执行NFS StorageClass部署配置 60](#_Toc42460956)

[8.5.4 部署测试应用验证NFS存储 60](#_Toc42460957)

[8.5.5 附：删除已部署的NFS StorageClass 64](#_Toc42460958)

[8.6 导入ImageStream并配置Sample Operator 64](#_Toc42460959)

[8.6.1 上传容器镜像 64](#_Toc42460960)

[8.6.2 修改OCP配置 65](#_Toc42460961)

[8.7 部署Gogs+PostgreSQL环境 68](#_Toc42460962)

[8.7.1 准备Gogs镜像 68](#_Toc42460963)

[8.7.2 使用Template结合StorageClass部署PostgreSQL数据库 69](#_Toc42460964)

[8.7.3 部署Gogs应用 69](#_Toc42460965)

[8.7.4 Gogs应用初始化 69](#_Toc42460966)

[8.7.5 Gogs配置持久化 71](#_Toc42460967)

[9 安装常见错误 73](#_Toc42460968)

[9.1 bootstrap网卡名错误导致启动失败 73](#_Toc42460969)

[9.1.1 现象 73](#_Toc42460970)

[9.1.2 确认 74](#_Toc42460971)

[9.1.3 修复错误 74](#_Toc42460972)

[9.2 无法现在 74](#_Toc42460973)

[9.2.1 现象 74](#_Toc42460974)

[9.2.1 确认 74](#_Toc42460975)

# OpenShift集群说明和部署规划

## 重要声明

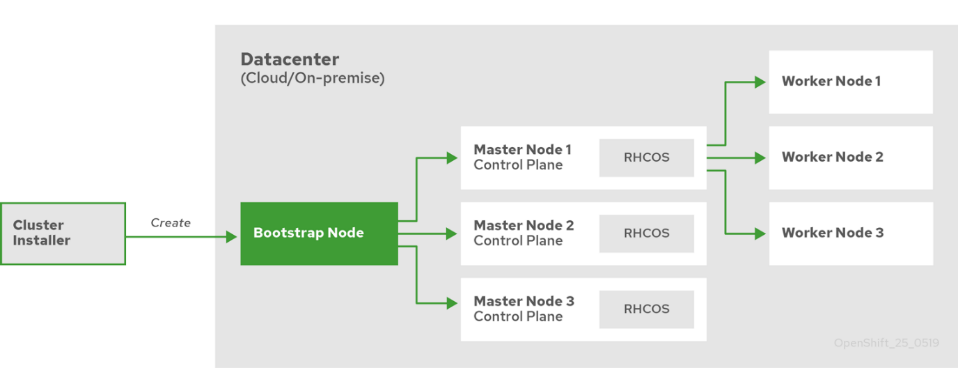
1. 本文内容只适用于安装运行OpenShift 4.3 - OpenShift 4.5的试用环境，而不适用于生产环境。
2. 本文中的操作只适用于和本文相同的OpenShift安装方式和部署架构，其它安装方式和部署架构请参考Red Hat官方文档。

## 本文的OpenShift集群安装方式

为了能适用于更多的测试环境，本文采用了基于BareMetal的UPI离线安装方式部署一个精简的OpenShift 4.5集群。在安装过程中建议使用KVM、Virtualbox、vSphere虚拟化环境。

## OpenShift集群部署过程和节点角色说明

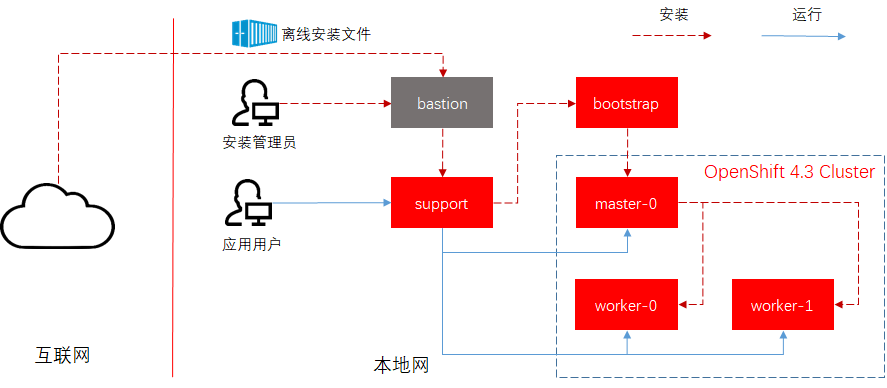
部署一套OpenShift集群的核心过程和相关节点角色如下：



* 启动bootstrap节点和master节点（如果使用UPI方式，需要手工启动）。
* master节点从bootstrap节点获取资源并启动。
* master节点通过bootstrap节点创建出etcd集群。
* bootstrap节点启动一个临时的Kubernetes control plane并连接使用上述的etcd集群。
* 临时的Kubernetes Control plane将最终的production control plane调度到master节点。
* 临时的Kubernetes Control plane关闭并将控制权交给production control plane。
* bootstrap节点将OpenShift Container Platform组件注入到production control plane。
* 安装程序关闭bootstrap节点（UPI部署将需要您手工操作）。
* 接下来，Control plane部署worker nodes。
* Control plane以Operator形式部署额外的服务。

## OpenShift集群部署拓扑

为了能在一个48G内存硬件（可以是多个物理节点组成）的环境安装OpenShift集群，本文将尽量合并集群中节点角色，以安装部署一个简化的OpenShift 4.x集群，该OpenShift集群拓扑如下图：



## 集群节点角色

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | 节点类型 | 功能角色 | 本文标识 |
| bastion | 可访问Internet和内网的一台RHEL环境  本文使用KVM+RHEL | 用来从Internet在线下载软件并验证运行在[Support]中的服务 | [Bastion] |
| bootstrap | KVM+CoreOS | 安装初始化，当安装完master节点后就可以关闭bootstrap | [Bootstrap] |
| support | KVM+RHEL | DNS 服务  NTP服务  YUM Reposityory服务  Container Image Registry  NFS Storage服务  HAProxy Load Balance  Cluster Installer（运行安装操作） | [Support] |
| master-0 | KVM+CoreOS | OCP Cluster-Master  OCP Cluster -Etcd | [Master] |
| worker-0 | KVM+CoreOS | OCP Cluster -Worker Node | [Worker] |
| worker-1 | KVM+CoreOS | OCP Cluster -Worker Node |

## 虚机节点资源配置规划

以下6个虚拟机全部运行在宿主机上。可以根据情况准备宿主机硬件，但每个宿主机最少16G内存，且所有宿主机内存总和最少为48 G。下表以3台宿主机（每台16G内存）为例规划虚拟机运行节点。

说明：由于代表Bootstrap和Work的虚拟机可以不同时运行，因此运行它们的虚机可以运行于同一个宿主机上，并且可对宿主机内存超配。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hostname | OS | CPU (Core) | MEM (GB) | DISK (GB) | 宿主机节点 |
| bastion | RHEL 7.6 / RHEL 7.7 | 1 | 2 | 10 | PC1 |
| support | RHEL 7.6 / RHEL 7.7 | 2 | 4 | 120 | PC2 |
| bootstrap | RHCOS | 2 | 10 | 100 | PC1 |
| master-0 | RHCOS | 4 | 14 | 120 | PC3 |
| worker-0 | RHCOS | 4 | 10 | 100 | PC1 |
| worker-1 | RHCOS | 4 | 10 | 100 | PC2 |

## 虚机节点网络配置规划

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hostname | IP Address | Gateway | NETMASK | DNS |
| bastion | 192.168.1.13 | 192.168.1.1 | 255.255.255.0 | 192.168.1.12  （即support节点的地址） |
| support | 192.168.1.12 |
| bootstrap | 192.168.1.100 |
| master-0 | 192.168.1.101 |
| worker-0 | 192.168.1.110 |
| worker-1 | 192.168.1.111 |

## DNS域名解析规划

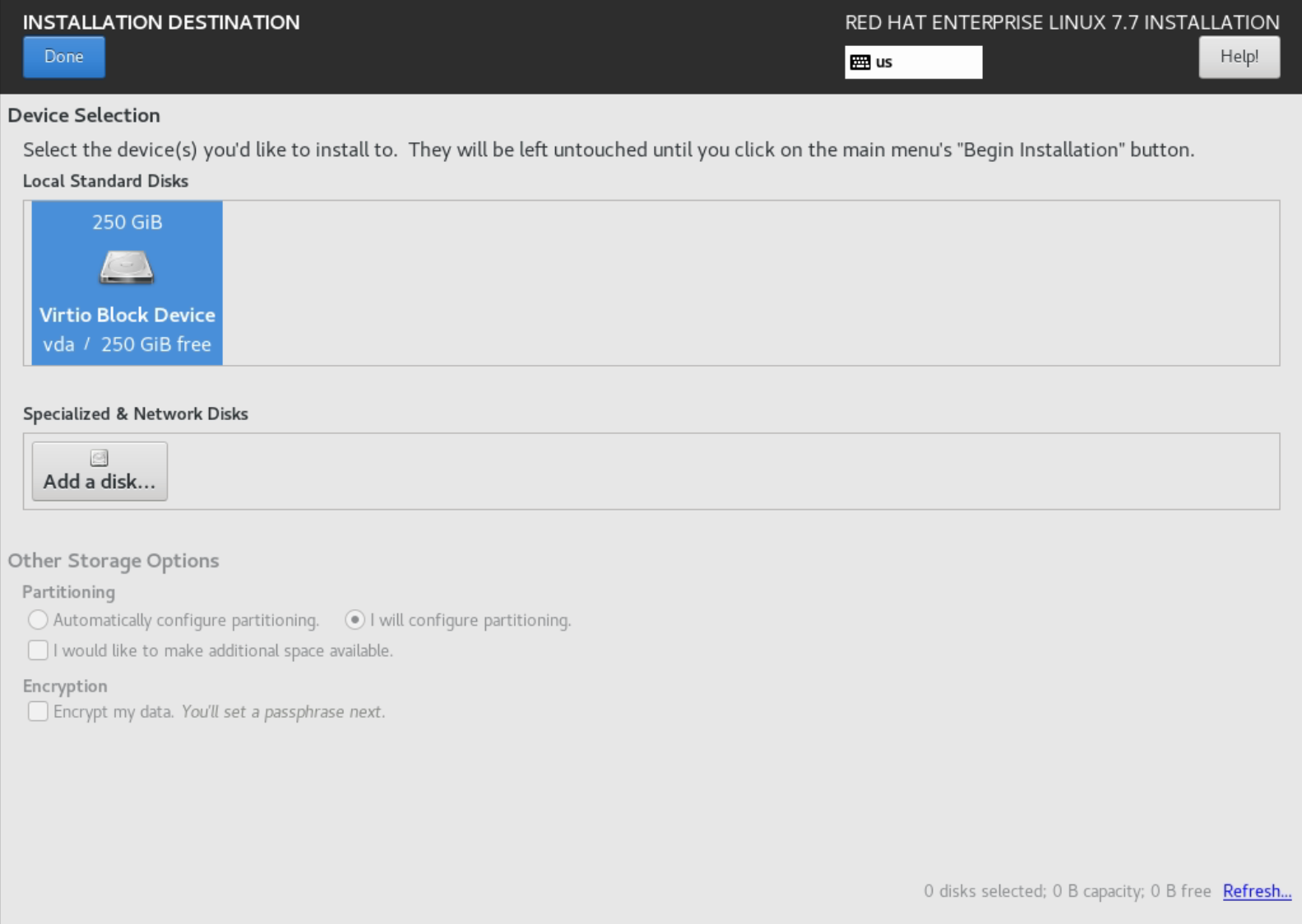
|  |  |
| --- | --- |
| DNS Part | Value |
| DOMAIN | example.internal |
| OCP\_CLUSTER\_ID | ocp4-1 |

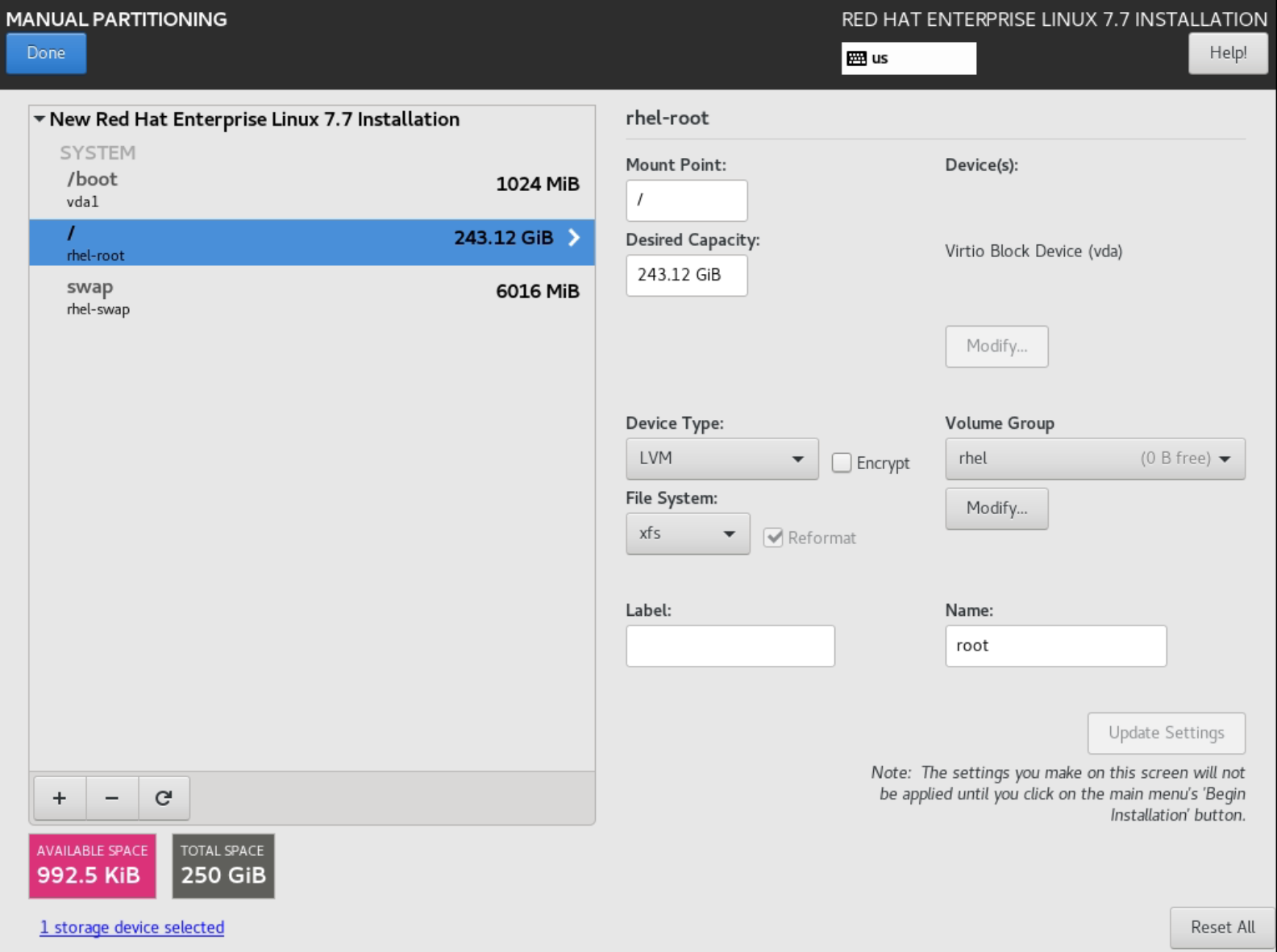
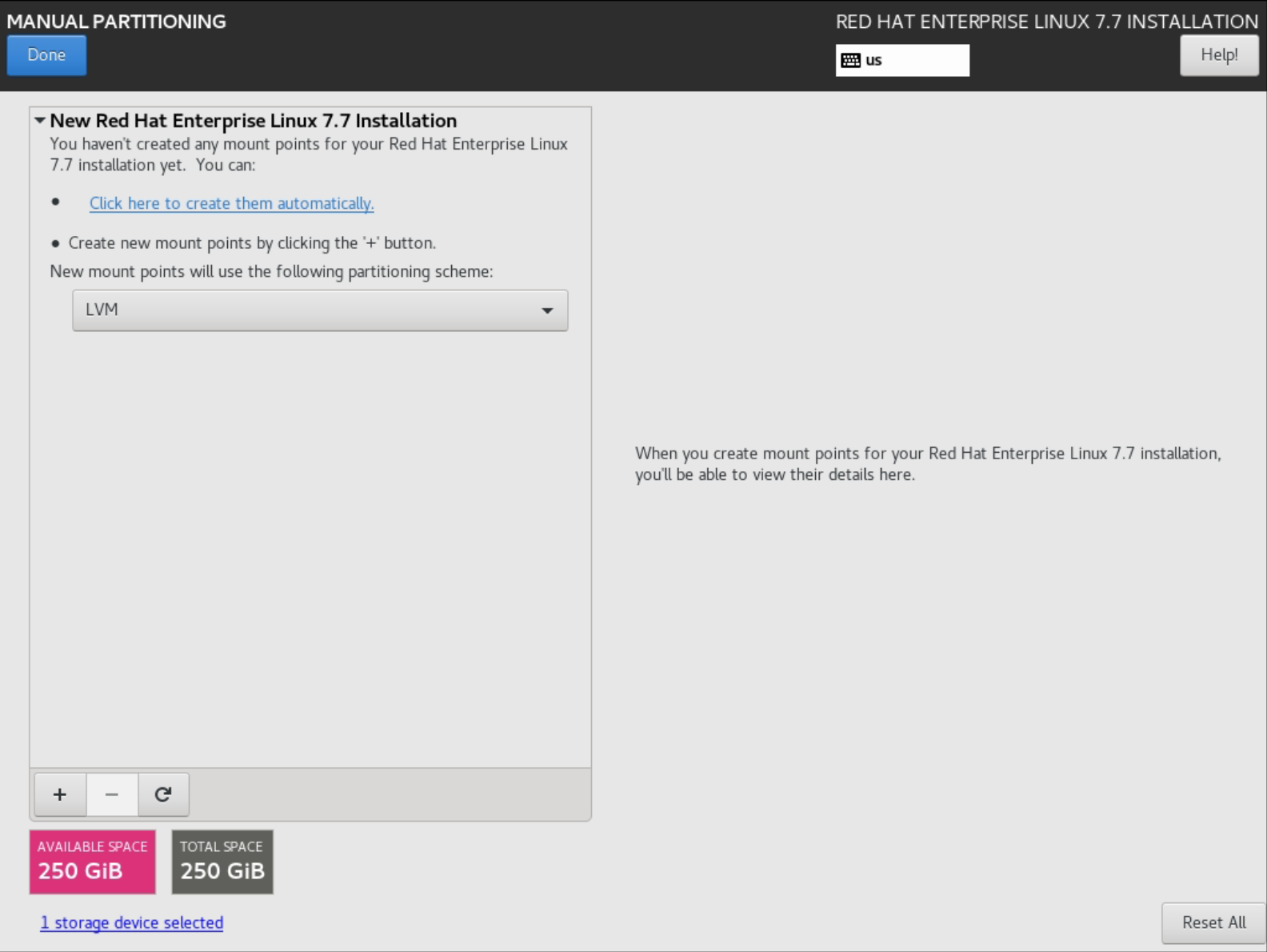
|  |  |
| --- | --- |
| DNS Name | IP |
| bastion.example.internal | 192.168.1.13 |
| support.example.internal  dns.example.internal  nfs.example.internal  yum.example.internal  registry.example.internal  nfs.example. internal  lb.ocp4-1.example.internal  api.ocp4-1.example.internal  api-int.ocp4-1.example.internal  \*.apps.ocp4-1.example.internal | 192.168.1.12 |
| bootstrap.ocp4-1.example.internal | 192.168.1.100 |
| master-0.ocp4-1.example.internal | 192.168.1.101 |
| worker-0.ocp4-1.example.internal | 192.168.1.110 |
| worker-1.ocp4-1.example.internal | 192.168.1.111 |

# 创建Bastion和Support虚机节点

按以下要求创建Bastion和Support虚机，并安装操作系统：

1. 按照“**虚机节点资源配置规划**”中的配置创建Bastion和Support虚拟机。
2. 最小化安装RHEL7.6/RHEL7.7操作系统。
3. 按照“**虚机节点网络配置规**”设置IP/Hostname/Gateway/NETMASK/DNS网络属性。
4. 建议在安装虚拟机的RHEL时**手动配置存储分区**，将“/”分区空间最大化。以support虚机为例：
   * 下图1：选中“I will configure patitioning”
   * 下图2：选择“Click here to create then automatically”
   * 下图3：删除“/home”，同时增加“/”空间，然后点击“Update Settings”，随后点击“Done”





# 获得离线安装文件

## 获得离线安装文件方法1

从以下网盘下载整个OCP-4.5.9目录。

https://pan.baidu.com/s/1TfYGjhDU7AYWw1LERcK6Ow 提取码:hj3b

## 获得离线安装文件方法2

**注意**：本方法必须有OpenShift的正式或试用订阅。

[Bastion]

[root@bastion~]#

### 获取OCP 4.5的最新版本号

export OCP\_VER=$(curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/stable-4.5/release.txt | \grep 'Name:' | awk '{print $NF}')

echo ${OCP\_VER}

### 创建目录

export OCP\_PATH=/data/OCP-${OCP\_VER}/ocp

mkdir -p ${OCP\_PATH}/{app-image,ocp-client,ocp-image,ocp-installer,rhcos,secret} ${OCP\_PATH}/yum

### 准备下载环境

#### 订阅账户登陆

rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

export SUBC\_USER=XXXXX

export SUBC\_PASSWD=XXXXX

subscription-manager register --force --user ${SUBC\_USER} --password ${SUBC\_PASSWD}

subscription-manager refresh

subscription-manager list --available --matches '\*OpenShift\*'

SKU: ES0xxxxxx

Contract: 11929122

Pool ID: 8a85f99xxxxxxxxxxxxxxxxxxxxxxxx

Provides Management: Yes

Available: 19

Suggested: 1

Service Level: Self-Support

Service Type: L1-L3

Subscription Type: Standard

Starts: 05/20/2019

Ends: 05/20/2020

System Type: Physical

subscription-manager attach --pool=<pool\_id>

#### 安装工具软件

subscription-manager repos --disable="\*"

subscription-manager repos \

--enable="rhel-7-server-rpms" \

--enable="rhel-7-server-extras-rpms" \

--enable="rhel-7-server-ose-4.3-rpms"

### 下载离线YUM源

#### 开启RHEL订阅频道

1. 关闭所有预先启用的yum频道

subscription-manager repos --disable="\*"

1. 仅启用与本次部署相关的yum源

subscription-manager repos \

--enable="rhel-7-server-rpms" \

--enable="rhel-7-server-extras-rpms" \

--enable="rhel-7-server-ose-4.3-rpms"

subscription-manager repos --list-enabled

yum -y install yum-utils createrepo

#### 批量下载软件包

for repo in $(subscription-manager repos --list-enabled |grep "Repo ID" |awk '{print $3}'); do

reposync --gpgcheck -lmn --repoid=${repo} --download\_path=${OCP\_PATH}/yum

createrepo -v ${OCP\_PATH}/yum/${repo} -o ${OCP\_PATH}/yum/${repo}

done

#### 检查下载后的软件包容量

du -lh ${OCP\_PATH}/yum --max-depth=1

268M ./rhel-7-server-extras-rpms

449M ./rhel-7-server-ose-4.3-rpms

4.6G ./rhel-7-server-rpms

#### 下载其它工具包

mkdir -p ${OCP\_PATH}/yum/rhel-7-server-addition-rpms

wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm -P ${OCP\_PATH}/yum/rhel-7-server-addition-rpms

rpm -ivh ${OCP\_PATH}/yum/rhel-7-server-addition-rpms/epel-release-latest-7.noarch.rpm

yum install --downloadonly --downloaddir=${OCP\_PATH}/yum/rhel-7-server-addition-rpms jq pv pigz python-pip python36

#### 压缩打包

for dir in $(ls --indicator-style=none ${OCP\_PATH}/yum/); do

tar -zcvf ${OCP\_PATH}/yum/${dir}.tar.gz ${dir};

done

ll -h ${OCP\_PATH}/yum/\*.tar.gz

-rw-r--r--. 1 root root 259M Oct 31 17:51 rhel-7-server-extras-rpms.tar.gz

-rw-r--r--. 1 root root 861M Oct 31 17:51 rhel-7-server-ose-4.3-rpms.tar.gz

-rw-r--r--. 1 root root 4710M Oct 31 17:51 rhel-7-server-rpms.tar.gz

#### 清除文件并退出订阅

rm -rf $(ls ${OCP\_PATH}/yum |egrep -v gz)

subscription-manager unregister

### 下载并安装OCP客户端

wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/${OCP\_VER}/openshift-client-linux-${OCP\_VER}.tar.gz \

-P ${OCP\_PATH}/ocp-client

tar -xzf ${OCP\_PATH}/ocp-client/openshift-client-linux-${OCP\_VER}.tar.gz -C /usr/local/sbin/

oc version

Client Version: 4.5.9

### 下载openshift-install

wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/${OCP\_VER}/openshift-install-linux-${OCP\_VER}.tar.gz \

-P ${OCP\_PATH}/ocp-installer

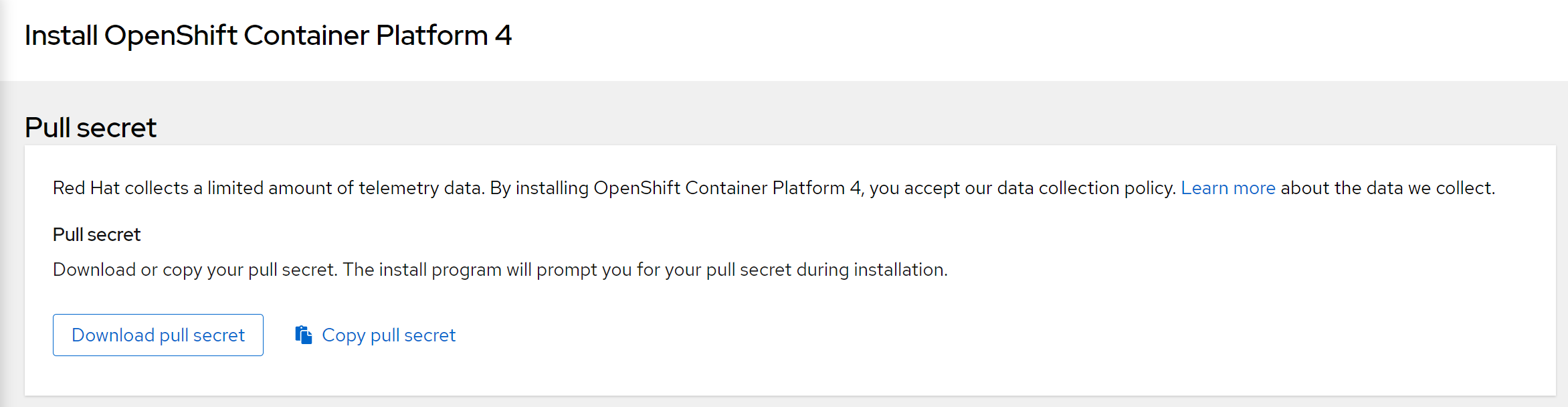
ll -h ${OCP\_PATH}/ocp-installer

-rw-r--r--. 1 root root 79M Mar 2 16:44 openshift-install-linux-4.5.9.tar.gz

### 下载OCP核心镜像

#### 获取下载镜像所需密钥

使用RedHat订阅账号登陆https://cloud.redhat.com/openshift/install/pull-secret，将图中secret复制到${OCP\_PATH}/secret/redhat-secret.json文件中。



**注意**：该文件用来从RedHat网站下载镜像，其有效期只有24小时。在失效前如没有OpenShift完成安装，需要重新下载。

配置指向redhat-secret.json文件的环境变量。

export REDHAT\_SECRET=${OCP\_PATH}/secret/redhat-secret.json

jq . ${REDHAT\_SECRET}

{

"auths": {

"cloud.openshift.com": {

"auth": "xxxxxxx",

"email": "xxx@xxx.com"

},

"quay.io": {

"auth": "xxxxxxxx",

"email": "lxxx@xxx.com"

},

"registry.connect.redhat.com": {

"auth": "xxxxxxxx",

"email": "xxx@xxx.com"

},

"registry.redhat.io": {

"auth": "xxxxxx",

"email": "xxx@xxx.com"

}

}

}

#### 环境变量配置

export PRODUCT\_REPO=openshift-release-dev

export RELEASE\_NAME=ocp-release

#### 验证待下载的镜像信息

oc adm release info "quay.io/${PRODUCT\_REPO}/${RELEASE\_NAME}:${OCP\_VER}-x86\_64"

Name: 4.5.9

Digest: sha256:1f0fd38ac0640646ab8e7fec6821c8928341ad93ac5ca3a48c513ab1fb63bc4b

Created: 2020-08-28T08:23:43Z

OS/Arch: linux/amd64

Manifests: 366

Pull From: quay.io/openshift-release-dev/ocp-release@sha256:1f0fd38ac0640646ab8e7fec6821c8928341ad93ac5ca3a48c513ab1fb63bc4b

Release Metadata:

Version: 4.5.9

Upgrades: 4.5.1, 4.5.2, 4.5.3, 4.5.5

Metadata:

description:

Metadata:

url: https://access.redhat.com/errata/RHBA-2020:1529

Component Versions:

kubernetes 1.18.3

Images:

NAME DIGEST

aws-machine-controllers sha256:23cbad0f3cdeae142a35376a447b74b93a60016bec012d6abcf95cf55cd7878e

azure-machine-controllers sha256:6d094bcc6584e136dfd488467c9dd02cc92244b6f9e7d121a628425ee7a6e9a5

baremetal-installer sha256:859ea095b9953369be5a0162309cbde3daafccb59c76facc0a8a7118a928b7fe

baremetal-machine-controllers sha256:6f2a7ee4191e915dc2a865fd1fdccfbfd42c86402976cab72bcd5350804cc33  
…

#### 启动镜像下载

oc adm release mirror -a ${REDHAT\_SECRET} \

--from=quay.io/${PRODUCT\_REPO}/${RELEASE\_NAME}:${OCP\_VER}-x86\_64 --to-dir=${OCP\_PATH}/ocp-image/mirror\_${OCP\_VER}

...

sha256:9cfbc2cab21db414106a348ca0b79ed6b2e26efdefb34fac10aca68c4030d492 file://openshift/release:4.5.9-cluster-version-operator

sha256:1c2e9ea65264bd94dab98617700f43ee1447d78e78d89ca097169df9cc7f6bbe file://openshift/release:4.5.9-openshift-controller-manager

sha256:dd28e8cb53f368619e33aa19803d2b1e5ace9d8e5ce1f10e3cd8e3f7c71da8a0 file://openshift/release:4.5.9-operator-lifecycle-manager

sha256:11eb4ad94024d0bac06348c82c3ebb8e29e4414b1ef5297a5137b641cb01681a file://openshift/release:4.5.9-prometheus-config-reloader

sha256:5b1cfb3b3e13e44060d018db0ae644695e8ab431cf06e03dea654a628de207e7 file://openshift/release:4.5.9-multus-admission-controller

sha256:032c6e1893edbea6e932ddca3579c7095fac757b929e30cb94ce2565dc8918fd file://openshift/release:4.5.9-multus-cni

info: Mirroring completed in 2h42m47.89s (579.5kB/s)

Success

Update image: openshift/release:4.5.9

To upload local images to a registry, run:

oc image mirror --from-dir=/data/OCP-4.5.9/ocp-image/ocp-image-4.5.9 file://openshift/release:4.5.9\* REGISTRY/REPOSITORY

如果出现如下错误输出，则重新运行上述镜像命令，直至上述输出结果的Success状态

error: unable to copy layer sha256:fcd63ccfdd0ccd78cc0b23a6e9cef0c400625c1506f3eed694d92c7ced04b4c4 to file://openshift/release: rename mirror\_4.5.9/v2/openshift/release/blobs/sha256:fcd63ccfdd0ccd78cc0b23a6e9cef0c400625c1506f3eed694d92c7ced04b4c4.download mirror\_4.5.9/v2/openshift/release/blobs/sha256:fcd63ccfdd0ccd78cc0b23a6e9cef0c400625c1506f3eed694d92c7ced04b4c4: no such file or directory

info: Mirroring completed in 2m33.72s (36.77MB/s)

error: one or more errors occurred while uploading images

#### 检查下载镜像的有效性

oc adm release info --dir=${OCP\_PATH}/ocp-image/mirror\_${OCP\_VER} file://openshift/release:${OCP\_VER}

Name: 4.5.9

Digest: sha256:64320fbf95d968fc6b9863581a92d373bc75f563a13ae1c727af37450579f61a

Created: 2020-09-06T12:05:47Z

OS/Arch: linux/amd64

Manifests: 366

Pull From: file://openshift/release@sha256:64320fbf95d968fc6b9863581a92d373bc75f563a13ae1c727af37450579f61a

Release Metadata:

Version: 4.5.9

Upgrades: 4.5.1, 4.5.2, 4.5.3, 4.5.5, 4.5.7

Metadata:

description:

Metadata:

url: https://access.redhat.com/errata/RHBA-2020:0676

Component Versions:

Kubernetes 1.18.3

Images:

NAME DIGEST

aws-machine-controllers sha256:9be189b8cdd9ee4bed0125613d732167c258bee78eddace0c3889b88dfd49559

azure-machine-controllers sha256:f1f3fa8e7912113284e06dd03ef516b47ab99483aaeb8104abc20330fc49bee5

baremetal-installer sha256:68d0a192cfc5f2967a14a9722c71ade1ac08101cbf945e4b2909ef8e6f5c3bda

baremetal-machine-controllers sha256:6aac729eca9c5e52b7a251197c6cbada0f4e7089645509968d09ba2756ca45c6

baremetal-operator sha256:b7e88a68aaa44b10933513534f9cdf0344caffa6580ffd44baacf380537c7226

baremetal-runtimecfg sha256:e468469f3f999258af00982ab47bfdf3c9ade0fef1df342cebd2d08881afe9a3

cli sha256:cc073e633967eeed8f8cdc3ae3fc5e4dba30af8a9462cd6858e1404a51c439ad

cli-artifacts sha256:ed8e405a44288bdfeda5f687c80daeee088147b4bef30eb38dd35c956c12c5d7

cloud-credential-operator sha256:c49c5d55a6d1c26d10ad6bd086ccfcf1d168526a7dc44f8ecdf96debb8bf1c8a

cluster-authentication-operator sha256:7217159e39a6ad787c820ac614f8d9d29dc5c312e53fb806f2a95f81254c624f

cluster-autoscaler sha256:0a5d231e35932f6250ed574d004b2786fa58464e1a4e5a107ddc4170c7ebef3c

cluster-autoscaler-operator sha256:5aca2b60800d51b727a205f1d6ee3a11d1c21337df0fc61530a16f7e9da75064

cluster-bootstrap sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b

cluster-config-operator sha256:1b7412f8e133acdfbc1de64c22a57a98689f70b1ff49a072da4b5402127924fa

cluster-dns-operator sha256:37e36726940cd93d02e58fddea538895e571062b5b8ba85149c398ddf3dd45f9

cluster-image-registry-operator sha256:e7dca90c83359f83bf6e6a40d374da2d5718ee0f39668ce13b51c7cac4602d04

cluster-ingress-operator sha256:84efa8f8d09642e767655bf20f652ae1de9bbfa85c3970fb41348009344e463d

cluster-kube-apiserver-operator sha256:048fffae544908d78aef0734bb1e92488384f87c7486cc50186679551d7aef2b

cluster-kube-controller-manager-operator sha256:447761b963d10216a3ea4fa25308b315859038eade322ba279d167a797572f7a

cluster-kube-scheduler-operator sha256:a24ecdc0ea3262fb60166d324b564b4b53962b914e21323836342a6dbb49e5af

cluster-machine-approver sha256:a7a1c4d95b6cf2c3cab4631627317cb8a23096e92380829e872043103db40ed7

cluster-monitoring-operator sha256:23bdb380f503605e7ba52ed1e4e9786f75fe5c829d9e20e995a063ab0e203bf2

cluster-network-operator sha256:a48c2f750ef9277c30546a8284603fe6f87b2f42afc32b3dbd1fca040a3ed8b4

cluster-node-tuned sha256:38f1f65811f3ebf4f19217b73fa9b7cb5da37c4009b24f36e18be875d8b5258a

cluster-node-tuning-operator sha256:fc1e38d12f03544e2e3e3f898d6246a2896dc57e77d5c0bfca965c9d9b56a75d

cluster-openshift-apiserver-operator sha256:cd393d8515bae7a07ee4d15e753e1cd10e46b02170626d0adec0dbfc76423330

cluster-openshift-controller-manager-operator sha256:5daecb6c1890df65759ffa123a2614ad56db11977f8fd7ee33918bcc20e943bb

cluster-policy-controller sha256:15c8d339541f064471e0a2902a4e6085be8f0b95e5f9d5c5fe405b3479a62c9d

cluster-samples-operator sha256:ff041c6cfde2ac918a2fe71b19c34420acf8f2c4a29d4a2e41df95f0cab59d50

cluster-storage-operator sha256:4c36caadce0bf8da22f81930c0c483d580100eaa80d41f5b32f30a973d5dece9

cluster-svcat-apiserver-operator sha256:5d75e400e41d4bbd530211f77ca58158ba322823bf54bb69cbbc59175b14aa41

cluster-svcat-controller-manager-operator sha256:3c37c02c7366fb2b22c07b0e72b7a160afab70bf839b8148b97008c5ea962513

cluster-update-keys sha256:9dfb0201a84fe90867717074a214da70c27283b748c0d3af954939edb354267e

cluster-version-operator sha256:9cfbc2cab21db414106a348ca0b79ed6b2e26efdefb34fac10aca68c4030d492

configmap-reloader sha256:dc0b63f2c321bf8a6c2a709c9faa42585cfa00b123f2a69c34bf91b9d019191f

console sha256:f7dbeb00c9ae47d6196aeb56b48b5590e209e85cccab39edf000bcad8bba89e4

console-operator sha256:a56a696c6e62ed8aa64283033d3140dbb24088de0ad8a3cded57b9b9a64bb612

container-networking-plugins sha256:55d45c17fcc1c635930b406817ed0c49788816c57399752fb03984def7c4298a

coredns sha256:d5a7c3a5346e1b411e27a8e22739e2b0b0f6b1eae61bbebb004cae703d76e63a

deployer sha256:871d962d57a1b43d6ceef2ce2906ad8dd926ce9d0b4bd5db99f96feb1ae355d6

docker-builder sha256:a079d14d21864283f048719b9ad8454c81a5e0ea79a71ca50446119345a84a66

docker-registry sha256:1ac97760bae00723c64864fdbc8c916298473a759fa8725dc21fb6fff23f1c4c

etcd sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3

gcp-machine-controllers sha256:a34335aa2f8faafca681d328da297a537ed97a294a730818b9803fb5ffce139c

grafana sha256:58187cb5209fe28d5f57649786e03c28ea9569819973460d4e3deeef8d3635fa

haproxy-router sha256:31bde5d258a4b3401ee730d9d87bbb66a1694c45f6cd1423802f10e8ce969b61

hyperkube sha256:abb1fc9d598bc113802dc344fb4391ac52b5fb3b43e413f8510b5f06dd96443e

insights-operator sha256:184d91726f7f5c6977c54840406d382706d31da5c37b327a0ad455753a4cde3e

installer sha256:7ac4e9113c104ae57740d2bb8e7ff7bec8ded91425a6405c7e8d1c9202ddb7e6

installer-artifacts sha256:4368cfa1c144c516be6ab9a70fab57ddfc3495ff6e060a6ef783442c55f5cd2e

ironic sha256:0a6b9de0a8337e07f6bf047c34ed107f80816e9b832ceaf5a7d1b0235f73740a

ironic-hardware-inventory-recorder sha256:324c88d9635cccf7ae578402f8dacc9e230eb2f9d234dc00bd78333d46dcd826

ironic-inspector sha256:f323ae4d47dc5aad7e36d94fd2a9ee6b9a928ecc8c1a8d99c1241797407986bb

ironic-ipa-downloader sha256:b17fa1125b5104cd5175a43c48f7f13261653c23b50aa403d8674972eb1ee908

ironic-machine-os-downloader sha256:464455fbbd97be1acb13d4256be78f18032931d18cb40a4287a284f49e017633

ironic-static-ip-manager sha256:0f0a3c764f315255ac5bee8f9934b4e339d28938ff43e36f4c44c4155f09c695

jenkins sha256:a5fe1d551f63d3b8bb7bc06933018a2c0010602a801d91253af9a44620b5f112

jenkins-agent-maven sha256:d54176211b27eee74e6d957a3c1f2149080f6aeed2995e1f4337f89c529ce95d

jenkins-agent-nodejs sha256:ca814925849e9d3fd3bb67de2e747a413ece27ca3cd13f312ec2c4b62d876faf

k8s-prometheus-adapter sha256:33e80b1706acb1cba44e47e4d00642c356eed3d735375c2b57616cc48b5a7ec5

keepalived-ipfailover sha256:64dd8aa0884ed5a2a594931937f3e09b0f6f35e1569dfd70cbfcda631597ab83

kube-client-agent sha256:016b572b6da4528e8a8aadc89dc52b67f3894b246c060c0d7843b0448aefb644

kube-etcd-signer-server sha256:8c364dc68a8a58e7f9867dd04cbcfe7427fda3c418440b986daeee348fa017be

kube-proxy sha256:38154bdd61a945c1e1ffd92666ad8c7b16e9d69084eeb05b5f11c581bd2ae5ff

kube-rbac-proxy sha256:c9818acc6c6cbe0235b76bbf5bbf92c15fa07f9ef4016356cdf02441edb5c8d2

kube-state-metrics sha256:f71c74ccc93e716a6080617094fe2bea88b365a198d92bcfa322f833f3474b3e

kuryr-cni sha256:7b7abcad7a07f676778a9fbc2ba57aad9c1cde20de8bdefc755ecd35e1814e60

kuryr-controller sha256:d340ccda7084c2365ceb628196be3af1ab512da08f058c49fb82d3bc665ab7ec

libvirt-machine-controllers sha256:8e5623c11e122707972291d2f978ee331409c36398454d249c246bbb8f64755f

local-storage-static-provisioner sha256:8e776ad883195f6736d57eade109c232dfe05a3eaffa6f19cd34b09f0483887f

machine-api-operator sha256:c143fe98d618d14861325e3780e6fcc9fdcef1dbb3d59256967192b61ec56966

machine-config-operator sha256:ea5cdeb8588fc8f540db4d1e976a9adc09b283b24a9dd29fa201cd999755c8d5

machine-os-content sha256:6d83a4e72ae352600de3957b2c400e1eb88e8efdde9540ca59a900f35e7c44ee

mdns-publisher sha256:afcce680870f916d48b1305b7fbc6b2fd71d57d3b8faabfbb4c51dc9f4d251b5

multus-admission-controller sha256:5b1cfb3b3e13e44060d018db0ae644695e8ab431cf06e03dea654a628de207e7

multus-cni sha256:032c6e1893edbea6e932ddca3579c7095fac757b929e30cb94ce2565dc8918fd

must-gather sha256:f1ffcdcbd684afd61ff1874b47e1c61a0f7adab93b7a21123a1e29b041d3dabf

oauth-proxy sha256:e2cc56931552bee158ed77d2f0a7e75202aba9204612f370d08630ba3992ec77

oauth-server sha256:b8b168dc91550c49f604b4629ec15a714f91731c5cd95408ab70f48b0ebfabb4

openshift-apiserver sha256:ee3f3657bef2f21492ba59f559ac0eb752e2a79a4efcdbf410957b83d3eebb9e

openshift-controller-manager sha256:1c2e9ea65264bd94dab98617700f43ee1447d78e78d89ca097169df9cc7f6bbe

openshift-state-metrics sha256:dac3749820e5877af3a8fc18c82a94eaa0a7da4363873e2a11a6ad985c2444e8

openstack-machine-controllers sha256:dd11e6d2a84e83fe1e27451bf798b8a249a3f151e132ce0ed5b679042bac496d

operator-lifecycle-manager sha256:dd28e8cb53f368619e33aa19803d2b1e5ace9d8e5ce1f10e3cd8e3f7c71da8a0

operator-marketplace sha256:b9abf9dd449ae8abcb3304707d69bdf2619af3b9a4b882d5b7cfa99416522aca

operator-registry sha256:68763083b100b5d82739dedb08d9c154cd3c8ee0435ed4fad5674cb9b013aec9

ovn-kubernetes sha256:139b355438153c7f2674d9ce67166d8ecb5b225f8f0223143d1c6f2bfcb54902

pod sha256:4674bc6d9907cd548a2e8cc6ba8fa8ec66963c221dbe9dda6bf19df579b9d828

prom-label-proxy sha256:bf6c117d16261065626be99bd3a3865db5a9c42eb04448efbf83cab6d5fdeddf

prometheus sha256:da24aebff8becac0d91fa5847703681694d71e77fc2e8a53068c211ae2d9ee47

prometheus-alertmanager sha256:4b5db4653b80fb2c8eb2c17ba466921e34c473ebfa144b2ac8943996fada7685

prometheus-config-reloader sha256:11eb4ad94024d0bac06348c82c3ebb8e29e4414b1ef5297a5137b641cb01681a

prometheus-node-exporter sha256:ec64199ad47081881e762e56269dfee7afac5660e0e22dc4f92e617af395546d

prometheus-operator sha256:f4616681b57c8eb9b3233b0a3c9cb933ac9010d34862d927c6a51868de707dd0

sdn sha256:b8ae77b8597232c348acd86c922b25d93f8784d9da1158c75d986feba2f49cf5

service-ca-operator sha256:cf3b599d9313e39ecb2b37e37bfe400de0968cd7a03be178156d1e038773ad76

service-catalog sha256:054f850ea37b5d362f500ac5ad0f0f20d9275e128aef5c654746d1bd2110821b

telemeter sha256:2ea67997eda7bd7920d8d872b1910aef6fd3902daa6ca0358182cfeadfb0e441

tests sha256:e2549abff65abff49471271148e6b04fe8bfd18eb4ca912782a2c50ed0cc98de

thanos sha256:2aad2c49a526c505762498fb745ebf2604b480c8945cd887ae2ac18b5ac963a8

#### 镜像打包

tar -zcvf ${OCP\_PATH}/ocp-image/ocp-image-${OCP\_VER}.tar -C ${OCP\_PATH}/ocp-image ./mirror\_${OCP\_VER}

#### 清理

rm -rf ${OCP\_PATH}/ocp-image/mirror\_${OCP\_VER}

### 下载离线ImageStream镜像包

#### 创建镜像列表文件

mkdir -p ${OCP\_PATH}/app-image/redhat-app

将应用镜像名输入到文件中，如下所示。以下是RedHat提供的应用镜像，可根据需要删减。

cat << EOF > ${OCP\_PATH}/app-image/redhat-app/app-images.txt

registry.redhat.io/3scale-amp21/apicast-gateway:1.4-2

registry.redhat.io/3scale-amp22/apicast-gateway:1.8

registry.redhat.io/3scale-amp23/apicast-gateway:latest

registry.redhat.io/3scale-amp24/apicast-gateway:latest

registry.redhat.io/3scale-amp25/apicast-gateway:latest

registry.redhat.io/3scale-amp26/apicast-gateway:latest

registry.redhat.io/devtools/go-toolset-rhel7:1.11.5

registry.redhat.io/dotnet/dotnet-21-rhel7:2.1

registry.redhat.io/dotnet/dotnet-21-runtime-rhel7:2.1

registry.redhat.io/dotnet/dotnet-22-rhel7:2.2

registry.redhat.io/dotnet/dotnet-22-runtime-rhel7:2.2

registry.redhat.io/dotnet/dotnet-30-rhel7:3.0

registry.redhat.io/dotnet/dotnet-30-runtime-rhel7:3.0

registry.redhat.io/fuse7/fuse-apicurito:1.2

registry.redhat.io/fuse7/fuse-apicurito:1.3

registry.redhat.io/fuse7/fuse-apicurito:1.4

registry.redhat.io/fuse7/fuse-apicurito-generator:1.2

registry.redhat.io/fuse7/fuse-apicurito-generator:1.3

registry.redhat.io/fuse7/fuse-apicurito-generator:1.4

registry.redhat.io/fuse7/fuse-console:1.0

registry.redhat.io/fuse7/fuse-console:1.0

registry.redhat.io/fuse7/fuse-console:1.1

registry.redhat.io/fuse7/fuse-console:1.2

registry.redhat.io/fuse7/fuse-console:1.3

registry.redhat.io/fuse7/fuse-console:1.4

registry.redhat.io/fuse7/fuse-eap-openshift:1.0

registry.redhat.io/fuse7/fuse-eap-openshift:1.0

registry.redhat.io/fuse7/fuse-eap-openshift:1.1

registry.redhat.io/fuse7/fuse-eap-openshift:1.2

registry.redhat.io/fuse7/fuse-eap-openshift:1.3

registry.redhat.io/fuse7/fuse-eap-openshift:1.4

registry.redhat.io/fuse7/fuse-java-openshift:1.0

registry.redhat.io/fuse7/fuse-java-openshift:1.0

registry.redhat.io/fuse7/fuse-java-openshift:1.1

registry.redhat.io/fuse7/fuse-java-openshift:1.2

registry.redhat.io/fuse7/fuse-java-openshift:1.3

registry.redhat.io/fuse7/fuse-java-openshift:1.4

registry.redhat.io/fuse7/fuse-karaf-openshift:1.0

registry.redhat.io/fuse7/fuse-karaf-openshift:1.0

registry.redhat.io/fuse7/fuse-karaf-openshift:1.1

registry.redhat.io/fuse7/fuse-karaf-openshift:1.2

registry.redhat.io/fuse7/fuse-karaf-openshift:1.3

registry.redhat.io/fuse7/fuse-karaf-openshift:1.4

registry.redhat.io/jboss-amq-6/amq62-openshift:1.1

registry.redhat.io/jboss-amq-6/amq62-openshift:1.2

registry.redhat.io/jboss-amq-6/amq62-openshift:1.3

registry.redhat.io/jboss-amq-6/amq62-openshift:1.4

registry.redhat.io/jboss-amq-6/amq62-openshift:1.5

registry.redhat.io/jboss-amq-6/amq62-openshift:1.6

registry.redhat.io/jboss-amq-6/amq62-openshift:1.7

registry.redhat.io/jboss-amq-6/amq63-openshift:1.0

registry.redhat.io/jboss-amq-6/amq63-openshift:1.1

registry.redhat.io/jboss-amq-6/amq63-openshift:1.2

registry.redhat.io/jboss-amq-6/amq63-openshift:1.3

registry.redhat.io/jboss-amq-6/amq63-openshift:1.4

registry.redhat.io/jboss-datagrid-6/datagrid65-client-openshift:1.0

registry.redhat.io/jboss-datagrid-6/datagrid65-client-openshift:1.1

registry.redhat.io/jboss-datagrid-6/datagrid65-openshift:1.2

registry.redhat.io/jboss-datagrid-6/datagrid65-openshift:1.3

registry.redhat.io/jboss-datagrid-6/datagrid65-openshift:1.4

registry.redhat.io/jboss-datagrid-6/datagrid65-openshift:1.5

registry.redhat.io/jboss-datagrid-6/datagrid65-openshift:1.6

registry.redhat.io/jboss-datagrid-7/datagrid71-client-openshift:1.0

registry.redhat.io/jboss-datagrid-7/datagrid71-openshift:1.0

registry.redhat.io/jboss-datagrid-7/datagrid71-openshift:1.1

registry.redhat.io/jboss-datagrid-7/datagrid71-openshift:1.2

registry.redhat.io/jboss-datagrid-7/datagrid71-openshift:1.3

registry.redhat.io/jboss-datagrid-7/datagrid72-openshift:1.0

registry.redhat.io/jboss-datagrid-7/datagrid72-openshift:1.1

registry.redhat.io/jboss-datagrid-7/datagrid72-openshift:1.2

registry.redhat.io/jboss-datagrid-7/datagrid73-openshift:1.0

registry.redhat.io/jboss-datagrid-7/datagrid73-openshift:1.1

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.0

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.1

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.2

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.3

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.4

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.5

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.6

registry.redhat.io/jboss-datavirt-6/datavirt64-driver-openshift:1.7

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.0

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.1

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.2

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.3

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.4

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.5

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.6

registry.redhat.io/jboss-datavirt-6/datavirt64-openshift:1.7

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.0

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.1

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.2

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.3

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.4

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.5

registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:1.6

registry.redhat.io/jboss-eap-6/eap64-openshift:1.1

registry.redhat.io/jboss-eap-6/eap64-openshift:1.2

registry.redhat.io/jboss-eap-6/eap64-openshift:1.3

registry.redhat.io/jboss-eap-6/eap64-openshift:1.4

registry.redhat.io/jboss-eap-6/eap64-openshift:1.5

registry.redhat.io/jboss-eap-6/eap64-openshift:1.6

registry.redhat.io/jboss-eap-6/eap64-openshift:1.7

registry.redhat.io/jboss-eap-6/eap64-openshift:1.8

registry.redhat.io/jboss-eap-6/eap64-openshift:1.9

registry.redhat.io/jboss-eap-6/eap64-openshift:latest

registry.redhat.io/jboss-eap-7/eap70-openshift:1.3

registry.redhat.io/jboss-eap-7/eap70-openshift:1.4

registry.redhat.io/jboss-eap-7/eap70-openshift:1.5

registry.redhat.io/jboss-eap-7/eap70-openshift:1.6

registry.redhat.io/jboss-eap-7/eap70-openshift:1.7

registry.redhat.io/jboss-eap-7/eap71-openshift:1.1

registry.redhat.io/jboss-eap-7/eap71-openshift:1.2

registry.redhat.io/jboss-eap-7/eap71-openshift:1.3

registry.redhat.io/jboss-eap-7/eap71-openshift:1.4

registry.redhat.io/jboss-eap-7/eap71-openshift:latest

registry.redhat.io/jboss-eap-7/eap72-openshift:1.0

registry.redhat.io/jboss-eap-7/eap72-openshift:1.1

registry.redhat.io/jboss-eap-7/eap72-openshift:latest

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift:12.0

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift:13.0

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift:14.0

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift:15.0

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift:16.0

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift-rhel8:17.0

registry.redhat.io/jboss-eap-7-tech-preview/eap-cd-openshift-rhel8:latest

registry.redhat.io/jboss-fuse-6/fis-java-openshift:1.0

registry.redhat.io/jboss-fuse-6/fis-java-openshift:2.0

registry.redhat.io/jboss-fuse-6/fis-karaf-openshift:1.0

registry.redhat.io/jboss-fuse-6/fis-karaf-openshift:2.0

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.0

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.1

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.2

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.3

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.4

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.5

registry.redhat.io/jboss-processserver-6/processserver64-openshift:1.6

registry.redhat.io/jboss-webserver-3/webserver30-tomcat7-openshift:1.1

registry.redhat.io/jboss-webserver-3/webserver30-tomcat7-openshift:1.2

registry.redhat.io/jboss-webserver-3/webserver30-tomcat7-openshift:1.3

registry.redhat.io/jboss-webserver-3/webserver30-tomcat8-openshift:1.1

registry.redhat.io/jboss-webserver-3/webserver30-tomcat8-openshift:1.2

registry.redhat.io/jboss-webserver-3/webserver30-tomcat8-openshift:1.3

registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:1.0

registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:1.1

registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:1.2

registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:1.3

registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:1.4

registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:1.0

registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:1.1

registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:1.2

registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:1.3

registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:1.4

registry.redhat.io/jboss-webserver-5/webserver50-tomcat9-openshift:1.0

registry.redhat.io/jboss-webserver-5/webserver50-tomcat9-openshift:1.1

registry.redhat.io/jboss-webserver-5/webserver50-tomcat9-openshift:1.2

registry.redhat.io/jboss-webserver-5/webserver50-tomcat9-openshift:latest

registry.redhat.io/openjdk/openjdk-11-rhel7:1.0

registry.redhat.io/openjdk/openjdk-11-rhel7:latest

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:1.0

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:1.1

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:1.2

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:1.3

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:1.4

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:1.5

registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:latest

registry.redhat.io/redhat-sso-7/sso70-openshift:1.3

registry.redhat.io/redhat-sso-7/sso70-openshift:1.4

registry.redhat.io/redhat-sso-7/sso71-openshift:1.0

registry.redhat.io/redhat-sso-7/sso71-openshift:1.1

registry.redhat.io/redhat-sso-7/sso71-openshift:1.2

registry.redhat.io/redhat-sso-7/sso71-openshift:1.3

registry.redhat.io/redhat-sso-7/sso72-openshift:1.0

registry.redhat.io/redhat-sso-7/sso72-openshift:1.1

registry.redhat.io/redhat-sso-7/sso72-openshift:1.2

registry.redhat.io/redhat-sso-7/sso72-openshift:1.3

registry.redhat.io/redhat-sso-7/sso72-openshift:1.4

registry.redhat.io/redhat-sso-7/sso73-openshift:1.0

registry.redhat.io/redhat-sso-7/sso73-openshift:1.0

registry.redhat.io/rhdm-7/rhdm74-decisioncentral-openshift:1.0

registry.redhat.io/rhdm-7/rhdm74-decisioncentral-openshift:1.1

registry.redhat.io/rhdm-7/rhdm74-kieserver-openshift:1.0

registry.redhat.io/rhdm-7/rhdm74-kieserver-openshift:1.1

registry.redhat.io/rhdm-7-tech-preview/rhdm74-optaweb-employee-rostering-openshift:1.0

registry.redhat.io/rhdm-7-tech-preview/rhdm74-optaweb-employee-rostering-openshift:1.1

registry.redhat.io/rhoar-nodejs/nodejs-10:latest

registry.redhat.io/rhoar-nodejs/nodejs-8:latest

registry.redhat.io/rhoar-nodejs-tech-preview/rhoar-nodejs-10-webapp:latest

registry.redhat.io/rhpam-7/rhpam74-businesscentral-monitoring-openshift:1.0

registry.redhat.io/rhpam-7/rhpam74-businesscentral-monitoring-openshift:1.1

registry.redhat.io/rhpam-7/rhpam74-businesscentral-openshift:1.0

registry.redhat.io/rhpam-7/rhpam74-businesscentral-openshift:1.1

registry.redhat.io/rhpam-7/rhpam74-kieserver-openshift:1.0

registry.redhat.io/rhpam-7/rhpam74-kieserver-openshift:1.1

registry.redhat.io/rhpam-7/rhpam74-smartrouter-openshift:1.0

registry.redhat.io/rhpam-7/rhpam74-smartrouter-openshift:1.1

registry.redhat.io/rhscl/httpd-24-rhel7:latest

registry.redhat.io/rhscl/mariadb-102-rhel7:latest

registry.redhat.io/rhscl/mongodb-32-rhel7:latest

registry.redhat.io/rhscl/mongodb-34-rhel7:latest

registry.redhat.io/rhscl/mongodb-36-rhel7:latest

registry.redhat.io/rhscl/mysql-57-rhel7:latest

registry.redhat.io/rhscl/mysql-80-rhel7:latest

registry.redhat.io/rhscl/nginx-110-rhel7:latest

registry.redhat.io/rhscl/nginx-112-rhel7:latest

registry.redhat.io/rhscl/nodejs-10-rhel7:latest

registry.redhat.io/rhscl/nodejs-8-rhel7:latest

registry.redhat.io/rhscl/perl-524-rhel7:latest

registry.redhat.io/rhscl/perl-526-rhel7:latest

registry.redhat.io/rhscl/php-70-rhel7:latest

registry.redhat.io/rhscl/php-71-rhel7:latest

registry.redhat.io/rhscl/php-72-rhel7:latest

registry.redhat.io/rhscl/postgresql-10-rhel7:latest

registry.redhat.io/rhscl/postgresql-96-rhel7:latest

registry.redhat.io/rhscl/python-27-rhel7:latest

registry.redhat.io/rhscl/python-35-rhel7:latest

registry.redhat.io/rhscl/python-36-rhel7:latest

registry.redhat.io/rhscl/redis-32-rhel7:latest

registry.redhat.io/rhscl/ruby-23-rhel7:latest

registry.redhat.io/rhscl/ruby-24-rhel7:latest

registry.redhat.io/rhscl/ruby-25-rhel7:latest

EOF

#### 测试下载

先以其中一个imagestream为例，测试下载过程是否正常

oc image mirror -a ${REDHAT\_SECRET} --filter-by-os=linux/amd64 registry.redhat.io/rhscl/ruby-25-rhel7:latest \

--dir=${OCP\_PATH}/app-image/redhat-app/images file://rhscl/ruby-25-rhel7:latest

<dir>

ruby-25-rhel7

blobs:

registry.access.redhat.com/rhscl/ruby-25-rhel7 sha256:cf5693de4d3cdd6f352978b87c8f89ead294eff44938598f57a91cf7a02417d2 1.582KiB

registry.access.redhat.com/rhscl/ruby-25-rhel7 sha256:1ae2a9485dabcc0a8d6515ab31aae576cd353c819ccf1487e3026dfec82f68f4 7.3KiB

registry.access.redhat.com/rhscl/ruby-25-rhel7 sha256:fd542ee251592bb3ed566aa42ccf98987a5607b64823720d19d68ae382e48bb0 6.876MiB

registry.access.redhat.com/rhscl/ruby-25-rhel7 sha256:f56c75ce461b282857e724ab7d07a223537f6ce6a5a4ad32900b6ea14787e6f9 13.46MiB

registry.access.redhat.com/rhscl/ruby-25-rhel7 sha256:23302e52b49d49a0a25da8ea870bc1973e7d51c9b306f3539cd397318bd8b0a5 72.73MiB

registry.access.redhat.com/rhscl/ruby-25-rhel7 sha256:63abcb3c00123e524cf21f49a099025395236c9c7e525d9ccaa0d8b42060a104 83.76MiB

manifests:

sha256:eb9e68f88d4c6a43624b965129f010ca5d1e5e2b662c76d9dcd808f738d7940c -> latest

stats: shared=0 unique=6 size=176.8MiB ratio=1.00

phase 0:

ruby-25-rhel7 blobs=6 mounts=0 manifests=1 shared=0

info: Planning completed in 910ms

uploading: file://ruby-25-rhel7 sha256:63abcb3c00123e524cf21f49a099025395236c9c7e525d9ccaa0d8b42060a104 83.76MiB

uploading: file://ruby-25-rhel7 sha256:f56c75ce461b282857e724ab7d07a223537f6ce6a5a4ad32900b6ea14787e6f9 13.46MiB

uploading: file://ruby-25-rhel7 sha256:23302e52b49d49a0a25da8ea870bc1973e7d51c9b306f3539cd397318bd8b0a5 72.73MiB

uploading: file://ruby-25-rhel7 sha256:fd542ee251592bb3ed566aa42ccf98987a5607b64823720d19d68ae382e48bb0 6.876MiB

sha256:eb9e68f88d4c6a43624b965129f010ca5d1e5e2b662c76d9dcd808f738d7940c file://ruby-25-rhel7:latest

info: Mirroring completed in 2.77s (66.79MB/s)

oc image info --dir=${OCP\_PATH}/app-image/redhat-app/images file://ruby-25-rhel7:latest

Name: ruby-25-rhel7:latest

Digest: sha256:eb9e68f88d4c6a43624b965129f010ca5d1e5e2b662c76d9dcd808f738d7940c

Media Type: application/vnd.docker.distribution.manifest.v2+json

Created: 12d ago

Image Size: 185.4MB in 5 layers

Layers: 76.26MB sha256:23302e52b49d49a0a25da8ea870bc1973e7d51c9b306f3539cd397318bd8b0a5

1.62kB sha256:cf5693de4d3cdd6f352978b87c8f89ead294eff44938598f57a91cf7a02417d2

7.21MB sha256:fd542ee251592bb3ed566aa42ccf98987a5607b64823720d19d68ae382e48bb0

87.83MB sha256:63abcb3c00123e524cf21f49a099025395236c9c7e525d9ccaa0d8b42060a104

14.12MB sha256:f56c75ce461b282857e724ab7d07a223537f6ce6a5a4ad32900b6ea14787e6f9

...

tree ${OCP\_PATH}/app-image/redhat-app/images

└── v2  
 └── rhscl

└── ruby-25-rhel7

├── blobs

│ ├── sha256:1ae2a9485dabcc0a8d6515ab31aae576cd353c819ccf1487e3026dfec82f68f4

│ ├── sha256:23302e52b49d49a0a25da8ea870bc1973e7d51c9b306f3539cd397318bd8b0a5

│ ├── sha256:63abcb3c00123e524cf21f49a099025395236c9c7e525d9ccaa0d8b42060a104

│ ├── sha256:cf5693de4d3cdd6f352978b87c8f89ead294eff44938598f57a91cf7a02417d2

│ ├── sha256:eb9e68f88d4c6a43624b965129f010ca5d1e5e2b662c76d9dcd808f738d7940c

│ ├── sha256:f56c75ce461b282857e724ab7d07a223537f6ce6a5a4ad32900b6ea14787e6f9

│ └── sha256:fd542ee251592bb3ed566aa42ccf98987a5607b64823720d19d68ae382e48bb0

└── manifests

├── latest -> /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/ruby-25-rhel7/manifests/sha256:eb9e68f88d4c6a43624b965129f010ca5d1e5e2b662c76d9dcd808f738d7940c

└── sha256:eb9e68f88d4c6a43624b965129f010ca5d1e5e2b662c76d9dcd808f738d7940c

5 directories, 9 files

删除测试文件

rm -rf ${OCP\_PATH}/app-image/redhat-app/images

#### 批量下载镜像

cat ${OCP\_PATH}/app-image/redhat-app/app-images.txt | while read line; do

echo “"================> Begin downloading $line <================"”

oc image mirror -a ${REDHAT\_SECRET} ${line} --filter-by-os=linux/amd64 --dir=${OCP\_PATH}/app-image/redhat-app/images \

file://$(echo ${line} | cut -d '/' -f2)/$(echo ${line} | cut -d '/' -f3)

done

du -lh ${OCP\_PATH}/app-image/redhat-app/images/v2 --max-depth=1

92M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/3scale-amp22

99M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/3scale-amp26

7.7G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/devtools

1008M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/dotnet

6.7G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/fuse7

3.4G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-amq-6

1.9G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-datagrid-6

3.4G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-datagrid-7

9.2G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-datavirt-6

3.2G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-decisionserver-6

5.2G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-eap-6

6.1G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-eap-7

3.2G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-eap-7-tech-preview

652M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-fuse-6

3.2G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-processserver-6

4.0G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-webserver-3

525M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/jboss-webserver-5

351M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/openjdk

1.1G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/redhat-openjdk-18

7.3G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/redhat-sso-7

2.5G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/rhdm-7

1.1G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/rhdm-7-tech-preview

375M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/rhoar-nodejs

188M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/rhoar-nodejs-tech-preview

4.2G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/rhpam-7

4.1G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/rhscl

90M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/3scale-amp21

107M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/3scale-amp23

98M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/3scale-amp24

97M /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/3scale-amp25

81G /data/OCP-4.5.9/ocp/app-image/redhat-app/images/v2/

#### 检查下载镜像

查看已下载镜像，查看是否有下载错误出现

cat ${OCP\_PATH}/app-image/redhat-app/app-images.txt | while read line; do

oc image info --dir=${OCP\_PATH}/app-image/redhat-app/images \

file://$(echo ${line} | cut -d '/' -f2)/$(echo ${line} | cut -d '/' -f3) | grep error

if [ $? -eq 0 ]; then

echo "ERROR for ${line}."

else

echo "RIGHT for ${line}."

fi

done

#### 批量打包镜像

for dir1 in $(ls --indicator-style=none ${OCP\_PATH}/app-image/redhat-app/images/v2); do

for dir2 in $(ls --indicator-style=none ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir1}); do

tar -zcvf ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir1}/${dir2}.tar.gz \

-C ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir1} ${dir2};

done

done

#### 清除下载文件

shopt -s extglob

for dir1 in $(ls --indicator-style=none ${OCP\_PATH}/app-image/redhat-app/images/v2); do

rm -rf ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir1}/!(\*.tar.gz)

done

### 下载CoreOS 4.5镜像

#### 获取CoreOS 4.5最新版本信息

RHCOS\_VER=$(curl -s https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.5/latest/sha256sum.txt | \

grep installer | awk -F\- '{print $2}' | head -1)

echo ${RHCOS\_VER}

4.5.9

#### 查看不同平台的CoreOS文件列表

curl -s https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.5/latest/sha256sum.txt | awk '{print $2}' | grep rhcos

rhcos-4.5.6-x86\_64-aws.x86\_64.vmdk.gz

rhcos-4.5.6-x86\_64-azure.x86\_64.vhd.gz

rhcos-4.5.6-x86\_64-gcp.x86\_64.tar.gz

rhcos-4.5.6-x86\_64-installer-initramfs.x86\_64.img

rhcos-4.5.6-x86\_64-installer-kernel-x86\_64

rhcos-4.5.6-x86\_64-installer.x86\_64.iso

rhcos-4.5.6-x86\_64-metal.x86\_64.raw.gz

rhcos-4.5.6-x86\_64-openstack.x86\_64.qcow2.gz

rhcos-4.5.6-x86\_64-ostree.x86\_64.tar

rhcos-4.5.6-x86\_64-qemu.x86\_64.qcow2.gz

rhcos-4.5.6-x86\_64-vmware.x86\_64.ova

#### 下载CoreOS 4.5镜像文件

wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.5/${RHCOS\_VER}/\

rhcos-${RHCOS\_VER}-x86\_64-installer.x86\_64.iso -P ${OCP\_PATH}/rhcos

wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.5/${RHCOS\_VER}/\

rhcos-${RHCOS\_VER}-x86\_64-metal.x86\_64.raw.gz -P ${OCP\_PATH}/rhcos

# 配置Support节点的服务

**注意**：如无说明，用root用户在Support节点上运行本节的命令。如有说明，请按说明在对应节点操作。

[Support]

[root@support ~]#

## 上传安装介质

### 安装介质文件说明

将包括安装介质的OCP-4.5.9目录传入[Support]节点的/data目录下，包括如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 软件包类型 | 软件包名称 | 大约容量 | 说明 |
| RPM Packages | OCP-4.5.9/yum/rhel-7-server-rpms.tar.gz | 4.6GB | 红帽官方YUM源 |
| OCP-4.5.9/yum/rhel-7-server-extras-rpms.tar.gz | 268MB | 红帽官方YUM源 |
| OCP-4.5.9/yum/rhel-7-server-ose-4.3-rpms.tar.gz | 449MB | 红帽官方YUM源 |
| OpenShift相关文件 | OCP-4.5.9/ocp/ocp-image/ocp-image-4.5.9.tar | 5.8GB | OCP 4.5.9核心镜像 |
| OCP-4.5.9/ocp/app-image/ |  | OCP应用镜像 |
| OCP-4.5.9/ocp/ocp-client | 26MB | OCP客户端 |
| OCP-4.5.9/ocp/ocp-installer | 78MB | OCP安装应用 |
| OCP-4.5.9/ocp/rhcos/\* | 1GB | CoreOS操作系统 |
| OCP-4.5.9/ocp/csi/nfs\* | 44MB | NFS StorageObject |

### 上传安装介质

假设所有安装文件在[Bastion]节点的OCP-4.5.9目录，那么就在[Bastion]节点执行以下操作。

[Bastion]

[root@bastion ~]#

在[Bastion]节点向hosts文件添加临时域名解析。

cat << EOF >> /etc/hosts

192.168.1.12 support.example.internal

EOF

在[Support]节点创建data目录，然后把“OCP-4.5.9”目录传到[Support]节点data目录下。

ssh root@support.example.internal "mkdir /data"

scp -r <YOUR-DIR>/OCP-4.5.9 root@support.example.internal:/data/

## 设置防火墙和SELinux

### 关闭firewalld防火墙

systemctl stop firewalld

systemctl disable firewalld

systemctl mask firewalld

### 设置selinux

sed -i 's/^SELINUX=.\*/SELINUX=permissive/' /etc/selinux/config

reboot

重启后检查状态变为permissive。

getenforce

permissive

## 设置环境变量

本文所有环境变量将使用以下shell中方法设置到./bashrc。

cat << EOF >> ~/.bashrc

#######################################  
setVAR(){

if [ \$# = 0 ]

then

echo "USAGE: "

echo " setVAR VAR\_NAME VAR\_VALUE # Set VAR\_NAME with VAR\_VALUE"

echo " setVAR VAR\_NAME # Delete VAR\_NAME"

elif [ \$# = 1 ]

then

sed -i "/\${1}/d" ~/.bashrc

source ~/.bashrc  
 unset \${1}

echo \${1} is empty

else

sed -i "/\${1}/d" ~/.bashrc

echo export \${1}=\"\${2}\" >> ~/.bashrc

source ~/.bashrc  
 echo \${1}="\${2}"

fi  
 echo ${VAR\_NAME}

}  
#######################################

EOF

source ~/.bashrc

setVAR OCP\_VER 4.5.9

setVAR RHCOS\_VER 4.5.6

setVAR YUM\_PATH /data/OCP-${OCP\_VER}/yum #存放yum源的目录  
setVAR OCP\_PATH /data/OCP-${OCP\_VER}/ocp #存放OCP原始安装介质的目录

## 配置本地临时YUM源

### 准备YUM源所需的文件

先解压缩文件，然后删除压缩文件

for file in $(ls ${YUM\_PATH}/\*.tar.gz); do tar -zxvf ${file} -C ${YUM\_PATH}/; done

rm -rf ${YUM\_PATH}/\*.tar.gz

### 配置本地临时YUM源

创建以下文件，配置本地临时YUM源

cat << EOF > /etc/yum.repos.d/base.repo

[rhel7.6]

name=rhel7.6

baseurl=file://${YUM\_PATH}/rhel-7-server-rpms

enabled=1

gpgcheck=0

EOF

安装createrepo, 然后创建YUM源索引

yum -y install createrepo

for dir in $(ls --indicator-style=none $YUM\_PATH); do createrepo ${YUM\_PATH}/${dir}/; done

### 创建基于HTTP的YUM服务

安装Apache HTTP服务，并将http的端口修改为8080

yum -y install httpd

systemctl enable httpd --now

sed -i -e 's/Listen 80/Listen 8080/g' /etc/httpd/conf/httpd.conf

cat /etc/httpd/conf/httpd.conf |grep "Listen 8080"

Listen 8080

**注意**：必须将yum目录所属首级目录/data以及所有子目录权限设为705。

chmod -R 705 /data

创建指向yum目录的httpd配置文件。

cat << EOF > /etc/httpd/conf.d/yum.conf

Alias /repo "${YUM\_PATH}"

<Directory "${YUM\_PATH}">

Options +Indexes +FollowSymLinks

Require all granted

</Directory>

<Location /repo>

SetHandler None

</Location>

EOF

重新启动 httpd 服务，然后验证可以访问到repo目录

systemctl restart httpd

curl <http://localhost:8080/repo/>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

<html>

<head>

<title>Index of /repo</title>

</head>

<body>

<h1>Index of /repo</h1>

<table>

<tr><th valign="top"><img src="/icons/blank.gif" alt="[ICO]"></th><th><a href="?C=N;O=D">Name</a></th><th><a href="?C=M;O=A">Last modified</a></th><th><a href="?C=S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></th></tr>

<tr><th colspan="5"><hr></th></tr>

<tr><td valign="top"><img src="/icons/back.gif" alt="[PARENTDIR]"></td><td><a href="/">Parent Directory</a> </td><td>&nbsp;</td><td align="right"> - </td><td>&nbsp;</td></tr>

<tr><td valign="top"><img src="/icons/folder.gif" alt="[DIR]"></td><td><a href="rhel-7-server-extras-rpms/">rhel-7-server-extras..&gt;</a></td><td align="right">2020-04-30 02:48 </td><td align="right"> - </td><td>&nbsp;</td></tr>

<tr><td valign="top"><img src="/icons/folder.gif" alt="[DIR]"></td><td><a href="rhel-7-server-ose-4.3-rpms/">rhel-7-server-ose-4...&gt;</a></td><td align="right">2020-04-30 02:48 </td><td align="right"> - </td><td>&nbsp;</td></tr>

<tr><td valign="top"><img src="/icons/folder.gif" alt="[DIR]"></td><td><a href="rhel-7-server-rpms/">rhel-7-server-rpms/</a> </td><td align="right">2020-04-30 05:59 </td><td align="right"> - </td><td>&nbsp;</td></tr>

<tr><th colspan="5"><hr></th></tr>

</table>

</body></html>

## 安装配置DNS服务

OpenShift 4建议的域名组成为：集群名+根域名 $OCP\_CLUSTER\_ID.$DOMAIN

对于etcd，OCP要求由etcd-$INDEX格式组成。本例中由于etcd安装于master上，因此etcd的域名实际也是指向各master节点。此外，etcd还需要\_etcd-server-ssl.\_tcp.$CLUSTERDOMMAIN的SRV记录，用于master寻找etcd节点，该域名指向etcd节点。

### 安装BIND服务

yum -y install bind bind-utils

systemctl enable named --now

### 设置BIND配置文件

先备份原始BIND配置文件，然后修改BIND配置，并重新加载配置

cp /etc/named.conf{,\_bak}

sed -i -e "s/listen-on port.\*/listen-on port 53 { any; };/" /etc/named.conf

sed -i -e "s/allow-query.\*/allow-query { any; };/" /etc/named.conf

rndc reload

grep -E 'listen-on port|allow-query' /etc/named.conf

listen-on port 53 { any; };  
 allow-query { any; };

|  |  |  |  |
| --- | --- | --- | --- |
|  | 💡 NOTE | |  |
|  |  | 如果有外网的解析需求，则请确保DNS服务器可以访问外网，并添加如下配置： |  |
| sed -i '/recursion yes;/a \  forward first; \  forwarders { 114.114.114.114; 8.8.8.8; };' /etc/named.conf |
| sed -i -e "s/dnssec-enable.\*/dnssec-enable no;/" /etc/named.conf |
| sed -i -e "s/dnssec-validation.\*/dnssec-validation no;/" /etc/named.conf |
| rndc reload |

### 配置Zone区域

#### 设置DNS环境变量

setVAR DOMAIN example.internal

setVAR OCP\_CLUSTER\_ID ocp4-1

setVAR BASTION\_IP 192.168.1.13

setVAR SUPPORT\_IP 192.168.1.12

setVAR DNS\_IP 192.168.1.12

setVAR NTP\_IP 192.168.1.12

setVAR YUM\_IP 192.168.1.12

setVAR REGISTRY\_IP 192.168.1.12

setVAR NFS\_IP 192.168.1.12

setVAR LB\_IP 192.168.1.12

setVAR BOOTSTRAP\_IP 192.168.1.100

setVAR MASTER0\_IP 192.168.1.101

setVAR WORKER0\_IP 192.168.1.110

setVAR WORKER1\_IP 192.168.1.111

#### 添加解析Zone区域

执行以下命令添加3个解析ZONE（如果要执行多次，需要手动删除以前增加的内容），它们分别为：

|  |  |
| --- | --- |
| 域名后缀 | 解释 |
| example.internal | 集群内部域名后缀：集群内部所有节点的主机名均采用该域名后缀 |
| ocp4-1.example.internal | OCP集群的域名，如本例中的集群名为ocp4-1，则域名为ocp4-1.example.internal |
| 168.192.in-addr.arpa | 用于集群内所有节点的反向解析 |

cat >> /etc/named.rfc1912.zones << EOF

zone "${DOMAIN}" IN {

type master;

file "${DOMAIN}.zone";

allow-transfer { any; };

};

zone "${OCP\_CLUSTER\_ID}.${DOMAIN}" IN {

type master;

file "${OCP\_CLUSTER\_ID}.${DOMAIN}.zone";

allow-transfer { any; };

};

zone "168.192.in-addr.arpa" IN {

type master;

file "168.192.in-addr.arpa.zone";

allow-transfer { any; };

};

EOF

#### 创建example.internal.zone区域配置文件

cat > /var/named/${DOMAIN}.zone << EOF

\$ORIGIN ${DOMAIN}.

\$TTL 1D

@ IN SOA ${DOMAIN}. admin.${DOMAIN}. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ IN NS dns.${DOMAIN}.

bastion IN A ${BASTION\_IP}

support IN A ${SUPPORT\_IP}

dns IN A ${DNS\_IP}

ntp IN A ${NTP\_IP}

yum IN A ${YUM\_IP}

registry IN A ${REGISTRY\_IP}

nfs IN A ${NFS\_IP}

EOF

cat /var/named/${DOMAIN}.zone

$ORIGIN example.internal.

$TTL 1D

@ IN SOA example.internal. admin.example.internal. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ IN NS dns.example.internal.

bastion IN A 192.168.1.13

support IN A 192.168.1.12

dns IN A 192.168.1.12

ntp IN A 192.168.1.12

yum IN A 192.168.1.12

registry IN A 192.168.1.12

nfs IN A 192.168.1.12

#### 创建ocp4-1.example.internal.zone区域配置文件

cat > /var/named/${OCP\_CLUSTER\_ID}.${DOMAIN}.zone << EOF

\$ORIGIN ${OCP\_CLUSTER\_ID}.${DOMAIN}.

\$TTL 1D

@ IN SOA ${OCP\_CLUSTER\_ID}.${DOMAIN}. admin.${OCP\_CLUSTER\_ID}.${DOMAIN}. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ IN NS dns.${DOMAIN}.

dns.${DOMAIN} IN A ${DNS\_IP}

lb IN A ${LB\_IP}

api IN A ${LB\_IP}

api-int IN A ${LB\_IP}

\*.apps IN A ${LB\_IP}

bootstrap IN A ${BOOTSTRAP\_IP}

master-0 IN A ${MASTER0\_IP}

etcd-0 IN A ${MASTER0\_IP}

worker-0 IN A ${WORKER0\_IP}

worker-1 IN A ${WORKER1\_IP}

\_etcd-server-ssl.\_tcp.${OCP\_CLUSTER\_ID}.${DOMAIN}. 8640 IN SRV 0 10 2380 etcd-0.${OCP\_CLUSTER\_ID}.${DOMAIN}.

EOF

cat /var/named/${OCP\_CLUSTER\_ID}.${DOMAIN}.zone

$ORIGIN ocp4-1.example.internal.

$TTL 1D

@ IN SOA ocp4-1.example.internal. admin.ocp4-1.example.internal. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ IN NS dns.example.internal.

dns.example.internal IN A 192.168.1.12

lb IN A 192.168.1.12

api IN A 192.168.1.12

api-int IN A 192.168.1.12

\*.apps IN A 192.168.1.12

bootstrap IN A 192.168.1.100

master-0 IN A 192.168.1.101

etcd-0 IN A 192.168.1.101

worker-0 IN A 192.168.1.110

worker-1 IN A 192.168.1.111

\_etcd-server-ssl.\_tcp.ocp4-1.example.internal. 8640 IN SRV 0 10 2380 etcd-0.ocp4-1.example.internal.

#### 创建168.192.in-addr.arpa.zone反向解析区域配置文件

**注意**：以下脚本中的反向IP如果有变化需要在此手动修改。

cat > /var/named/168.192.in-addr.arpa.zone << EOF

\$TTL 1D

@ IN SOA ${DOMAIN}. admin.${DOMAIN}. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ IN NS dns.${DOMAIN}.

13.1.168.192.in-addr.arpa. IN PTR bastion.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR support.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR dns.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR ntp.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR yum.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR registry.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR nfs.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR lb.${OCP\_CLUSTER\_ID}.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR api.${OCP\_CLUSTER\_ID}.${DOMAIN}.

12.1.168.192.in-addr.arpa. IN PTR api-int.${OCP\_CLUSTER\_ID}.${DOMAIN}.

100.1.168.192.in-addr.arpa. IN PTR bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN}.

101.1.168.192.in-addr.arpa. IN PTR master-0.${OCP\_CLUSTER\_ID}.${DOMAIN}.

110.1.168.192.in-addr.arpa. IN PTR worker-0.${OCP\_CLUSTER\_ID}.${DOMAIN}.

111.1.168.192.in-addr.arpa. IN PTR worker-1.${OCP\_CLUSTER\_ID}.${DOMAIN}.

EOF

cat /var/named/168.192.in-addr.arpa.zone

$TTL 1D

@ IN SOA example.internal. admin.example.internal. (

0 ; serial

1D ; refresh

1H ; retry

1W ; expire

3H ) ; minimum

@ IN NS dns.example.internal.

13.1.168.192.in-addr.arpa. IN PTR bastion.example.internal.

12.1.168.192.in-addr.arpa. IN PTR support.example.internal.

12.1.168.192.in-addr.arpa. IN PTR dns.example.internal.

12.1.168.192.in-addr.arpa. IN PTR ntp.example.internal.

12.1.168.192.in-addr.arpa. IN PTR yum.example.internal.

12.1.168.192.in-addr.arpa. IN PTR registry.example.internal.

12.1.168.192.in-addr.arpa. IN PTR nfs.example.internal.

12.1.168.192.in-addr.arpa. IN PTR lb.ocp4-1.example.internal.

12.1.168.192.in-addr.arpa. IN PTR api.ocp4-1.example.internal.

12.1.168.192.in-addr.arpa. IN PTR api-int.ocp4-1.example.internal.

100.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4-1.example.internal.

101.1.168.192.in-addr.arpa. IN PTR master-0.ocp4-1.example.internal.

110.1.168.192.in-addr.arpa. IN PTR worker-0.ocp4-1.example.internal.

111.1.168.192.in-addr.arpa. IN PTR worker-1.ocp4-1.example.internal.

### 重启BIND服务

重启BIND服务，然后检查没有错误日志。

systemctl restart named

rndc reload

journalctl -u named

### 测试正反向DNS解析

1. 正向解析测试

dig nfs.${DOMAIN} +short

192.168.1.12

dig yum.${DOMAIN} +short

192.168.1.12

dig registry.${DOMAIN} +short

192.168.1.12

dig ntp.${DOMAIN} +short

192.168.1.12

dig lb.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.12

dig api.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.12

dig api-int.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.12

dig bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.100

dig master-0.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.101

dig etcd-0.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.101

dig worker-0.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.110

dig worker-1.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.111

dig \_etcd-server-ssl.\_tcp.${OCP\_CLUSTER\_ID}.${DOMAIN} SRV +short

0 10 2380 etcd-0.ocp4-1.example.internal.

dig \*.apps.${OCP\_CLUSTER\_ID}.${DOMAIN} +short

192.168.1.12

1. 反向解析测试

dig -x 192.168.1.13 +short

bastion.example.internal.

dig -x 192.168.1.12 +short

lb.ocp4-1.example.internal.

nfs.example.internal.

api.ocp4-1.example.internal.

api-int.ocp4-1.example.internal.

ntp.example.internal.

support.example.internal.

yum.example.internal.

registry.example.internal.

dns.example.internal.

dig -x 192.168.1.100 +short

bootstrap.ocp4-1.example.internal.

dig -x 192.168.1.101 +short

master-0.ocp4-1.example.internal.

dig -x 192.168.1.110 +short

worker-0.ocp4-1.example.internal.

dig -x 192.168.1.111 +short

worker-1.ocp4-1.example.internal.

## 配置正式YUM源

### 配置Support节点的YUM源

setVAR YUM\_DOMAIN yum.${DOMAIN}:8080

1. 创建yum repo配置文件

cat > /etc/yum.repos.d/ocp.repo << EOF

[rhel-7-server]

name=rhel-7-server

baseurl=http://${YUM\_DOMAIN}/repo/rhel-7-server-rpms/

enabled=1

gpgcheck=0

[rhel-7-server-extras]

name=rhel-7-server-extras

baseurl=http://${YUM\_DOMAIN}/repo/rhel-7-server-extras-rpms/

enabled=1

gpgcheck=0

[rhel-7-server-ose-4.3-rpms]

name=rhel-7-server-ose-4.3-rpms

baseurl=http://${YUM\_DOMAIN}/repo/rhel-7-server-ose-4.3-rpms/

enabled=1

gpgcheck=0

EOF

1. 删除临时yum源

mv /etc/yum.repos.d/base.repo{,.bak}

1. 更新正式yum源的缓存

yum clean all

yum makecache

yum repolist

repo id repo name status

rhel-7-server rhel-7-server 5,469

rhel-7-server-extras rhel-7-server-extras 152

rhel-7-server-ose-4.3-rpms rhel-7-server-ose-4.3-rpms 90

repolist: 5,711

### 配置Bastion节点的YUM源

[Bastion]

[root@bastion ~]#

将Support节点的yum repo文件复制到Bastion本地

scp root@support.example.internal:/etc/yum.repos.d/ocp.repo /etc/yum.repos.d/ocp.repo

更新Bastion节点YUM源，然后查看包括的repo列表。

yum clean all

yum makecache

yum repolist

repo id repo name status

rhel-7-server rhel-7-server 5,469

rhel-7-server-extras rhel-7-server-extras 152

rhel-7-server-ose-4.3-rpms rhel-7-server-ose-4.3-rpms 90

repolist: 5,711

### 安装基础软件包，验证YUM源

分别在Support和Bastion节点安装一下软件包，验证2个几点的YUM源是正常的。

yum -y install wget git net-tools bridge-utils bash-completion kexec-tools tmux screen jq tree httpd-tools

## 再次验证DNS服务

[Bastion]

[root@bastion ~]#

1. 正向解析测试

dig nfs.example.internal +short

192.168.1.12

dig yum.example.internal +short

192.168.1.12

dig registry.${DOMAIN} +short

192.168.1.12

dig ntp.example.internal +short

192.168.1.12

dig lb.example.internal +short

192.168.1.12

dig api.example.internal +short

192.168.1.12

dig api-int.example.internal +short

192.168.1.12

dig bootstrap.example.internal +short

192.168.1.100

dig master-0.example.internal +short

192.168.1.101

dig etcd-0.example.internal +short

192.168.1.101

dig worker-0.example.internal +short

192.168.1.110

dig worker-1.example.internal +short

192.168.1.111

dig \_etcd-server-ssl.\_tcp.ocp4-1.example.internal SRV +short

0 10 2380 etcd-0.ocp4-1.example.internal.

dig api.ocp4-1.example.internal +short

192.168.1.12

dig \*.apps.ocp4-1.example.internal +short

192.168.1.12

1. 反向解析测试

dig -x 192.168.1.13 +short

bastion.example.internal.

dig -x 192.168.1.12 +short

lb.ocp4-1.example.internal.

nfs.example.internal.

api.ocp4-1.example.internal.

api-int.ocp4-1.example.internal.

ntp.example.internal.

support.example.internal.

yum.example.internal.

registry.example.internal.

dns.example.internal.

dig -x 192.168.1.100 +short

bootstrap.ocp4-1.example.internal.

dig -x 192.168.1.101 +short

master-0.ocp4-1.example.internal.

dig -x 192.168.1.110 +short

worker-0.ocp4-1.example.internal.

dig -x 192.168.1.111 +short

worker-1.ocp4-1.example.internal.

## 部署NTP服务

### 设置正确的时区

timedatectl list-timezones |grep Asia/Shanghai

timedatectl set-timezone Asia/Shanghai

### 安装chrony

yum -y install chrony

systemctl enable chronyd --now

systemctl status chronyd

chronyd.service - NTP client/server

Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)

Active: active (running) since Mon 2015-07-06 16:29:54 EDT; 15s ago

Main PID: 2530 (chronyd)

CGroup: /system.slice/chronyd.service

ââ2530 /usr/sbin/chronyd -u chrony

### 配置chrony

1. 先备份原始chrony.conf配置文件，在修改配置文件

cp /etc/chrony.conf{,.bak}

sed -i -e "s/^server\*/#&/g" \

-e "s/#local stratum 10/local stratum 10/g" \

-e "s/#allow 192.168.0.0\/16/allow all/g" \

/etc/chrony.conf

cat >> /etc/chrony.conf << EOF

server ntp.${DOMAIN} iburst

EOF

cat /etc/chrony.conf

# Use public servers from the pool.ntp.org project.

# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#server 0.rhel.pool.ntp.org iburst

#server 1.rhel.pool.ntp.org iburst

#server 2.rhel.pool.ntp.org iburst

#server 3.rhel.pool.ntp.org iburst

# Record the rate at which the system clock gains/losses time.

driftfile /var/lib/chrony/drift

# Allow the system clock to be stepped in the first three updates

# if its offset is larger than 1 second.

makestep 1.0 3

# Enable kernel synchronization of the real-time clock (RTC).

rtcsync

# Enable hardware timestamping on all interfaces that support it.

#hwtimestamp \*

# Increase the minimum number of selectable sources required to adjust

# the system clock.

#minsources 2

# Allow NTP client access from local network.

allow all

# Serve time even if not synchronized to a time source.

local stratum 10

# Specify file containing keys for NTP authentication.

#keyfile /etc/chrony.keys

# Specify directory for log files.

logdir /var/log/chrony

# Select which information is logged.

#log measurements statistics tracking

server ntp.example.internal iburst

1. 重启chrony服务

systemctl restart chronyd

### 检查chrony服务端启动

ps -auxw |grep chrony

chrony 13882 0.0 0.0 22524 1160 ? S 14:57 0:00 /usr/sbin/chronyd

root 13911 0.0 0.0 112708 976 pts/0 R+ 15:12 0:00 grep --color=auto chrony

ss -lnup |grep chronyd

UNCONN 0 0 \*:123 \*:\* users:(("chronyd",pid=14217,fd=3))

UNCONN 0 0 127.0.0.1:323 \*:\* users:(("chronyd",pid=14217,fd=1))

UNCONN 0 0 ::1:323 :::\* users:(("chronyd",pid=14217,fd=2))

chronyc tracking

Reference ID : C0A81E0B (ntp.example.internal)

Stratum : 12

Ref time (UTC) : Fri Mar 20 13:46:44 2020

System time : 0.000002813 seconds fast of NTP time

Last offset : -0.002774593 seconds

RMS offset : 0.003830482 seconds

Frequency : 32.672 ppm fast

Residual freq : -24.591 ppm

Skew : 41.876 ppm

Root delay : 0.000733790 seconds

Root dispersion : 0.052910943 seconds

Update interval : 64.9 seconds

Leap status : Normal

### 验证chrony服务同步

[Bastion]

[root@bastion ~]#

在验证环境安装chrony

yum -y install chrony

systemctl enable chronyd --now

systemctl status chronyd

● chronyd.service - NTP client/server

Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)

Active: active (running) since Tue 2020-05-05 16:31:24 CST; 59min ago

Docs: man:chronyd(8)

man:chrony.conf(5)

CGroup: /system.slice/chronyd.service

└─5951 /usr/sbin/chronyd

配置chrony，然后重启chrony服务

cat >> /etc/chrony.conf << EOF

server ntp.example.internal iburst

EOF

systemctl restart chronyd

执行手工同步，查看时间同步源

chronyc -a makestep

chronyc sources -v

210 Number of sources = 1

.-- Source mode '^' = server, '=' = peer, '#' = local clock.

/ .- Source state '\*' = current synced, '+' = combined , '-' = not combined,

| / '?' = unreachable, 'x' = time may be in error, '~' = time too variable.

|| .- xxxx [ yyyy ] +/- zzzz

|| Reachability register (octal) -. | xxxx = adjusted offset,

|| Log2(Polling interval) --. | | yyyy = measured offset,

|| \ | | zzzz = estimated error.

|| | | \

MS Name/IP address Stratum Poll Reach LastRx Last sample

===============================================================================

^\* ntp.example.internal 11 7 377 76 +8268ns[ +12us] +/- 2253us

## 部署本地Docker Registry

该Docker Registry镜像库用于提供OCP安装过程所需的容器镜像。

### 创建Docker Registry相关目录

setVAR REGISTRY\_PATH /data/registry ## 容器镜像库存放的根目录

mkdir -p ${REGISTRY\_PATH}/{auth,certs,data}

### 创建访问Docker Registry的证书

openssl req -newkey rsa:4096 -nodes -sha256 -keyout ${REGISTRY\_PATH}/certs/registry.key -x509 -days 365 \

-out ${REGISTRY\_PATH}/certs/registry.crt \

-subj "/C=CN/ST=BEIJING/L=BJ/O=REDHAT/OU=IT/CN=registry.${DOMAIN}/emailAddress=admin@${DOMAIN}"

Generating a 4096 bit RSA private key

...................++

.......................................++

writing new private key to 'registry.key'

-----

openssl x509 -in ${REGISTRY\_PATH}/certs/registry.crt -text | head -n 14

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

98:a6:8e:6a:6a:54:bd:5a

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=CN, ST=BEIJING, L=BJ, O=REDHAT, OU=IT, CN=registry.example.internal/emailAddress=admin@example.internal

Validity

Not Before: May 5 09:19:47 2020 GMT

Not After : May 5 09:19:47 2021 GMT

Subject: C=CN, ST=BEIJING, L=BJ, O=REDHAT, OU=IT, CN=registry.example.internal/emailAddress=admin@example.internal

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (4096 bit)

### 安装Docker Registry

1. 安装docker-distribution

yum -y install docker-distribution

1. 创建Registry认证凭据，允许用openshift/redhat登录。

htpasswd -bBc ${REGISTRY\_PATH}/auth/htpasswd openshift redhat

cat ${REGISTRY\_PATH}/auth/htpasswd

openshift:$2y$05$W1SgxVO/h/6Pbp32NDk4mOkj0YxSUCl1/l5IzA6oyWbm4P/kkmrSu

1. 创建docker-distribution配置文件

setVAR REG\_DOMAIN registry.${DOMAIN}:5000 ## 容器镜像库的访问域名

cat << EOF > /etc/docker-distribution/registry/config.yml

version: 0.1

log:

fields:

service: registry

storage:

cache:

layerinfo: inmemory

filesystem:

rootdirectory: ${REGISTRY\_PATH}/data

delete:

enabled: false

auth:

htpasswd:

realm: basic-realm

path: ${REGISTRY\_PATH}/auth/htpasswd

http:

addr: 0.0.0.0:5000

host: https://${REG\_DOMAIN}

tls:

certificate: ${REGISTRY\_PATH}/certs/registry.crt

key: ${REGISTRY\_PATH}/certs/registry.key

EOF

cat /etc/docker-distribution/registry/config.yml

version: 0.1

log:

fields:

service: registry

storage:

cache:

layerinfo: inmemory

filesystem:

rootdirectory: /data/registry/data

delete:

enabled: true

auth:

htpasswd:

realm: basic-realm

path: /data/registry/auth/htpasswd

http:

addr: 0.0.0.0:5000

host: https://registry.example.internal:5000

tls:

certificate: /data/registry/certs/registry.crt

key: /data/registry/certs/registry.key

1. 启动镜像库服务

systemctl enable docker-distribution --now

systemctl status docker-distribution

● docker-distribution.service - v2 Registry server for Docker

Loaded: loaded (/usr/lib/systemd/system/docker-distribution.service; enabled; vendor preset: disabled)

Active: active (running) since Sat 2019-08-10 00:01:51 CST; 15ms ago

Main PID: 19837 (registry)

CGroup: /system.slice/docker-distribution.service

└─19837 /usr/bin/registry serve /etc/docker-distribution/registry/config.yml

Aug 10 00:01:51 support.example.internal systemd[1]: Started v2 Registry server for Docker.

Aug 10 00:01:51 support.example.internal support[10238]: time="2020-07-07T01:08:48-04:00" level=warning msg="No HTTP secret provided - generated random secret. This may cause...

Aug 10 00:01:51 support.example.internal support[10238]: time="2020-07-07T01:08:48-04:00" level=info msg="Starting upload purge in 7m0s" go.version=go1.9.2 instance....+unknown"

Aug 10 00:01:51 support.example.internal support[10238]: time="2020-07-07T01:08:48-04:00" level=info msg="redis not configured" go.version=go1.9.2 instance.id=d6f7e2...+unknown"

Aug 10 00:01:51 support.example.internal support[10238]: time="2020-07-07T01:08:48-04:00" level=info msg="using inmemory blob descriptor cache" go.version=go1.9.2 in...+unknown"

Aug 10 00:01:51 support.example.internal support[10238]: time="2020-07-07T01:08:48-04:00" level=info msg="listening on [::]:5000, tls" go.version=go1.9.2 instance.id...+unknown"

Hint: Some lines were ellipsized, use -l to show in full.

### 从本地访问Docker Registry

cp ${REGISTRY\_PATH}/certs/registry.crt /etc/pki/ca-trust/source/anchors/

update-ca-trust

curl -u openshift:redhat https://${REG\_DOMAIN}/v2/\_catalog

{"repositories":[]}

### 从远程访问Docker Registry

[Bastion]

[root@bastion ~]#

scp root@support.example.internal:/etc/pki/ca-trust/source/anchors/registry.crt /etc/pki/ca-trust/source/anchors/

update-ca-trust

curl -u openshift:redhat https://registry.example.internal:5000/v2/\_catalog

{"repositories":[]}

## 部署HAProxy负载均衡服务

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Port | Machines | Internal | External | Description |
| 6443 | 用于bootstrap和master节点  在master节点初始化完毕后，需要从LB上移除bootstrap记录 | √ | √ | Kubernetes API server |
| 22623 | 用于bootstrap和master节点  在master节点初始化完毕后，需要从LB上移除bootstrap记录 | √ |  | Machine Config server |
| 443 | 指向运行Ingress router pods的worker node | √ | √ | HTTPS traffic |
| 80 | 指向运行Ingress router pods的worker node | √ | √ | HTTP traffic |

1. 安装Haproxy

yum -y install haproxy

systemctl enable haproxy --now

1. 添加haproxy.cfg配置文件

cat <<EOF > /etc/haproxy/haproxy.cfg

# Global settings

#---------------------------------------------------------------------

global

maxconn 20000

log /dev/log local0 info

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid

user haproxy

group haproxy

daemon

# turn on stats unix socket

stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------

# common defaults that all the 'listen' and 'backend' sections will

# use if not designated in their block

#---------------------------------------------------------------------

defaults

mode http

log global

option httplog

option dontlognull

# option http-server-close

option forwardfor except 127.0.0.0/8

option redispatch

retries 3

timeout http-request 10s

timeout queue 1m

timeout connect 10s

timeout client 300s

timeout server 300s

timeout http-keep-alive 10s

timeout check 10s

maxconn 20000

listen stats

bind :9000

mode http

stats enable

stats uri /

frontend openshift-api-server

bind \*:6443

mode tcp

option tcplog

default\_backend openshift-api-server

frontend machine-config-server

bind \*:22623

mode tcp

option tcplog

default\_backend machine-config-server

frontend ingress-http

bind \*:80

mode tcp

option tcplog

default\_backend ingress-http

frontend ingress-https

bind \*:443

mode tcp

option tcplog

default\_backend ingress-https

backend openshift-api-server

balance source

mode tcp

server bootstrap bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN}:6443 check

server master-0 master-0.${OCP\_CLUSTER\_ID}.${DOMAIN}:6443 check

backend machine-config-server

balance source

mode tcp

server bootstrap bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN}:22623 check

server master-0 master-0.${OCP\_CLUSTER\_ID}.${DOMAIN}:22623 check

backend ingress-http

balance source

mode tcp

server worker-0 worker-0.${OCP\_CLUSTER\_ID}.${DOMAIN}:80 check

server worker-1 worker-1.${OCP\_CLUSTER\_ID}.${DOMAIN}:80 check

backend ingress-https

balance source

mode tcp

server worker-0 worker-0.${OCP\_CLUSTER\_ID}.${DOMAIN}:443 check

server worker-1 worker-1.${OCP\_CLUSTER\_ID}.${DOMAIN}:443 check

EOF

cat /etc/haproxy/haproxy.cfg

...

backend openshift-api-server

balance source

mode tcp

server bootstrap bootstrap.ocp4-1.example.internal:6443 check

server master-0 master-0.ocp4-1.example.internal:6443 check

backend machine-config-server

balance source

mode tcp

server bootstrap bootstrap.ocp4-1.example.internal:22623 check

server master-0 master-0.ocp4-1.example.internal:22623 check

backend ingress-http

balance source

mode tcp

server worker-0 worker-0.ocp4-1.example.internal:80 check

server worker-1 worker-1.ocp4-1.example.internal:80 check

backend ingress-https

balance source

mode tcp

server worker-0 worker-0.ocp4-1.example.internal:443 check

server worker-1 worker-1.ocp4-1.example.internal:443 check

1. 重启HAProxy服务

systemctl restart haproxy

1. 检查HAProxy服务端口

ss -lntp |grep haproxy

LISTEN 0 128 \*:9000 \*:\* users:(("haproxy",pid=5968,fd=5))

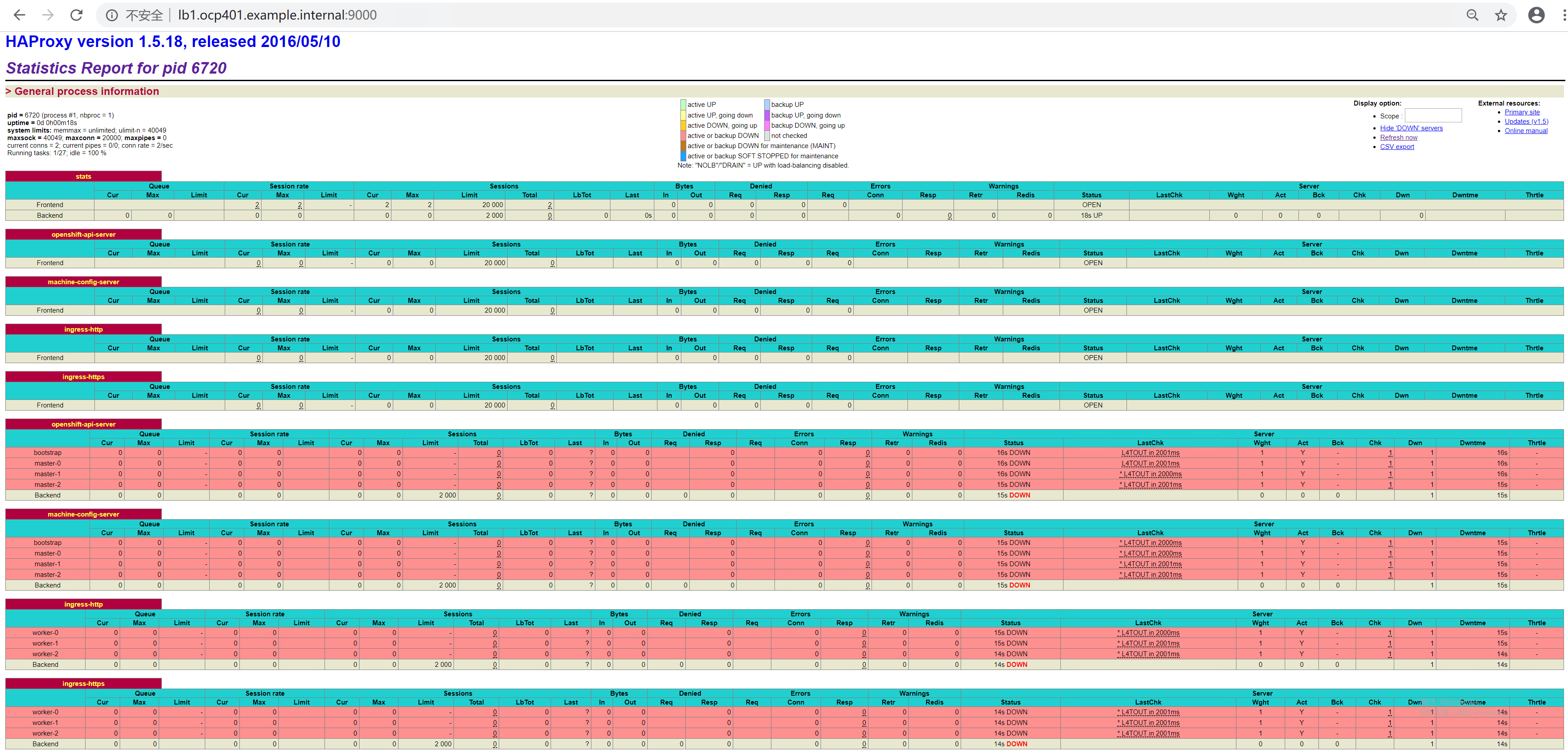
LISTEN 0 128 \*:6443 \*:\* users:(("haproxy",pid=5968,fd=7))

LISTEN 0 128 \*:80 \*:\* users:(("haproxy",pid=5968,fd=9))

LISTEN 0 128 \*:443 \*:\* users:(("haproxy",pid=5968,fd=10))

LISTEN 0 128 \*:22623 \*:\* users:(("haproxy",pid=5968,fd=8))

访问如下页面http://lb.ocp4-1.example.internal:9000/，确认每行颜色和下图一致。



# 定制化安装文件

**注意**：如无说明，本节的命令在Support节点上运行。如有说明，请按说明在对应节点操作。

[Support]

[root@support~]#

## 设置基础环境变量

## REG\_SECRET：指定一个文件，用来保存podman登录本地docker registry时所生成的secret，以便以后可免密登录

setVAR REG\_SECRET ${OCP\_PATH}/secret/registry-secret.json

setVAR REPO\_NAME ocp4/openshift4 ## 在Docker Registry中存放OpenShift核心镜像的Repository

echo ${OCP\_VER},${RHCOS\_VER},${REG\_SECRET},${REG\_DOMAIN},${REPO\_NAME}

4.5.9,4.5.9,/data/OCP-4.5.9/ocp/secret/registry-secret.json,registry.example.internal:5000,ocp4/openshift4

## 准备OpenShift核心镜像

将OpenShift核心镜像导入到本地的Docker Registry中，以便在安装过程中使用。

### 安装oc客户端

tar -xzf ${OCP\_PATH}/ocp-client/openshift-client-linux-${OCP\_VER}.tar.gz -C /usr/local/sbin/

oc version

Client Version: 4.5.9

### 安装镜像操作工具并登陆Docker Registry

yum -y install podman skopeo

用安装Docker Registry时指定的openshift/redhat登录，并生成由$REG\_SECRET指定的免密访问文件。

podman login -u openshift -p redhat --authfile ${REG\_SECRET} ${REG\_DOMAIN}

cat ${REG\_SECRET}

{

"auths": {

"registry.example.internal:5000": {

"auth": "b3BlbnNoaWZ0OnJlZGhhdA=="

}

}

}

### 向Docker Registry导入OpenShift核心镜像

tar -xvf ${OCP\_PATH}/ocp-image/ocp-image-${OCP\_VER}.tar -C ${OCP\_PATH}/ocp-image/

rm -f ${OCP\_PATH}/ocp-image/ocp-image-${OCP\_VER}.tar

oc image mirror -a ${REG\_SECRET} \

--dir=${OCP\_PATH}/ocp-image/mirror\_${OCP\_VER} file://openshift/release:${OCP\_VER}\* ${REG\_DOMAIN}/${REPO\_NAME}

查看已经导入镜像库镜像数量，然后查看镜像信息。

curl -u openshift:redhat https://${REG\_DOMAIN}/v2/\_catalog

{"repositories":["ocp4/openshift4"]}

curl -u openshift:redhat -s https://${REG\_DOMAIN}/v2/${REPO\_NAME}/tags/list |jq -M '.["tags"][]' | wc -l

110

curl -u openshift:redhat -s https://${REG\_DOMAIN}/v2/${REPO\_NAME}/tags/list |jq -M '.["name"] + ":" + .["tags"][]'

oc adm release info -a ${REG\_SECRET} "${REG\_DOMAIN}/${REPO\_NAME}:${OCP\_VER}" | grep -A 200 -i "Images"

Images:

NAME DIGEST

aws-machine-controllers sha256:23cbad0f3cdeae142a35376a447b74b93a60016bec012d6abcf95cf55cd7878e

azure-machine-controllers sha256:6d094bcc6584e136dfd488467c9dd02cc92244b6f9e7d121a628425ee7a6e9a5

baremetal-installer sha256:859ea095b9953369be5a0162309cbde3daafccb59c76facc0a8a7118a928b7fe

baremetal-machine-controllers sha256:6f2a7ee4191e915dc2a865fd1fdccfbfd42c86402976cab72bcd5350804cc338  
...

## 准备CoreOS启动文件及其HTTP访问服务

### 准备CoreOS启动文件的HTTP访问服务

setVAR BOOT\_FILE\_PATH /data/boot-files ## 用来存放所有启动CoreOS所需文件的目录

setVAR RHCOS\_ISO\_PATH ${BOOT\_FILE\_PATH}/rhcos-iso ## 用来存启动CoreOS所需ISO和RAW文件的目录

cat << EOF > /etc/httpd/conf.d/rhcos.conf

Alias /rhcos-iso "${RHCOS\_ISO\_PATH}"

<Directory "${RHCOS\_ISO\_PATH}">

Options +Indexes +FollowSymLinks

Require all granted

</Directory>

<Location /rhcos >

SetHandler None

</Location>

EOF

### 准备CoreOS Metal Raw文件

mkdir -p ${RHCOS\_ISO\_PATH}

cp ${OCP\_PATH}/rhcos/rhcos-${RHCOS\_VER}-x86\_64-metal.x86\_64.raw.gz ${RHCOS\_ISO\_PATH}/

### 准备CoreOS ISO文件

#### 挂载模板ISO

mkdir -p ${BOOT\_FILE\_PATH}/{rhcos-temp,rhcos-mnt}

mount -o loop -t iso9660 ${OCP\_PATH}/rhcos/rhcos-${RHCOS\_VER}-x86\_64-installer.x86\_64.iso ${BOOT\_FILE\_PATH}/rhcos-mnt

tree ${BOOT\_FILE\_PATH}/rhcos-mnt

rhcos-file/

├── EFI

│   └── redhat

│   └── grub.cfg

├── images

│   ├── efiboot.img

│   ├── initramfs.img

│   └── vmlinuz

├── isolinux

│   ├── boot.cat

│   ├── boot.msg

│   ├── isolinux.bin

│   ├── isolinux.cfg

│   ├── ldlinux.c32

│   ├── libcom32.c32

│   ├── libutil.c32

│   └── vesamenu.c32

├── README.md

└── zipl.prm

#### 创建定制ISO引导文件

setVAR NET\_IF\_NAME enp0s3 ## CoreOS启动时缺省创建的网卡名。不同IaaS软件使用的缺省名称不一样，KVM中是ens3, VMWare是ens192。

setVAR GATEWAY 192.168.1.1 ## CoreOS启动时使用的GATEWAY

setVAR NETMASK 255.255.255.0 ## CoreOS启动时使用的NETMASK  
setVAR RHCOS\_METAL\_URL http://${YUM\_DOMAIN}/rhcos-iso/rhcos-${RHCOS\_VER}-x86\_64-metal.x86\_64.raw.gz

mkdir -p ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}

modify\_cfg(){

for file in "${BOOT\_FILE\_PATH}/rhcos-mnt/EFI/redhat/grub.cfg" "${BOOT\_FILE\_PATH}/rhcos-mnt/isolinux/isolinux.cfg"; do

IGN\_FILE\_URL=http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/${1}.ign

# Append the proper image and ignition urls

sed -e '/coreos.inst=yes/s|$| coreos.inst.install\_dev=sda coreos.inst.image\_url='"${RHCOS\_METAL\_URL}"' coreos.inst.ignition\_url='"${IGN\_FILE\_URL}"' ip='"${IP}"'::'"${GATEWAY}"':'"${NETMASK}"':'"${FQDN}"':'"${NET\_IF\_NAME}"':none nameserver='"${DNS\_IP}"'|' ${file} > ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/${NODE\_NAME}\_${file##\*/}

# Boot directly in the installation

sed -i -e 's/default vesamenu.c32/default linux/g' -e 's/timeout 600/timeout 15/g' ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/${NODE\_NAME}\_${file##\*/}

done

}

#创建BOOTSTRAP启动定制文件

NODE\_NAME="bootstrap"

NODE\_TYPE="bootstrap"

IP=${BOOTSTRAP\_IP}

FQDN="${NODE\_NAME}.${OCP\_CLUSTER\_ID}.${DOMAIN}"

modify\_cfg ${NODE\_TYPE}

cat ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/${NODE\_NAME}\_isolinux.cfg |grep append

append initrd=/images/initramfs.img nomodeset rd.neednet=1 coreos.inst=yes coreos.inst.install\_dev=sda coreos.inst.image\_url=http://yum.example.internal:8080/rhcos-iso/rhcos-4.5.6-x86\_64-metal.x86\_64.raw.gz coreos.inst.ignition\_url=http://yum.example.internal:8080/ignition/ocp4-1/bootstrap.ign ip=192.168.1.100::192.168.1.1:255.255.255.0:bootstrap.ocp4-1.example.internal:ens3:none nameserver=192.168.1.12

cat ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/${NODE\_NAME}\_grub.cfg |grep linux

linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes coreos.inst.install\_dev=sda coreos.inst.image\_url=http://yum.example.internal:8080/rhcos-iso/rhcos-4.5.6-x86\_64-metal.x86\_64.raw.gz coreos.inst.ignition\_url=http://yum.example.internal:8080/ignition/ocp4-1/bootstrap.ign ip=192.168.1.100::192.168.1.1:255.255.255.0:bootstrap.ocp4-1.example.internal:ens3:none nameserver=192.168.1.12

#创建master-0启动定制文件

NODE\_NAME="master-0"

NODE\_TYPE="master"

IP=${MASTER0\_IP}

FQDN="${NODE\_NAME}.${OCP\_CLUSTER\_ID}.${DOMAIN}"

modify\_cfg ${NODE\_TYPE}

#创建worker-0启动定制文件

NODE\_NAME="worker-0"

NODE\_TYPE="worker"

IP=${WORKER0\_IP}

FQDN="${NODE\_NAME}.${OCP\_CLUSTER\_ID}.${DOMAIN}"

modify\_cfg ${NODE\_TYPE}

#创建worker-1启动定制文件

NODE\_NAME="worker-1"

NODE\_TYPE="worker"

IP=${WORKER1\_IP}

FQDN="${NODE\_NAME}.${OCP\_CLUSTER\_ID}.${DOMAIN}"

modify\_cfg ${NODE\_TYPE}

ll ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}

-rw-r--r--. 1 root root 888 May 10 10:01 bootstrap\_grub.cfg

-rw-r--r--. 1 root root 2197 May 10 10:01 bootstrap\_isolinux.cfg

-rw-r--r--. 1 root root 884 May 10 10:03 master-0\_grub.cfg

-rw-r--r--. 1 root root 2193 May 10 10:03 master-0\_isolinux.cfg

-rw-r--r--. 1 root root 884 May 10 10:04 worker-0\_grub.cfg

-rw-r--r--. 1 root root 2193 May 10 10:04 worker-0\_isolinux.cfg

-rw-r--r--. 1 root root 884 May 10 10:04 worker-1\_grub.cfg

-rw-r--r--. 1 root root 2193 May 10 10:04 worker-1\_isolinux.cfg

#### 创建定制ISO文件

yum -y install genisoimage

setVAR VOL\_ID $(isoinfo -d -i ${OCP\_PATH}/rhcos/rhcos-${RHCOS\_VER}-x86\_64-installer.x86\_64.iso |awk '/Volume set id/ {print $4}')

echo ${VOL\_ID}

rhcos-43.81.202003191953.0

cat <<EOF > ${BOOT\_FILE\_PATH}/nodes.txt

bootstrap

master-0

worker-0

worker-1

EOF

\cp -pRf ${BOOT\_FILE\_PATH}/rhcos-mnt/\* ${BOOT\_FILE\_PATH}/rhcos-temp/

for node in $(cat ${BOOT\_FILE\_PATH}/nodes.txt); do

# Overwrite the grub.cfg and isolinux.cfg files for each node type

for file in "${BOOT\_FILE\_PATH}/rhcos-temp/EFI/redhat/grub.cfg" "${BOOT\_FILE\_PATH}/rhcos-temp/isolinux/isolinux.cfg"; do

/bin/cp -f ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/${node}\_${file##\*/} ${file}

done

# As regular user!

genisoimage -U -A 'RHCOS-x86\_64' -V 'RHCOS-x86\_64' -volset ${VOL\_ID} -J -joliet-long -r -v -T \

-b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table \

-eltorito-alt-boot -efi-boot images/efiboot.img -no-emul-boot \

-o ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/${node}.iso ${BOOT\_FILE\_PATH}/rhcos-temp

done

ll ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/\*.iso

-rw-r--r--. 1 root root 88047616 May 10 10:21 /data/boot-files/rhcos-iso/ocp4-1/bootstrap.iso

-rw-r--r--. 1 root root 88047616 May 10 10:21 /data/boot-files/rhcos-iso/ocp4-1/master-0.iso

-rw-r--r--. 1 root root 88047616 May 10 10:21 /data/boot-files/rhcos-iso/ocp4-1/worker-0.iso

-rw-r--r--. 1 root root 88047616 May 10 10:21 /data/boot-files/rhcos-iso/ocp4-1/worker-1.iso

umount ${BOOT\_FILE\_PATH}/rhcos-mnt

#### 检查定制的ISO 文件

mount -o loop -t iso9660 ${RHCOS\_ISO\_PATH}/${OCP\_CLUSTER\_ID}/master-0.iso ${BOOT\_FILE\_PATH}/rhcos-mnt/

cat ${BOOT\_FILE\_PATH}/rhcos-mnt/EFI/redhat/grub.cfg | grep linux

cat ${BOOT\_FILE\_PATH}/rhcos-mnt/isolinux/isolinux.cfg | grep append

umount ${BOOT\_FILE\_PATH}/rhcos-mnt

### 验证可访问到CorsOS的ISO和RWS启动文件

systemctl restart httpd

确认访问文件链接的返回结果都是“HTTP/1.1 200 OK”。

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/ocp4-1/bootstrap.iso

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/ocp4-1/master-0.iso

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/ocp4-1/worker-0.iso

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/ocp4-1/worker-1.iso

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/rhcos-${RHCOS\_VER}-x86\_64-metal.x86\_64.raw.gz

HTTP/1.1 200 OK

Date: Sun, 10 May 2020 06:41:07 GMT

Server: Apache/2.4.6 (Red Hat Enterprise Linux)

Last-Modified: Sun, 10 May 2020 01:51:20 GMT

…

## 准备Ignition引导文件

### 安装openshift-install

tar -xzf ${OCP\_PATH}/ocp-installer/openshift-install-linux-${OCP\_VER}.tar.gz -C /usr/local/sbin/

openshift-install version

openshift-install v4.5.9

built from commit db4411451af55e0bab7258d25bdabd91ea48382f

release image quay.io/openshift-release-dev/ocp-release@sha256:1f0fd38ac0640646ab8e7fec6821c8928341ad93ac5ca3a48c513ab1fb63bc4b

### 准备install-config.yaml文件

#### 创建Ingition秘钥目录

setVAR IGN\_PATH ${BOOT\_FILE\_PATH}/ignition/${OCP\_CLUSTER\_ID} ## 存放Ignition相关文件的目录

mkdir -p ${IGN\_PATH}/ssh-key ## 存放Ignition相关秘钥的目录

#### 创建CoreOS SSH访问密钥

该密钥用于登陆CoreOS系统，获取日志等排错用途。

ssh-keygen -N '' -f ${IGN\_PATH}/ssh-key/id\_rsa

Generating public/private rsa key pair.

Your identification has been saved in /data/boot-files/ignition/ssh-key/id\_rsa.

Your public key has been saved in /data/boot-files/ignition/ssh-key/id\_rsa.pub.

The key fingerprint is:

SHA256:gbI5iI7CMjba+xKrLD/xtOsPXhJEURVJSA7U+vQn8bM root@support

The key's randomart image is:

+---[RSA 2048]----+

| .==o++o |

| . oo.. |

| o.o . |

| . o.+. .. |

|. . =o .So |

|+ o .o. o + |

|== \*o.. o o |

|Bo=.o+ E |

|+++\*=o. |

+----[SHA256]-----+

ll ${IGN\_PATH}/ssh-key

-rw------- 1 root root 1679 Mar 28 16:13 id\_rsa

-rw-r--r-- 1 root root 412 Mar 28 16:13 id\_rsa.pub

#### 创建无证书的install-config.yaml文件

先设置环境变量，然后创建无证书的install-config.yaml文件

setVAR REPLICA\_WORKER 0 ## 在安装阶段，将WORKER的数量设为0

setVAR REPLICA\_MASTER 1 ## 本文档的OpenShift集群只有1个master节点  
setVAR PULL\_SECRET "\$(jq -c . \${REG\_SECRET})" ## 在安装过程使用${REG\_SECRET}拉取镜像  
setVAR SSH\_PRI\_FILE ${IGN\_PATH}/ssh-key/id\_rsa ## 节点之间访问的私钥文件名

setVAR SSH\_PUB\_STR "\$(cat ${IGN\_PATH}/ssh-key/id\_rsa.pub)" ## 节点之间访问的公钥文件内容

echo ${PULL\_SECRET}

{"auths":{"registry.example.internal:5000":{"auth":"b3BlbnNoaWZ0OnJlZGhhdA=="}}}

echo ${SSH\_PUB\_STR}

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCpbZjzT/jN9KtVw72gcUyRx+gaZUAUlH73hFnlZha462Zag3lF6XQyDh1BEUQaMGBQ8VtrQmr0gmasinqscerfMwZCCp8cRfkFcqKRNWe00r/Offeyv8LCqM7SHxF9lNg7RT/3Qlf86Tdo0AFXV6LBJef3Dt3Uo+wsvVckYf2Qu6dzEt3Vll55GGOLBdT4VdWWXs5cbZM+GL4hyfHpcWo/hp0sjXR8Mny9Y6Ezr/cCGv1Qz5LfQP+m/sK+OIjK012lzYoYWqSqDnTlkQNalWNMh+9lTrSyrn2lw/rZBDmiXsfDhPpCsDh/c8ks6wusM+UfLX+3XIQF90VtwPb6N1eR root@support

cat << EOF > ${IGN\_PATH}/install-config.yaml

apiVersion: v1

baseDomain: ${DOMAIN}

compute:

- hyperthreading: Enabled

name: worker

replicas: ${REPLICA\_WORKER}

controlPlane:

hyperthreading: Enabled

name: master

replicas: ${REPLICA\_MASTER}

metadata:

name: ${OCP\_CLUSTER\_ID}

networking:

clusterNetworks:

- cidr: 10.128.0.0/14

hostPrefix: 23

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

none: {}

fips: false

pullSecret: '${PULL\_SECRET}'

sshKey: '${SSH\_PUB\_STR}'

imageContentSources:

- mirrors:

- ${REG\_DOMAIN}/${REPO\_NAME}

source: quay.io/openshift-release-dev/ocp-release

- mirrors:

- ${REG\_DOMAIN}/${REPO\_NAME}

source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

EOF

cat ${IGN\_PATH}/install-config.yaml

apiVersion: v1

baseDomain: example.internal

compute:

- hyperthreading: Enabled

name: worker

replicas: 0

controlPlane:

hyperthreading: Enabled

name: master

replicas: 1

metadata:

name: ocp4-1

networking:

clusterNetworks:

- cidr: 10.128.0.0/14

hostPrefix: 23

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

none: {}

fips: false

pullSecret: '{"auths":{"registry.example.internal:5000":{"auth":"b3BlbnNoaWZ0OnJlZGhhdA=="}}}'

sshKey: 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCpbZjzT/jN9KtVw72gcUyRx+gaZUAUlH73hFnlZha462Zag3lF6XQyDh1BEUQaMGBQ8VtrQmr0gmasinqscerfMwZCCp8cRfkFcqKRNWe00r/Offeyv8LCqM7SHxF9lNg7RT/3Qlf86Tdo0AFXV6LBJef3Dt3Uo+wsvVckYf2Qu6dzEt3Vll55GGOLBdT4VdWWXs5cbZM+GL4hyfHpcWo/hp0sjXR8Mny9Y6Ezr/cCGv1Qz5LfQP+m/sK+OIjK012lzYoYWqSqDnTlkQNalWNMh+9lTrSyrn2lw/rZBDmiXsfDhPpCsDh/c8ks6wusM+UfLX+3XIQF90VtwPb6N1eR root@support'

imageContentSources:

- mirrors:

- registry.example.internal:5000/ocp4/openshift4

source: quay.io/openshift-release-dev/ocp-release

- mirrors:

- registry.example.internal:5000/ocp4/openshift4

source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

#### 附加Docker Registry镜像库的证书到install-config.yaml文件

cp /etc/pki/ca-trust/source/anchors/registry.crt ${IGN\_PATH}/

sed -i -e 's/^/ /' ${IGN\_PATH}/registry.crt

echo "additionalTrustBundle: |" >> ${IGN\_PATH}/install-config.yaml

cat ${IGN\_PATH}/registry.crt >> ${IGN\_PATH}/install-config.yaml

#### 查看最终的install-config.yaml文件

重要说明：由于install-config.yaml中的安装证书有效期只有24小时，因此如果在生成该文件后24小时没有安装好OpenShift集群，需要从5.4.2重新操作生成install-config.yaml和其他所有安装前的准备步骤（所有以前生成的文件可以删除掉）。

cat ${IGN\_PATH}/install-config.yaml

apiVersion: v1

baseDomain: example.internal

compute:

- hyperthreading: Enabled

name: worker

replicas: 0

controlPlane:

hyperthreading: Enabled

name: master

replicas: 1

metadata:

name: ocp4-1

networking:

clusterNetworks:

- cidr: 10.128.0.0/14

hostPrefix: 23

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

none: {}

fips: false

pullSecret: '{"auths":{"registry.example.internal:5000":{"auth":"b3BlbnNoaWZ0OnJlZGhhdA=="}}}'

sshKey: 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCuiOyEoCpk+txEcrHjMdgtowvGhzZYCp5/IYztQ9ZPjAz2GyTclMP3nXJta067CtfFT0JrzA+zSFVbMuj7mhnTlXe/V+efoF3jUrNKDPr3Lr4MB4OmeTAxlnQeHVxTztdEuBACrkRNOS4EgQp1dCEabuAPidpi79NNR6wZvzcxGc5z2ckWufpRKzXnn1uKGKoZ+L5z4K28MmAwl2uMZz0gAb2nS6WIEJBpxgq5hh2A448nHvZi0QAJbCd4Lt5VGW7lOqfeR/ykrHDei6ToA6GusgnuUIiwb4VrCQoQrs/xSwA8P4jx+9dIG4WVcvuaoa0AlSxtwWk8v9tf5shksFYD root@support'

imageContentSources:

- mirrors:

- registry.example.internal:5000/ocp4/openshift4

source: quay.io/openshift-release-dev/ocp-release

- mirrors:

- registry.example.internal:5000/ocp4/openshift4

source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

additionalTrustBundle: |

-----BEGIN CERTIFICATE-----

MIIGLzCCBBegAwIBAgIJANCOQeP+s/1nMA0GCSqGSIb3DQEBCwUAMIGtMQswCQYD

VQQGEwJjbjEQMA4GA1UECAwHYmVpamluZzEaMBgGA1UEBwwRZ2VpamkICAgICGJl

aWppbmcxDzANBgNVBAoMBnJlZGhhdDEUMBIGA1UECwwLc2EICFBAQQgIJkExIjAg

BgNVBAMMGXJlZ2lzdHJ5LmV4YW1wbGUuaW50ZXJuYWwxJTAjBgkqhkiG9w0BCQEW

FmFkbWluQGV4YW1wbGUuaW50ZXJuYWwwHhcNMjAwNTAxMDM0MDA3WhcNMjEwNTAx

MDM0MDA3WjCBrTELMAkGA1UEBhMCY24xEDAOBgNVBAgMB2JlaWppbmcxGjAYBgNV

BAcMEWdlaWppCAgICAhiZWlqaW5nMQ8wDQYDVQQKDAZyZWRoYXQxFDASBgNVBAsM

C3NhCAhQQEEICCZBMSIwIAYDVQQDDBlyZWdpc3RyeS5leGFtcGxlLmludGVybmFs

MSUwIwYJKoZIhvcNAQkBFhZhZG1pbkBleGFtcGxlLmludGVybmFsMIICIjANBgkq

hkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAqWmEVWggiY7Y9531yiTmBhlj1Nc9nrpi

3VUIiTtbUyJlSpuqBh9XfR7Y57ox3EQJb2kvqfs5cJDX87Jm2KCInnaF1pZm9Zzc

Fu7w57uVJFcuENvT9l01V/0lmxk9D+QbEcL+fiYLkqrmCUuTrd0Me0a/QTBbIcIz

7B7DW9FMPWrmsZJXd0glhCjgn5HGi9pZlYznnAvHC2/iQNE6XcKLIexn8NczBihL

Dm1f8iaFzsPVTmcARuT2WcSc7XvIfMFvnRbT+GpT9dPYzoKDHgrdRgjOQwCQM0kb

g7c9rBsY4i0lzoGfkrufy3+d6v9n2X+CMDRJASJMEFZLQE/Py5iVGpcP7TEZroQb

r3dTQ2OSj8Kd02OQ08i1NhCmCP/npupTaYggpzPD/ggwf3dLf1s0eURqGRLrtAVu

F7Gj9TVg6xb53ckQRxaCzv2M2BhkU2RWQ0Ork1hSOmN1tPCTfZHlL1HS9x0N4iZR

m3/W3Ck0VemlGEPNd+2OEJJXDKtmu3y4SuhExZYUgUWze6bwoioenXR/PMX6W5M7

KaCgwWx6IathgieyswuJttaXOCqFk5+UXTAgjvS8AD9r/nXWMoUAy8mGLvKdMPtu

eYcvo4Bx66ImfERrPyRVwB4QsHv+GyuDWzA0D1iHOkh52KfzNXwj1GNzhqeDlnM4

yPs///KwI7ECAwEAAaNQME4wHQYDVR0OBBYEFNQG47ALG5etUrwF2CttvStMdNkW

MB8GA1UdIwQYMBaAFNQG47ALG5etUrwF2CttvStMdNkWMAwGA1UdEwQFMAMBAf8w

DQYJKoZIhvcNAQELBQADggIBAJfyXvvGM9HFK1YOqFrmKIH5V5gTD0tZhIiYVs5R

Oc5H/l2zfZiTm4Vg5p/NBlZOQ7XoYnnCl0wBrLIG+6qqE/KL+pEX3iatrQ0y8Ve8

87wj2EGuIgwBE9HUwHOJLAk8e2qlvnSr9Dwd1F/rdbk00ULY5u5UuSbJEHl3LYCg

gO0K/NPeBOoo4pGhjxPJ7AZ34bFQro0Wzu07C65NCV14W/qUjMFERfcWi2UsnhWT

+I3SExLC+cml+LLKQ0HdNzceMhnPflJkqlOXIH0QHjwtjz1Qy7YVRs9NNhGAx3sJ

veU/gEu/nmXwG7//RlrVE5xKyL9iwMw13U033Athy0KfphAtMotaWLk3s+PpwI+R

xnsQt629A79E1r0+esmMPWmDjQDWbDuVXYKn7h+nPqq5wCj/izEXhFNY8pJSnNzS

Mn3xCy0TJpLJK/DMehDsUXlXgmaWCidtxL2Gs5FrsA4nWma/UFAmwGM4gqeQl9Ll

Q5Hmi2/BUpKkz5V+sNhehO8qFlQ7vibMe+yGWbekKjWsN+aZNu3fdkucVtNaVsJN

0BDcqflZetHDzKQ/XILjnXGkXcvDOT47Z/psFHfP1NOKZZEMFW/L1hNfeqa69kGD

AgFZRAT+tQN8GgsMxNvrkSQwb7gjqACsO+1L9QQW6sMsfACQKuqqzvljYL8pXK4m

0sVe

-----END CERTIFICATE-----

#### 备份install-config.yaml文件

由于后续创建ign文件的过程会自动删除install-config.yaml文件。为了后续重复使用，可将其备份。

cp ${IGN\_PATH}/install-config.yaml{,.`date '+%s'`.bak}

ll ${IGN\_PATH}

-rw-r--r--. 1 root root 3361 May 10 10:47 install-config.yaml

-rw-r--r--. 1 root root 3361 May 10 10:49 install-config.yaml.1589078988.bak

-rw-r--r--. 1 root root 2209 May 10 10:47 registry.crt

drwxr-xr-x. 2 root root 38 May 10 10:32 ssh-key

### 创建manifest文件

openshift-install create manifests --dir ${IGN\_PATH}

tree ${IGN\_PATH}/manifests/ ${IGN\_PATH}/openshift/

manifests/

├── 04-openshift-machine-config-operator.yaml

├── cluster-config.yaml

├── cluster-dns-02-config.yml

├── cluster-infrastructure-02-config.yml

├── cluster-ingress-02-config.yml

├── cluster-network-01-crd.yml

├── cluster-network-02-config.yml

├── cluster-proxy-01-config.yaml

├── cluster-scheduler-02-config.yml

├── cvo-overrides.yaml

├── etcd-ca-bundle-configmap.yaml

├── etcd-client-secret.yaml

├── etcd-host-service-endpoints.yaml

├── etcd-host-service.yaml

├── etcd-metric-client-secret.yaml

├── etcd-metric-serving-ca-configmap.yaml

├── etcd-metric-signer-secret.yaml

├── etcd-namespace.yaml

├── etcd-service.yaml

├── etcd-serving-ca-configmap.yaml

├── etcd-signer-secret.yaml

├── image-content-source-policy-0.yaml

├── image-content-source-policy-1.yaml

├── kube-cloud-config.yaml

├── kube-system-configmap-root-ca.yaml

├── machine-config-server-tls-secret.yaml

└── openshift-config-secret-pull-secret.yaml

└── user-ca-bundle-config.yaml

openshift/

├── 99\_kubeadmin-password-secret.yaml

├── 99\_openshift-cluster-api\_master-user-data-secret.yaml

├── 99\_openshift-cluster-api\_worker-user-data-secret.yaml

├── 99\_openshift-machineconfig\_99-master-ssh.yaml

├── 99\_openshift-machineconfig\_99-worker-ssh.yaml

└── openshift-install-manifests.yaml

0 directories, 34 files

修改mastersSchedulable为false。

sed -i 's/mastersSchedulable: true/mastersSchedulable: false/g' ${IGN\_PATH}/manifests/cluster-scheduler-02-config.yml

cat ${IGN\_PATH}/manifests/cluster-scheduler-02-config.yml

apiVersion: config.openshift.io/v1

kind: Scheduler

metadata:

creationTimestamp: null

name: cluster

spec:

mastersSchedulable: false

policy:

name: ""

status: {}

### 创建Ignition引导文件

openshift-install create ignition-configs --dir ${IGN\_PATH}/

ll ${IGN\_PATH}/\*.ign

-rw-r-----. 1 root root 296788 May 10 10:52 /data/boot-files/ignition/ocp4-1/bootstrap.ign

-rw-r-----. 1 root root 1828 May 10 10:51 /data/boot-files/ignition/ocp4-1/master.ign

-rw-r-----. 1 root root 1828 May 10 10:51 /data/boot-files/ignition/ocp4-1/worker.ign

jq .ignition.config ${IGN\_PATH}/master.ign

{

"append": [

{

"source": "https://api-int.ocp4-1.example.internal:22623/config/master",

"verification": {}

}

]

}

jq .ignition.config ${IGN\_PATH}/worker.ign

{

"append": [

{

"source": "https://api-int.ocp4-1.example.internal:22623/config/worker",

"verification": {}

}

]

}

### 创建Ignition引导文件下载目录

chmod 604 ${IGN\_PATH}/\*.ign

cat << EOF > /etc/httpd/conf.d/ignition.conf

Alias /ignition "${IGN\_PATH}/../"

<Directory "${IGN\_PATH}/../">

Options +Indexes +FollowSymLinks

Require all granted

</Directory>

<Location /ignition >

SetHandler None

</Location>

EOF

systemctl restart httpd

curl http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/bootstrap.ign

curl http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/master.ign

curl http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/worker.ign

# 创建Bootstrap、Master、Worker虚拟机节点

具体根据不同的IaaS环境和虚机节点配置要求创建bootstrap、master-0、worker-0、worker-1虚拟机节点，方法和过程略。**需要注意以下事项**：

1. 将硬盘的启动优先级设置为最高。每个虚拟机挂载对应名称的iso文件作为启动盘。
2. 使用网桥类型的网络。

# 安装OCP集群

**注意**：如无说明，本节的命令在Support节点上运行。如有说明，请按说明在对应节点操作。

[Suport]

[root@support ~]#

确认可以正常下载以下文件：

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/rhcos-${RHCOS\_VER}-x86\_64-metal.x86\_64.raw.gz

HTTP/1.1 200 OK

Date: Sun, 10 May 2020 06:41:07 GMT

Server: Apache/2.4.6 (Red Hat Enterprise Linux)

Last-Modified: Sun, 10 May 2020 01:51:20 GMT

ETag: "2fe55e7d-5a5417783d323"

Accept-Ranges: bytes

Content-Length: 803561085

Content-Type: application/x-gzip

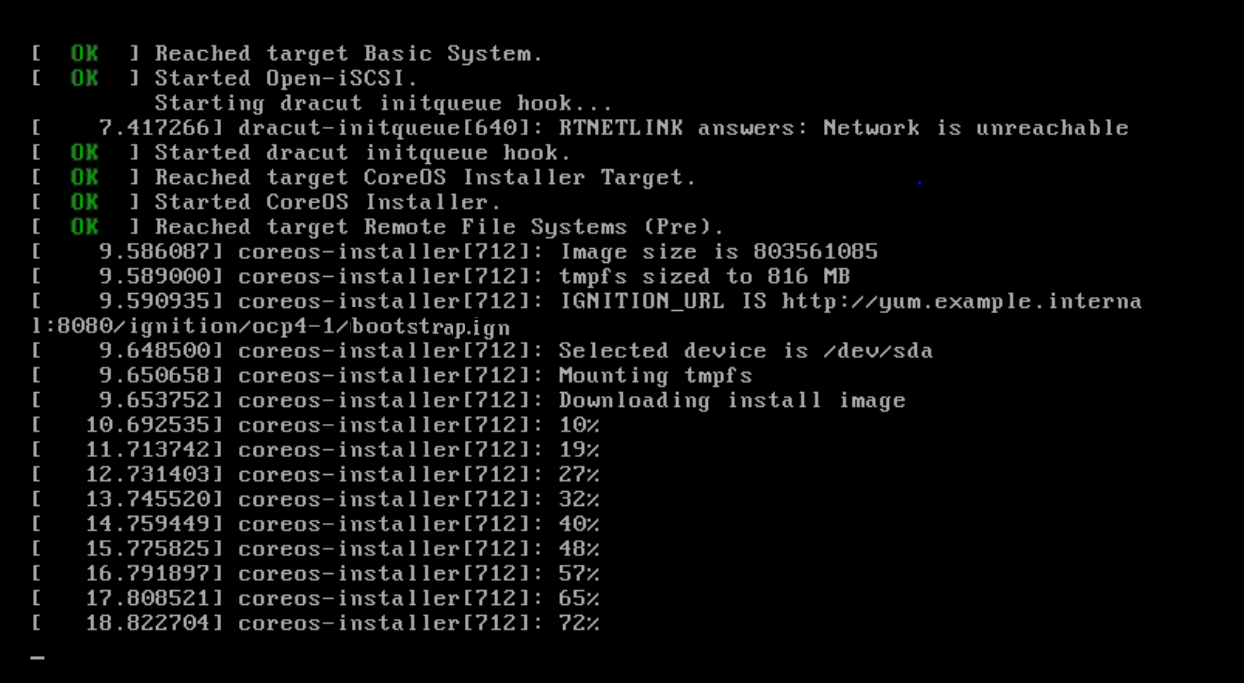
curl -I -s http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/bootstrap.ign

curl -I -s http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/master.ign

curl -I -s http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/worker.ign

## 第一阶段：部署bootstrap阶段

用生成的bootstrap.iso文件启动bootstrap节点，等待bootstrap到login界面。注意如下图所示的关键阶段，如果出现错误，请参见第9章安装常见错误。



待bootstrap启动出现login界面后，在Support节点远程检查bootstrap的相关配置：

1. 复制证书到用户缺省目录，以便可以用证书登录bootstrap、master和worker节点。

mkdir ~/.kube

cp ${IGN\_PATH}/auth/kubeconfig ~/.kube/config

1. 检查bootstrap节点的镜像库mirror配置是否按照install-config.yaml的内容进行配置

rm -rf ~/.ssh/known\_hosts

ssh -i ${SSH\_PRI\_FILE} core@bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN} "sudo cat /etc/containers/registries.conf"

[[registry]]

location = "quay.io/openshift-release-dev/ocp-release"

insecure = false

mirror-by-digest-only = true

[[registry.mirror]]

location = "registry.example.internal:5000/ocp4/openshift4"

insecure = false

[[registry]]

location = "quay.io/openshift-release-dev/ocp-v4.0-art-dev"

insecure = false

mirror-by-digest-only = true

[[registry.mirror]]

location = "registry.example.internal:5000/ocp4/openshift4"

insecure = false

1. 跟踪bootstrap的日志以识别安装进度，当循环出现如下红色字体提示的内容的时候，并且haproxy的web监控界面openshift-api-server和machine-config-server的bootstrap部分变为绿色时，说明bootstrap的引导服务已经启动，此时可进入下一个阶段。

ssh -i ${SSH\_PRI\_FILE} core@bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN} "journalctl -b -f -u bootkube.service"

Mar 24 08:55:07 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: Error: unhealthy cluster

Mar 24 08:55:07 bootstrap.ocp4-1.example.internal podman[5552]: 2020-03-24 08:55:07.756792774 +0000 UTC m=+5.633717216 container died a2236978cc89998463f0f35cce9aa38be2c87cd44511822e3cf780ce6b1ba4f3 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Mar 24 08:55:07 bootstrap.ocp4-1.example.internal podman[5552]: 2020-03-24 08:55:07.857573367 +0000 UTC m=+5.734497764 container remove a2236978cc89998463f0f35cce9aa38be2c87cd44511822e3cf780ce6b1ba4f3 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Mar 24 08:55:07 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: etcdctl failed. Retrying in 5 seconds...

Mar 24 08:55:12 bootstrap.ocp4-1.example.internal podman[5617]: 2020-03-24 08:55:12.998090624 +0000 UTC m=+0.112860888 container create 022e4ae97e2ea26d55ec5485277b35b521173b8655efa940fabf66b65c353ac6 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Mar 24 08:55:13 bootstrap.ocp4-1.example.internal podman[5617]: 2020-03-24 08:55:13.404615467 +0000 UTC m=+0.519386260 container init 022e4ae97e2ea26d55ec5485277b35b521173b8655efa940fabf66b65c353ac6 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Mar 24 08:55:13 bootstrap.ocp4-1.example.internal podman[5617]: 2020-03-24 08:55:13.42201796 +0000 UTC m=+0.536788219 container start 022e4ae97e2ea26d55ec5485277b35b521173b8655efa940fabf66b65c353ac6 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Mar 24 08:55:13 bootstrap.ocp4-1.example.internal podman[5617]: 2020-03-24 08:55:13.422499989 +0000 UTC m=+0.537270213 container attach 022e4ae97e2ea26d55ec5485277b35b521173b8655efa940fabf66b65c353ac6 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

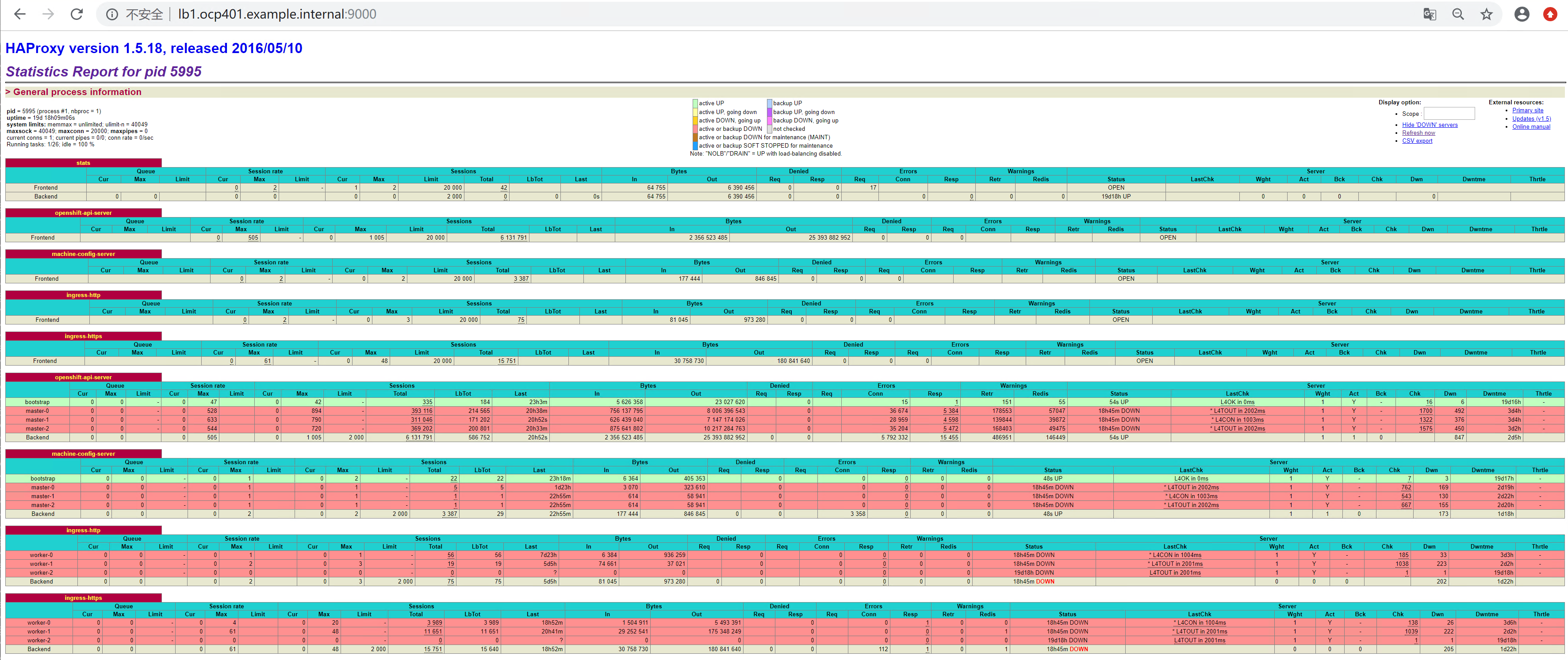
Mar 24 08:55:18 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: {"level":"warn","ts":"2020-03-24T08:55:18.436Z","caller":"clientv3/retry\_interceptor.go:61","msg":"retrying of unary invoker failed","target":"endpoint://client-d7c98144-0f92-4ea8-ac8a-f33ef2e57d60/etcd-0.ocp4-1.example.internal:2379","attempt":0,"error":"rpc error: code = DeadlineExceeded desc = latest connection error: connection error: desc = \"transport: Error while dialing dial tcp 192.168.1.101:2379: connect: connection refused\""}

Mar 24 08:55:18 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: {"level":"warn","ts":"2020-03-24T08:55:18.437Z","caller":"clientv3/retry\_interceptor.go:61","msg":"retrying of unary invoker failed","target":"endpoint://client-f3064bc0-5d13-44b6-b538-0223be5cbab8/etcd-2.ocp4-1.example.internal:2379","attempt":0,"error":"rpc error: code = DeadlineExceeded desc = latest connection error: connection error: desc = \"transport: Error while dialing dial tcp 192.168.30.53:2379: connect: no route to host\""}

Mar 24 08:55:18 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: {"level":"warn","ts":"2020-03-24T08:55:18.437Z","caller":"clientv3/retry\_interceptor.go:61","msg":"retrying of unary invoker failed","target":"endpoint://client-ed4e4a5a-28ba-4f9a-bb1f-487ab44cf978/etcd-1.ocp4-1.example.internal:2379","attempt":0,"error":"rpc error: code = DeadlineExceeded desc = latest connection error: connection error: desc = \"transport: Error while dialing dial tcp 192.168.30.52:2379: connect: no route to host\""}

Mar 24 08:55:18 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: https://etcd-0.ocp4-1.example.internal:2379 is unhealthy: failed to commit proposal: context deadline exceeded

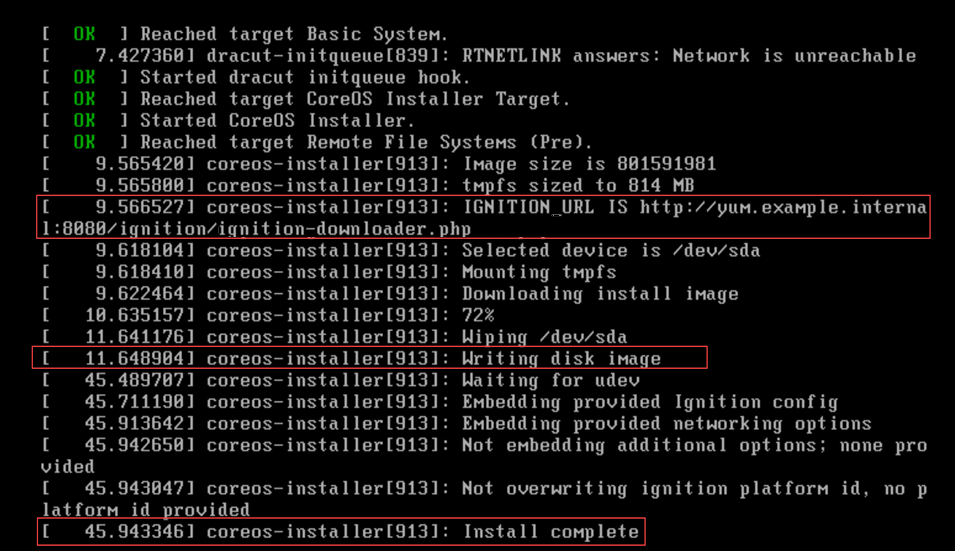
访问如下地址<http://lb.ocp4-1.example.internal:9000/>，结果和下图一致。



## 第二阶段：部署master阶段

用对应的master iso文件启动master节点。如出现下述红色提示错误，请检查是否能访问如下链接：

http://yum.example.internal:8080/ignition/ocp4-1/master.ign





待master启动到login界面后，在support节点检查master节点的镜像库配置是否按照install-config.yaml的内容进行配置

ssh -i ${SSH\_PRI\_FILE} core@master-0.${OCP\_CLUSTER\_ID}.${DOMAIN} "sudo cat /etc/containers/registries.conf"

unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

[[registry]]

prefix = ""

location = "quay.io/openshift-release-dev/ocp-release"

mirror-by-digest-only = true

[[registry.mirror]]

location = "registry.example.internal:5000/ocp4/openshift4"

[[registry]]

prefix = ""

location = "quay.io/openshift-release-dev/ocp-v4.0-art-dev"

mirror-by-digest-only = true

[[registry.mirror]]

location = "registry.example.internal:5000/ocp4/openshift4"

检查是否能够正常访问registry

ssh -i ${SSH\_PRI\_FILE} core@master-0.${OCP\_CLUSTER\_ID}.${DOMAIN} \

"curl -s -u openshift:redhat https://registry.${DOMAIN}:5000/v2/\_catalog"

{"repositories":["ocp4/openshift4"]}

注意：OCP 4.5 缺省屏蔽了单master节点的集群，此时master节点在安装过程中会反复启动。因此在安装集群单master集群的时候，可以在support节点运行以下命令开启单master集群特性，随后master节点会自动继续进行安装。如果是安装3 master节点的集群，可以忽略此命令。

oc patch etcd cluster -p='{"spec": {"unsupportedConfigOverrides": {"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}' --type=merge

安装过程中可以通过查看如下日志来跟踪安装过程。注意以下日志的红色字体部分，这些内容指示master的不同安装阶段

ssh -i ${SSH\_PRI\_FILE} core@bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN} "journalctl -b -f -u bootkube.service"

Apr 02 07:52:32 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: etcdctl failed. Retrying in 5 seconds...

Apr 02 07:52:37 bootstrap.ocp4-1.example.internal podman[9743]: 2020-04-02 07:52:37.420734353 +0000 UTC m=+0.123608378 container create 01ee65f2e086dc78da5b19b44e6e953e5407c9f0fde0fdc97bfadcfdd193048b (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:37 bootstrap.ocp4-1.example.internal podman[9743]: 2020-04-02 07:52:37.762838208 +0000 UTC m=+0.465712180 container init 01ee65f2e086dc78da5b19b44e6e953e5407c9f0fde0fdc97bfadcfdd193048b (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:37 bootstrap.ocp4-1.example.internal podman[9743]: 2020-04-02 07:52:37.775831629 +0000 UTC m=+0.478705654 container start 01ee65f2e086dc78da5b19b44e6e953e5407c9f0fde0fdc97bfadcfdd193048b (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:37 bootstrap.ocp4-1.example.internal podman[9743]: 2020-04-02 07:52:37.776199842 +0000 UTC m=+0.479073820 container attach 01ee65f2e086dc78da5b19b44e6e953e5407c9f0fde0fdc97bfadcfdd193048b (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: {"level":"warn","ts":"2020-04-02T07:52:42.788Z","caller":"clientv3/retry\_interceptor.go:61","msg":"retrying of unary invoker failed","target":"endpoint://client-d6e57854-3cf5-4b1e-b3a2-943cabd23c4a/etcd-1.ocp4-1.example.internal:2379","attempt":0,"error":"rpc error: code = DeadlineExceeded desc = latest connection error: connection error: desc = \"transport: Error while dialing dial tcp 192.168.30.52:2379: connect: connection refused\""}

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: https://etcd-2.ocp4-1.example.internal:2379 is healthy: successfully committed proposal: took = 15.233356ms

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: https://etcd-0.ocp4-1.example.internal:2379 is healthy: successfully committed proposal: took = 15.47325ms

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: https://etcd-1.ocp4-1.example.internal:2379 is unhealthy: failed to commit proposal: context deadline exceeded

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: Error: unhealthy cluster

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal podman[9743]: 2020-04-02 07:52:42.807847779 +0000 UTC m=+5.510721908 container died 01ee65f2e086dc78da5b19b44e6e953e5407c9f0fde0fdc97bfadcfdd193048b (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal podman[9743]: 2020-04-02 07:52:42.837268375 +0000 UTC m=+5.540142504 container remove 01ee65f2e086dc78da5b19b44e6e953e5407c9f0fde0fdc97bfadcfdd193048b (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:42 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: etcdctl failed. Retrying in 5 seconds...

Apr 02 07:52:47 bootstrap.ocp4-1.example.internal podman[9835]: 2020-04-02 07:52:47.996789518 +0000 UTC m=+0.136384452 container create 829cebcb4c046f09d4da6e9e51a917935c86109f4b6fec532012d61adee90e53 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9835]: 2020-04-02 07:52:48.334257301 +0000 UTC m=+0.473852230 container init 829cebcb4c046f09d4da6e9e51a917935c86109f4b6fec532012d61adee90e53 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9835]: 2020-04-02 07:52:48.349547358 +0000 UTC m=+0.489142318 container start 829cebcb4c046f09d4da6e9e51a917935c86109f4b6fec532012d61adee90e53 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9835]: 2020-04-02 07:52:48.349647887 +0000 UTC m=+0.489242831 container attach 829cebcb4c046f09d4da6e9e51a917935c86109f4b6fec532012d61adee90e53 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: https://etcd-2.ocp4-1.example.internal:2379 is healthy: successfully committed proposal: took = 13.075766ms

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: https://etcd-0.ocp4-1.example.internal:2379 is healthy: successfully committed proposal: took = 13.965011ms

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: https://etcd-1.ocp4-1.example.internal:2379 is healthy: successfully committed proposal: took = 15.674414ms

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9835]: 2020-04-02 07:52:48.389774977 +0000 UTC m=+0.529370029 container died 829cebcb4c046f09d4da6e9e51a917935c86109f4b6fec532012d61adee90e53 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9835]: 2020-04-02 07:52:48.422634478 +0000 UTC m=+0.562229468 container remove 829cebcb4c046f09d4da6e9e51a917935c86109f4b6fec532012d61adee90e53 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:bc97106373ffddb5b7afe2a9010de54098c95545a654c7990b882680528c29e3, name=etcdctl)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: etcd cluster up. Killing etcd certificate signer...

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9905]: 2020-04-02 07:52:48.64857007 +0000 UTC m=+0.203975960 container died c22cbeb9e8a5b804f5c46c27cff4f025925f7a490ac04eb4dd37ef9b3c76e67c (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:8c364dc68a8a58e7f9867dd04cbcfe7427fda3c418440b986daeee348fa017be, name=etcd-signer)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9905]: 2020-04-02 07:52:48.649691905 +0000 UTC m=+0.205097737 container stop c22cbeb9e8a5b804f5c46c27cff4f025925f7a490ac04eb4dd37ef9b3c76e67c (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:8c364dc68a8a58e7f9867dd04cbcfe7427fda3c418440b986daeee348fa017be, name=etcd-signer)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal podman[9905]: 2020-04-02 07:52:48.696206682 +0000 UTC m=+0.251612569 container remove c22cbeb9e8a5b804f5c46c27cff4f025925f7a490ac04eb4dd37ef9b3c76e67c (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:8c364dc68a8a58e7f9867dd04cbcfe7427fda3c418440b986daeee348fa017be, name=etcd-signer)

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: c22cbeb9e8a5b804f5c46c27cff4f025925f7a490ac04eb4dd37ef9b3c76e67c

Apr 02 07:52:48 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: Starting cluster-bootstrap...

Apr 02 07:52:50 bootstrap.ocp4-1.example.internal podman[9953]: 2020-04-02 07:52:50.847699724 +0000 UTC m=+2.126817243 image pull

Apr 02 07:52:50 bootstrap.ocp4-1.example.internal podman[9953]: 2020-04-02 07:52:50.86769796 +0000 UTC m=+2.146815411 container create 8e4e96ce73d726b6ec9f41831b3750c71b07f7d614f4ece6366bbd19b93cc05a (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b, name=amazing\_gates)

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal podman[9953]: 2020-04-02 07:52:51.22092578 +0000 UTC m=+2.500043267 container init 8e4e96ce73d726b6ec9f41831b3750c71b07f7d614f4ece6366bbd19b93cc05a (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b, name=amazing\_gates)

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal podman[9953]: 2020-04-02 07:52:51.235761057 +0000 UTC m=+2.514878510 container start 8e4e96ce73d726b6ec9f41831b3750c71b07f7d614f4ece6366bbd19b93cc05a (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b, name=amazing\_gates)

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal podman[9953]: 2020-04-02 07:52:51.236017917 +0000 UTC m=+2.515135386 container attach 8e4e96ce73d726b6ec9f41831b3750c71b07f7d614f4ece6366bbd19b93cc05a (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b, name=amazing\_gates)

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: Starting temporary bootstrap control plane...

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: E0402 07:52:51.254786 1 reflector.go:134] github.com/openshift/cluster-bootstrap/pkg/start/status.go:66: Failed to list \*v1.Pod: Get https://localhost:6443/api/v1/pods: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#1] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#2] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#3] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:51 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#4] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:52 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#5] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:52 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: E0402 07:52:52.258247 1 reflector.go:134] github.com/openshift/cluster-bootstrap/pkg/start/status.go:66: Failed to list \*v1.Pod: Get https://localhost:6443/api/v1/pods: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:52 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#6] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:52 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#7] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:52 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#8] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:52 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#9] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:53 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: [#10] failed to fetch discovery: Get https://localhost:6443/api?timeout=32s: dial tcp [::1]:6443: connect: connection refused

Apr 02 07:52:53 bootstrap.ocp4-1.example.internal bootkube.sh[2796]: E0402 07:52:53.259075 1 reflector.go:134] github.com/openshift/cluster-bootstrap/pkg/start/status.go:66: Failed to list \*v1.Pod: Get https://localhost:6443/api/v1/pods: dial tcp [::1]:6443: connect: connection refused

...

econfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#142] failed to create some manifests:

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#143] failed to create some manifests:

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#144] failed to create some manifests:

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#145] failed to create some manifests:

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:04:55 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#1400] failed to create some manifests:

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#1401] failed to create some manifests:

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: [#1402] failed to create some manifests:

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-master-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-master-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: "99\_openshift-machineconfig\_99-worker-ssh.yaml": unable to get REST mapping for "99\_openshift-machineconfig\_99-worker-ssh.yaml": no matches for kind "MachineConfig" in version "machineconfiguration.openshift.io/v1"

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Created "99\_openshift-machineconfig\_99-master-ssh.yaml" machineconfigs.v1.machineconfiguration.openshift.io/99-master-ssh -n

Mar 30 10:09:07 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Created "99\_openshift-machineconfig\_99-worker-ssh.yaml" machineconfigs.v1.machineconfiguration.openshift.io/99-worker-ssh -n

Mar 30 10:09:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:09:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:09:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler DoesNotExist

Mar 30 10:09:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:10:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:10:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Pending

Mar 30 10:10:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:10:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:10:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:10:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:10:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler RunningNotReady

Mar 30 10:10:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:10:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Pending

Mar 30 10:10:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:10:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:10:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:10:49 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:10:49 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler RunningNotReady

Mar 30 10:10:49 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:10:49 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:11:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:11:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Ready

Mar 30 10:11:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:11:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:12:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Pending

Mar 30 10:12:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager DoesNotExist

Mar 30 10:12:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:12:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver DoesNotExist

Mar 30 10:12:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Ready

Mar 30 10:12:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager Pending

Mar 30 10:12:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:12:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver Pending

Mar 30 10:12:54 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager RunningNotReady

Mar 30 10:12:54 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:12:54 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver RunningNotReady

Mar 30 10:12:54 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler RunningNotReady

Mar 30 10:13:04 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:13:04 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver RunningNotReady

Mar 30 10:13:04 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Ready

Mar 30 10:13:04 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager RunningNotReady

Mar 30 10:13:09 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-controller-manager/kube-controller-manager Ready

Mar 30 10:13:09 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-cluster-version/cluster-version-operator Ready

Mar 30 10:13:09 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-apiserver/kube-apiserver Ready

Mar 30 10:13:09 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Pod Status:openshift-kube-scheduler/openshift-kube-scheduler Ready

Mar 30 10:13:09 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: All self-hosted control plane components successfully started

Mar 30 10:13:09 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Sending bootstrap-success event.Waiting for remaining assets to be created.

Mar 30 10:13:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_00\_cluster-version-operator\_00\_namespace.yaml" namespaces.v1./openshift-cluster-version -n as it already exists

Mar 30 10:13:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_00\_cluster-version-operator\_01\_clusteroperator.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/clusteroperators.config.openshift.io -n as it already exists

Mar 30 10:13:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_00\_cluster-version-operator\_01\_clusterversion.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/clusterversions.config.openshift.io -n as it already exists

Mar 30 10:13:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_00\_cluster-version-operator\_02\_roles.yaml" clusterrolebindings.v1.rbac.authorization.k8s.io/cluster-version-operator -n as it already exists

Mar 30 10:13:14 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_00\_cluster-version-operator\_03\_deployment.yaml" deployments.v1.apps/cluster-version-operator -n openshift-cluster-version as it already exists

Mar 30 10:13:15 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_03\_authorization-openshift\_01\_rolebindingrestriction.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/rolebindingrestrictions.authorization.openshift.io -n as it already exists

Mar 30 10:13:15 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_03\_config-operator\_01\_operatorhub.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/operatorhubs.config.openshift.io -n as it already exists

Mar 30 10:13:15 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_03\_config-operator\_01\_proxy.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/proxies.config.openshift.io -n as it already exists

Mar 30 10:13:16 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_03\_quota-openshift\_01\_clusterresourcequota.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/clusterresourcequotas.quota.openshift.io -n as it already exists

Mar 30 10:13:16 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_03\_security-openshift\_01\_scc.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/securitycontextconstraints.security.openshift.io -n as it already exists

Mar 30 10:13:17 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_apiserver.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/apiservers.config.openshift.io -n as it already exists

Mar 30 10:13:17 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_authentication.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/authentications.config.openshift.io -n as it already exists

Mar 30 10:13:17 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_build.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/builds.config.openshift.io -n as it already exists

Mar 30 10:13:18 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_console.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/consoles.config.openshift.io -n as it already exists

Mar 30 10:13:18 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_dns.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/dnses.config.openshift.io -n as it already exists

Mar 30 10:13:19 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_featuregate.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/featuregates.config.openshift.io -n as it already exists

Mar 30 10:13:19 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_image.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/images.config.openshift.io -n as it already exists

Mar 30 10:13:19 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_imagecontentsourcepolicy.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/imagecontentsourcepolicies.operator.openshift.io -n as it already exists

Mar 30 10:13:20 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_infrastructure.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/infrastructures.config.openshift.io -n as it already exists

Mar 30 10:13:20 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_ingress.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/ingresses.config.openshift.io -n as it already exists

Mar 30 10:13:21 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_network.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/networks.config.openshift.io -n as it already exists

Mar 30 10:13:21 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_oauth.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/oauths.config.openshift.io -n as it already exists

Mar 30 10:13:21 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_project.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/projects.config.openshift.io -n as it already exists

Mar 30 10:13:22 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0000\_10\_config-operator\_01\_scheduler.crd.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/schedulers.config.openshift.io -n as it already exists

Mar 30 10:13:22 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "0001\_00\_cluster-version-operator\_03\_service.yaml" services.v1./cluster-version-operator -n openshift-cluster-version as it already exists

Mar 30 10:13:23 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "00\_openshift-kube-apiserver-ns.yaml" namespaces.v1./openshift-kube-apiserver -n as it already exists

Mar 30 10:13:23 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "00\_openshift-kube-apiserver-operator-ns.yaml" namespaces.v1./openshift-kube-apiserver-operator -n as it already exists

Mar 30 10:13:23 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "00\_openshift-kube-controller-manager-ns.yaml" namespaces.v1./openshift-kube-controller-manager -n as it already exists

Mar 30 10:13:24 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "00\_openshift-kube-controller-manager-operator-ns.yaml" namespaces.v1./openshift-kube-controller-manager-operator -n as it already exists

Mar 30 10:13:24 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "00\_openshift-kube-scheduler-ns.yaml" namespaces.v1./openshift-kube-scheduler -n as it already exists

Mar 30 10:13:25 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "04-openshift-machine-config-operator.yaml" namespaces.v1./openshift-machine-config-operator -n as it already exists

Mar 30 10:13:25 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "99\_kubeadmin-password-secret.yaml" secrets.v1./kubeadmin -n kube-system as it already exists

Mar 30 10:13:25 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "99\_openshift-cluster-api\_master-user-data-secret.yaml" secrets.v1./master-user-data -n openshift-machine-api as it already exists

Mar 30 10:13:26 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "99\_openshift-cluster-api\_worker-user-data-secret.yaml" secrets.v1./worker-user-data -n openshift-machine-api as it already exists

Mar 30 10:13:26 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "99\_openshift-machineconfig\_99-master-ssh.yaml" machineconfigs.v1.machineconfiguration.openshift.io/99-master-ssh -n as it already exists

Mar 30 10:13:27 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "99\_openshift-machineconfig\_99-worker-ssh.yaml" machineconfigs.v1.machineconfiguration.openshift.io/99-worker-ssh -n as it already exists

Mar 30 10:13:27 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cco-cloudcredential\_v1\_credentialsrequest.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/credentialsrequests.cloudcredential.openshift.io -n as it already exists

Mar 30 10:13:27 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cco-namespace.yaml" namespaces.v1./openshift-cloud-credential-operator -n as it already exists

Mar 30 10:13:28 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-config.yaml" configmaps.v1./cluster-config-v1 -n kube-system as it already exists

Mar 30 10:13:28 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-dns-02-config.yml" dnses.v1.config.openshift.io/cluster -n as it already exists

Mar 30 10:13:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-infrastructure-02-config.yml" infrastructures.v1.config.openshift.io/cluster -n as it already exists

Mar 30 10:13:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-ingress-00-custom-resource-definition.yaml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/ingresscontrollers.operator.openshift.io -n as it already exists

Mar 30 10:13:29 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-ingress-00-namespace.yaml" namespaces.v1./openshift-ingress-operator -n as it already exists

Mar 30 10:13:30 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-ingress-02-config.yml" ingresses.v1.config.openshift.io/cluster -n as it already exists

Mar 30 10:13:30 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-network-01-crd.yml" customresourcedefinitions.v1beta1.apiextensions.k8s.io/networks.operator.openshift.io -n as it already exists

Mar 30 10:13:31 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-network-02-config.yml" networks.v1.config.openshift.io/cluster -n as it already exists

Mar 30 10:13:31 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-proxy-01-config.yaml" proxies.v1.config.openshift.io/cluster -n as it already exists

Mar 30 10:13:31 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-role-binding-kube-apiserver.yaml" clusterrolebindings.v1.rbac.authorization.k8s.io/kube-apiserver -n as it already exists

Mar 30 10:13:32 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-role-kube-apiserver.yaml" clusterroles.v1.rbac.authorization.k8s.io/kube-apiserver -n as it already exists

Mar 30 10:13:32 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cluster-scheduler-02-config.yml" schedulers.v1.config.openshift.io/cluster -n as it already exists

Mar 30 10:13:33 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "configmap-admin-kubeconfig-client-ca.yaml" configmaps.v1./admin-kubeconfig-client-ca -n openshift-config as it already exists

Mar 30 10:13:33 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "configmap-csr-controller-ca.yaml" configmaps.v1./csr-controller-ca -n openshift-config-managed as it already exists

Mar 30 10:13:33 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "configmap-sa-token-signing-certs.yaml" configmaps.v1./sa-token-signing-certs -n openshift-config-managed as it already exists

Mar 30 10:13:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "csr-bootstrap-role-binding.yaml" clusterrolebindings.v1.rbac.authorization.k8s.io/system-bootstrap-node-bootstrapper -n as it already exists

Mar 30 10:13:34 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "cvo-overrides.yaml" clusterversions.v1.config.openshift.io/version -n openshift-cluster-version as it already exists

Mar 30 10:13:35 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-ca-bundle-configmap.yaml" configmaps.v1./etcd-ca-bundle -n openshift-config as it already exists

Mar 30 10:13:35 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-client-secret.yaml" secrets.v1./etcd-client -n openshift-config as it already exists

Mar 30 10:13:35 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-host-service-endpoints.yaml" endpoints.v1./host-etcd -n openshift-etcd as it already exists

Mar 30 10:13:36 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-host-service.yaml" services.v1./host-etcd -n openshift-etcd as it already exists

Mar 30 10:13:36 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-metric-client-secret.yaml" secrets.v1./etcd-metric-client -n openshift-config as it already exists

Mar 30 10:13:37 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-metric-serving-ca-configmap.yaml" configmaps.v1./etcd-metric-serving-ca -n openshift-config as it already exists

Mar 30 10:13:37 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-metric-signer-secret.yaml" secrets.v1./etcd-metric-signer -n openshift-config as it already exists

Mar 30 10:13:37 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-namespace.yaml" namespaces.v1./openshift-etcd -n as it already exists

Mar 30 10:13:38 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-service.yaml" services.v1./etcd -n openshift-etcd as it already exists

Mar 30 10:13:38 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-serving-ca-configmap.yaml" configmaps.v1./etcd-serving-ca -n openshift-config as it already exists

Mar 30 10:13:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "etcd-signer-secret.yaml" secrets.v1./etcd-signer -n openshift-config as it already exists

Mar 30 10:13:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "image-content-source-policy-0.yaml" imagecontentsourcepolicies.v1alpha1.operator.openshift.io/image-policy-0 -n as it already exists

Mar 30 10:13:39 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "image-content-source-policy-1.yaml" imagecontentsourcepolicies.v1alpha1.operator.openshift.io/image-policy-1 -n as it already exists

Mar 30 10:13:40 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "kube-apiserver-serving-ca-configmap.yaml" configmaps.v1./initial-kube-apiserver-server-ca -n openshift-config as it already exists

Mar 30 10:13:40 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "kube-cloud-config.yaml" secrets.v1./kube-cloud-cfg -n kube-system as it already exists

Mar 30 10:13:41 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "kube-system-configmap-root-ca.yaml" configmaps.v1./root-ca -n kube-system as it already exists

Mar 30 10:13:41 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "machine-config-server-tls-secret.yaml" secrets.v1./machine-config-server-tls -n openshift-machine-config-operator as it already exists

Mar 30 10:13:41 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "openshift-config-secret-pull-secret.yaml" secrets.v1./pull-secret -n openshift-config as it already exists

Mar 30 10:13:42 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "openshift-install-manifests.yaml" configmaps.v1./openshift-install-manifests -n openshift-config as it already exists

Mar 30 10:13:42 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-aggregator-client-signer.yaml" secrets.v1./aggregator-client-signer -n openshift-kube-apiserver-operator as it already exists

Mar 30 10:13:43 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-control-plane-client-signer.yaml" secrets.v1./kube-control-plane-signer -n openshift-kube-apiserver-operator as it already exists

Mar 30 10:13:43 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-csr-signer-signer.yaml" secrets.v1./csr-signer-signer -n openshift-kube-controller-manager-operator as it already exists

Mar 30 10:13:43 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-initial-kube-controller-manager-service-account-private-key.yaml" secrets.v1./initial-service-account-private-key -n openshift-config as it already exists

Mar 30 10:13:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-kube-apiserver-to-kubelet-signer.yaml" secrets.v1./kube-apiserver-to-kubelet-signer -n openshift-kube-apiserver-operator as it already exists

Mar 30 10:13:44 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-loadbalancer-serving-signer.yaml" secrets.v1./loadbalancer-serving-signer -n openshift-kube-apiserver-operator as it already exists

Mar 30 10:13:45 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-localhost-serving-signer.yaml" secrets.v1./localhost-serving-signer -n openshift-kube-apiserver-operator as it already exists

Mar 30 10:13:45 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "secret-service-network-serving-signer.yaml" secrets.v1./service-network-serving-signer -n openshift-kube-apiserver-operator as it already exists

Mar 30 10:13:45 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Skipped "user-ca-bundle-config.yaml" configmaps.v1./user-ca-bundle -n openshift-config as it already exists

Mar 30 10:13:46 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: Tearing down temporary bootstrap control plane...

Mar 30 10:13:46 bootstrap.ocp4-1.example.internal podman[7436]: 2020-03-30 10:13:46.073685613 +0000 UTC m=+608.955116142 container died 5eaecb122e1ab742fd135e15e15b737406c3fb483b08100a58864a8affe70414 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b, name=gifted\_khayyam)

Mar 30 10:13:46 bootstrap.ocp4-1.example.internal podman[7436]: 2020-03-30 10:13:46.161873823 +0000 UTC m=+609.043304257 container remove 5eaecb122e1ab742fd135e15e15b737406c3fb483b08100a58864a8affe70414 (image=quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:222fbfd3323ec347babbda1a66929019221fcee82cfc324a173b39b218cf6c4b, name=gifted\_khayyam)

Mar 30 10:13:46 bootstrap.ocp4-1.example.internal bootkube.sh[2710]: bootkube.service complete

出现上述最后两条红色字体后，说明bootstrap的任务已经完成，可以已经进入后续安装部署节点

另外，我们也可以通过如下方法了解安装进程：

tail -f ${IGN\_PATH}/.openshift\_install.log

time="2020-04-02T15:32:12+08:00" level=debug msg=" Fetching Install Config..."

time="2020-04-02T15:32:12+08:00" level=debug msg=" Reusing previously-fetched Install Config"

time="2020-04-02T15:32:12+08:00" level=debug msg="Generating Metadata..."

time="2020-04-02T16:00:54+08:00" level=debug msg="OpenShift Installer v4.5.9"

time="2020-04-02T16:00:54+08:00" level=debug msg="Built from commit 82f9a63c06956b3700a69475fbd14521e139aa1e"

time="2020-04-02T16:00:54+08:00" level=info msg="Waiting up to 30m0s for the Kubernetes API at https://api.ocp4-1.example.internal:6443..."

time="2020-04-02T16:00:54+08:00" level=info msg="API v1.18.3 up"

time="2020-04-02T16:00:54+08:00" level=info msg="Waiting up to 30m0s for bootstrapping to complete..."

time="2020-04-02T16:00:54+08:00" level=debug msg="Bootstrap status: complete"

time="2020-04-02T16:00:54+08:00" level=info msg="It is now safe to remove the bootstrap resources"

openshift-install wait-for bootstrap-complete --log-level debug --dir ${IGN\_PATH}

DEBUG OpenShift Installer 4.5.9

DEBUG Built from commit db4411451af55e0bab7258d25bdabd91ea48382f

INFO Waiting up to 30m0s for the Kubernetes API at https://api.ocp4-1.example.internal:6443...

INFO Waiting up to 30m0s for the Kubernetes API at https://api.ocp4-1.example.internal:6443...

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: the server could not find the requested resource

DEBUG Still waiting for the Kubernetes API: Get https://api.ocp4-1.example.internal:6443/version?timeout=32s: EOF

INFO API v1.14.6+8bbaf43 up

INFO Waiting up to 30m0s for bootstrapping to complete...

DEBUG Bootstrap status: complete

INFO It is now safe to remove the bootstrap resources

real 11m33.655s

user 0m0.396s

sys 0m0.175s

到此，master节点配置已完成，我们可以关闭bootstrap节点点，继续进行下一个阶段部署。

ssh -i ${SSH\_PRI\_FILE} core@bootstrap.${OCP\_CLUSTER\_ID}.${DOMAIN} "sudo shutdown"

|  |  |  |  |
| --- | --- | --- | --- |
|  | 💡 NOTE | |  |
|  |  | 在安装过程中，也可以通过以下方法查看master节点的日志 |  |
| ssh -i ${SSH\_PRI\_FILE} core@master-0.${OCP\_CLUSTER\_ID}.${DOMAIN} "journalctl -xef" |

## 第三阶段：配置master阶段

本阶段无需人工干预，仅需要根据以下操作跟踪master节点的自动配置过程即可。

检查节点状态，确保master的STATUS均为Ready状态

oc get node

NAME STATUS ROLES AGE VERSION

master-0.ocp4-1.example.internal Ready master 34m v1.18.3

检查master配置过程

oc get clusteroperators

NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE

authentication Unknown Unknown True 8m32s

cloud-credential 4.5.9 True False False 15m

cluster-autoscaler 4.5.9 True False False 6m28s

console 4.5.9 Unknown True False 7m28s

dns 4.5.9 True False False 10m

image-registry 4.5.9 True False False 7m6s

ingress unknown False True True 7m1s

insights 4.5.9 True False False 12m

kube-apiserver 4.5.9 True False False 9m41s

kube-controller-manager 4.5.9 True False False 9m46s

kube-scheduler 4.5.9 True False False 9m28s

machine-api 4.5.9 True False False 10m

machine-config 4.5.9 True False False 9m57s

marketplace 4.5.9 True False False 6m25s

monitoring False True True 2m

network 4.5.9 True False False 11m

node-tuning 4.5.9 True False False 8m18s

openshift-apiserver 4.5.9 True False False 7m50s

openshift-controller-manager 4.5.9 True False False 9m36s

openshift-samples 4.5.9 True False False 3m6s

operator-lifecycle-manager 4.5.9 True False False 11m

operator-lifecycle-manager-catalog 4.5.9 True False False 11m

operator-lifecycle-manager-packageserver 4.5.9 True False False 8m14s

service-ca 4.5.9 True False False 11m

service-catalog-apiserver 4.5.9 True False False 8m33s

service-catalog-controller-manager 4.5.9 True False False 8m33s

storage 4.5.9 True False False 7m5s

当出现下面两种提示，则说明可始进行下一阶段部署。

oc get clusterversion

NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version False True 13m Working towards 4.5.9: 99% complete, waiting on authentication, console, ingress, monitoring

oc get clusterversion

NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version False True 17m Unable to apply 4.5.9: some cluster operators have not yet rolled out

## 第四阶段：部署worker阶段

启动worker节点。然后通过如下命令，等待出现csr批准请求。

oc project default

watch -n 5 oc get csr

NAME AGE SIGNERNAME REQUESTOR CONDITION

csr-2fb7m 12m kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-2n9x2 54m kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-59drg 10m kubernetes.io/kubelet-serving system:node:worker-0.ocp4.example.internal Approved,Issued

csr-d295j 12m kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-mjf2f 54m kubernetes.io/kubelet-serving system:node:master-0.ocp4.example.internal Approved,Issued

csr-ssr8t 10m kubernetes.io/kubelet-serving system:node:worker-1.ocp4.example.internal Approved,Issued

oc get csr -o name | xargs oc adm certificate approve

certificatesigningrequest.certificates.k8s.io/csr-2jl6p approved

certificatesigningrequest.certificates.k8s.io/csr-7f62t approved

certificatesigningrequest.certificates.k8s.io/csr-f9l8b approved

certificatesigningrequest.certificates.k8s.io/csr-jj785 approved

certificatesigningrequest.certificates.k8s.io/csr-mb9zf approved

certificatesigningrequest.certificates.k8s.io/csr-s9szw approved

oc get csr

NAME AGE SIGNERNAME REQUESTOR CONDITION

csr-2fb7m 10m kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-2n9x2 52m kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-59drg 7m58s kubernetes.io/kubelet-serving system:node:worker-0.ocp4.example.internal Approved,Issued

csr-d295j 10m kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-mjf2f 51m kubernetes.io/kubelet-serving system:node:master-0.ocp4.example.internal Approved,Issued

csr-ssr8t 7m59s kubernetes.io/kubelet-serving system:node:worker-1.ocp4.example.internal Approved,Issued

oc get node

NAME STATUS ROLES AGE VERSION

master-0.ocp4-1.example.internal Ready master 56m v1.18.3

worker-0.ocp4-1.example.internal NotReady worker 39s v1.18.3

worker-1.ocp4-1.example.internal NotReady worker 33s v1.18.3

oc get node

NAME STATUS ROLES AGE VERSION

master-0.ocp4-1.example.internal Ready master 58m v1.18.3

worker-0.ocp4-1.example.internal Ready worker 108s v1.18.3

worker-1.ocp4-1.example.internal Ready worker 102s v1.18.3

oc get clusteroperators

NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE

authentication 4.5.9 True False False 61m

cloud-credential 4.5.9 True False False 118m

cluster-autoscaler 4.5.9 True False False 102m

console 4.5.9 True False False 61m

dns 4.5.9 True False False 113m

image-registry 4.5.9 True False False 109m

ingress 4.5.9 True False False 73m

insights 4.5.9 True False False 109m

kube-apiserver 4.5.9 True False False 112m

kube-controller-manager 4.5.9 True False False 112m

kube-scheduler 4.5.9 True False False 109m

machine-api 4.5.9 True False False 113m

machine-config 4.5.9 True False False 113m

marketplace 4.5.9 True False False 104m

monitoring 4.5.9 True False False 58m

network 4.5.9 True False False 115m

node-tuning 4.5.9 True False False 105m

openshift-apiserver 4.5.9 True False False 110m

openshift-controller-manager 4.5.9 True False False 109m

openshift-samples 4.5.9 True False False 98m

operator-lifecycle-manager 4.5.9 True False False 109m

operator-lifecycle-manager-catalog 4.5.9 True False False 109m

operator-lifecycle-manager-packageserver 4.5.9 True False False 61m

service-ca 4.5.9 True False False 115m

service-catalog-apiserver 4.5.9 True False False 108m

service-catalog-controller-manager 4.5.9 True False False 108m

storage 4.5.9 True False False 105m

执行以下命令来查看集群部署是否完成。以下红色字体部分，说明集群已经部署完成。请记下kubeadmin和对应的登录密码。

oc get clusterversion

NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version 4.5.9 True False 2m54s Cluster version is 4.5.9

tail -f ${IGN\_PATH}/.openshift\_install.log

time="2020-04-02T18:32:24+08:00" level=info msg="Waiting up to 30m0s for the cluster at https://api.ocp4-1.example.internal:6443 to initialize..."

time="2020-04-02T18:32:24+08:00" level=debug msg="Cluster is initialized"

time="2020-04-02T18:32:24+08:00" level=info msg="Waiting up to 10m0s for the openshift-console route to be created..."

time="2020-04-02T18:32:24+08:00" level=debug msg="Route found in openshift-console namespace: console"

time="2020-04-02T18:32:24+08:00" level=debug msg="Route found in openshift-console namespace: downloads"

time="2020-04-02T18:32:24+08:00" level=debug msg="OpenShift console route is created"

time="2020-04-02T18:32:24+08:00" level=info msg="Install complete!"

time="2020-04-02T18:32:24+08:00" level=info msg="To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/data/ocp/upi/baremetal/auth/kubeconfig'"

time="2020-04-02T18:32:24+08:00" level=info msg="Access the OpenShift web-console here: https://console-openshift-console.apps.ocp4-1.example.internal"

time="2020-04-02T18:32:24+08:00" level=info msg="Login to the console with user: **kubeadmin, password: u26K4-pFVtr-a5WEZ-rYd2X**"

openshift-install wait-for install-complete --log-level debug --dir ${IGN\_PATH}

INFO Waiting up to 30m0s for the cluster at https://api.ocp4-1.example.internal:6443 to initialize...

INFO Waiting up to 10m0s for the openshift-console route to be created...

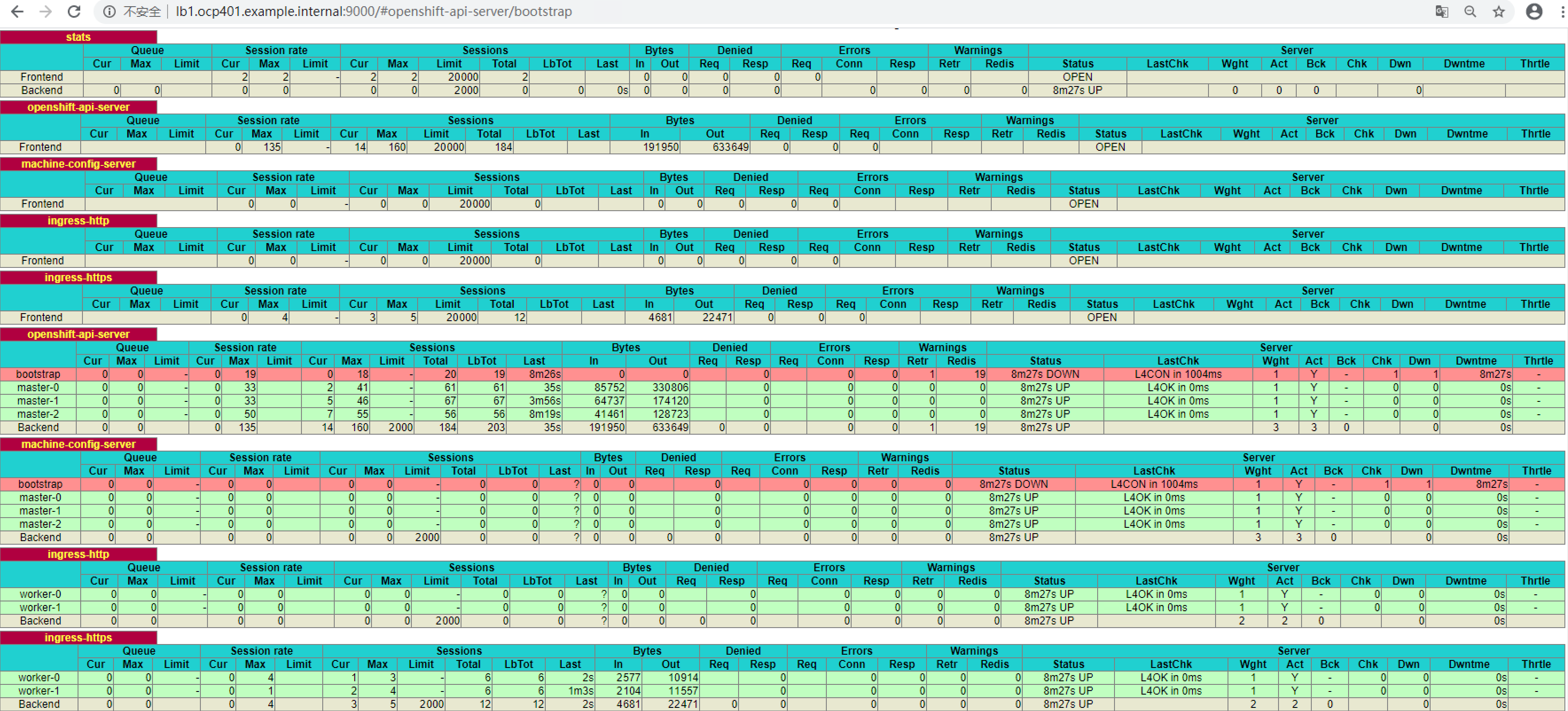
INFO Install complete!

INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/data/ocp/upi/baremetal/auth/kubeconfig'

INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp4-1.example.internal

INFO Login to the console with user: kubeadmin, password: u26K4-pFVtr-a5WEZ-rYd2X

观察haproxy监控页面，如下所示：

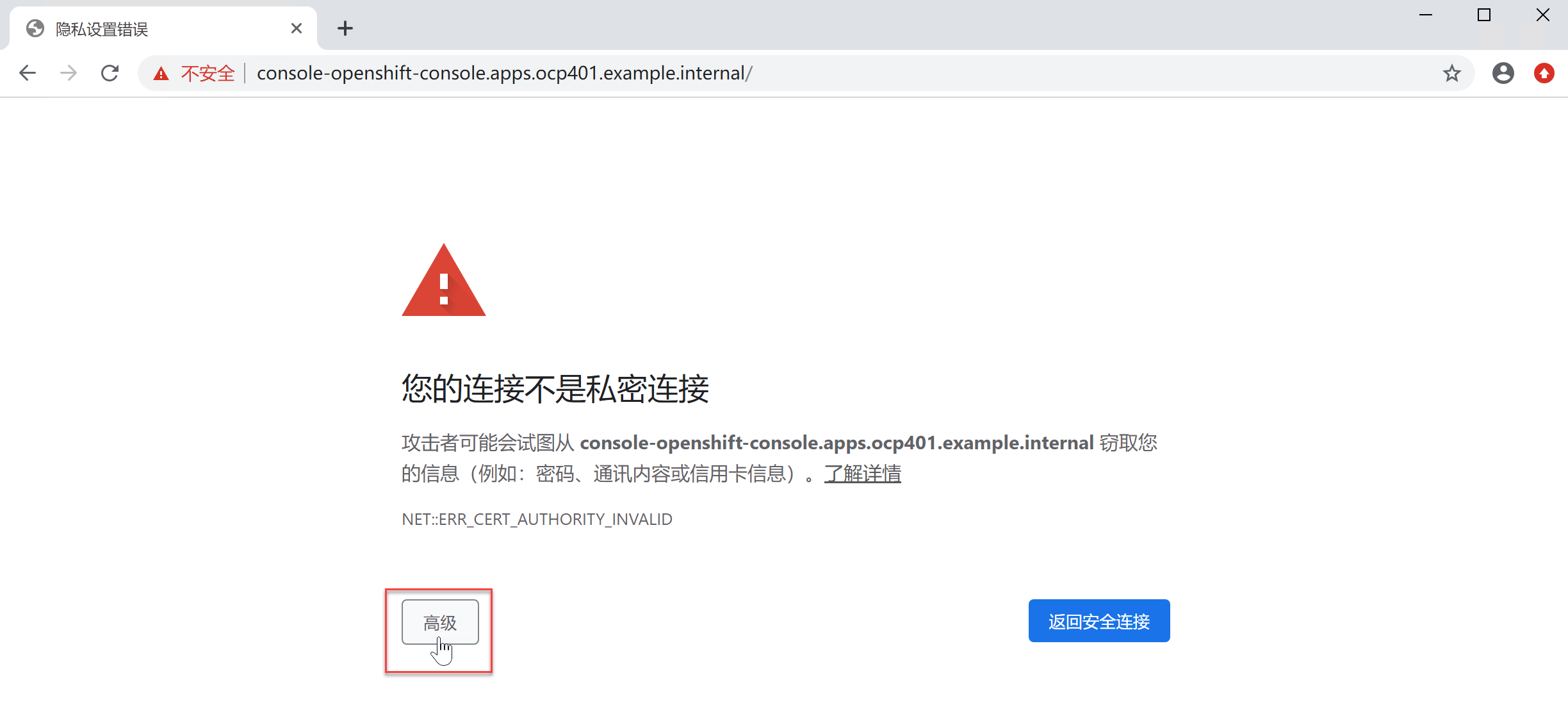


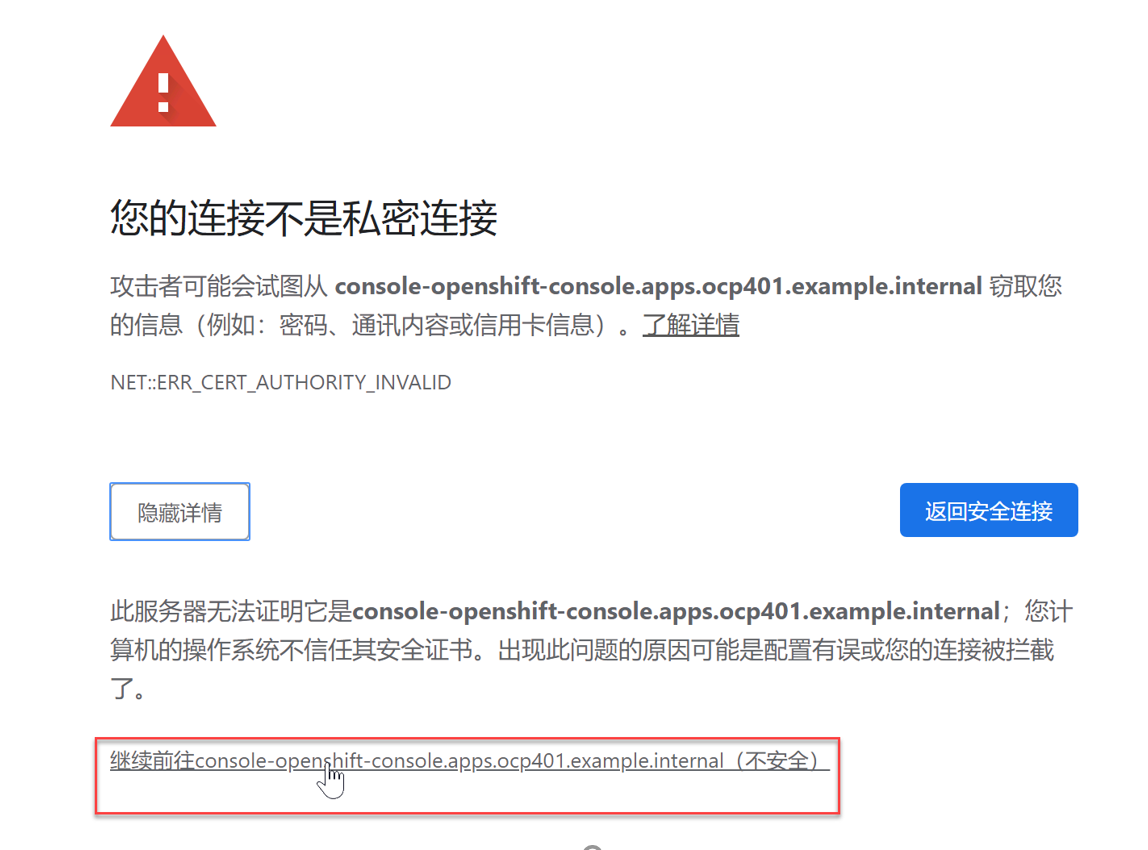
注意：为了解析域名，可在运行浏览器节点的hosts文件中增加以下IP和域名的解析对应。

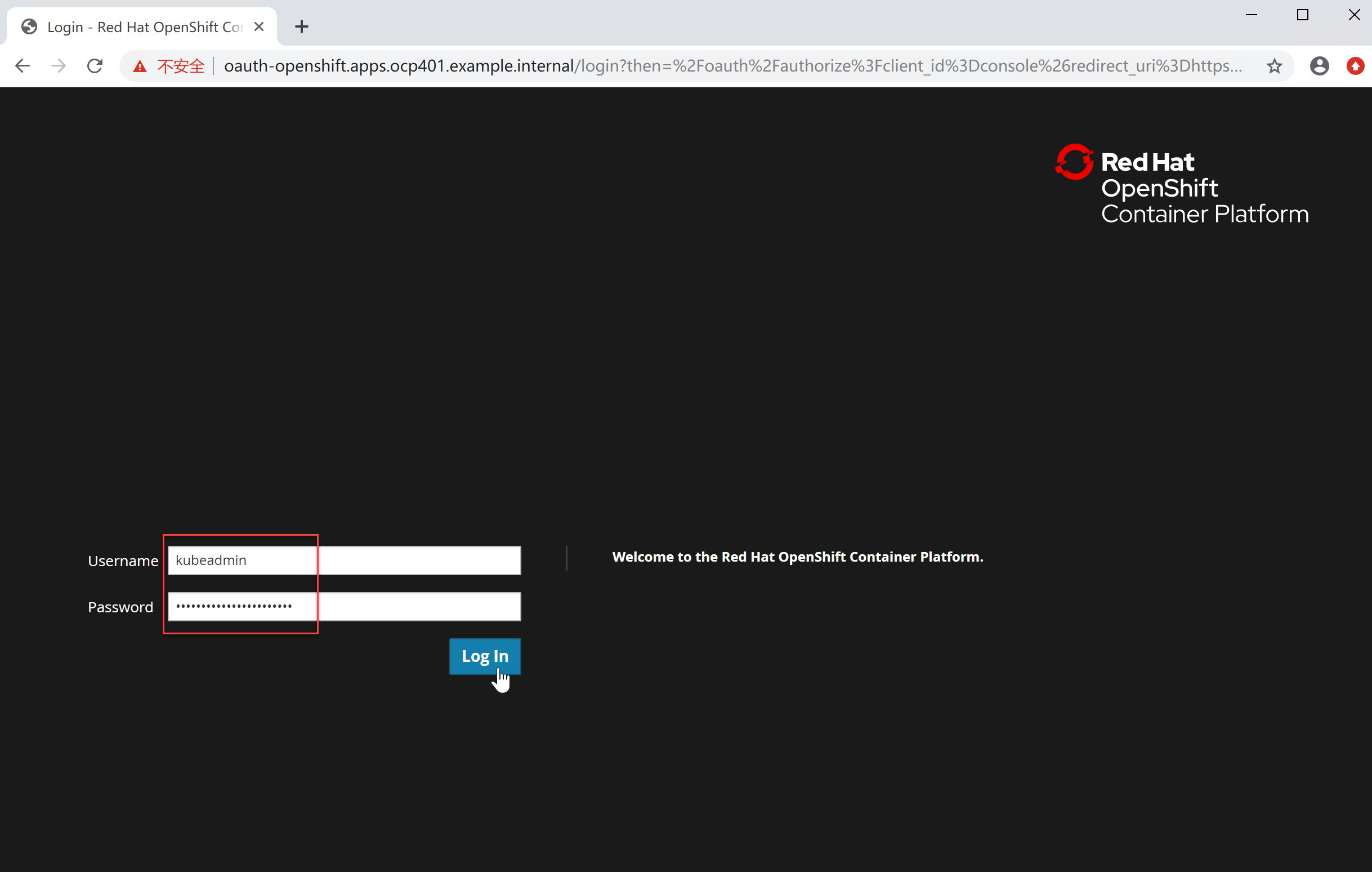
<SUPPORT\_IP> console-openshift-console.apps.ocp4.example.internal

<SUPPORT\_IP> oauth-openshift.apps.ocp4.example.internal

在浏览器中访问https://console-openshift-console.apps.ocp4-1.example.internal，进入OpenShift Console







# 集群初始化和功能验证

[Support]

[root@support~]#

## 用户管理

### 新建集群管理员

创建包含admin用户和对应密码的文件：users.htpasswd

htpasswd -bBc users.htpasswd admin P@ssw0rd

基于users.htpasswd文件创建secret验证库

oc create secret generic htpass-secret --from-file=htpasswd=users.htpasswd -n openshift-config

创建基于HTPasswd的IdentityProvider，并提供验证库htpass-secret

cat << EOF | oc apply -f -

---

apiVersion: config.openshift.io/v1

kind: OAuth

metadata:

name: cluster

spec:

identityProviders:

- name: htpasswd\_provider

mappingMethod: claim

type: HTPasswd

htpasswd:

fileData:

name: htpass-secret

EOF

授予admin用户cluster-admin权限

oc adm policy add-cluster-role-to-user cluster-admin admin --rolebinding-name=cluster-admin

clusterrole.rbac.authorization.k8s.io/cluster-admin added: "admin"

oc describe clusterrolebindings cluster-admin

Name: cluster-admin

Labels: kubernetes.io/bootstrapping=rbac-defaults

Annotations: rbac.authorization.kubernetes.io/autoupdate: true

Role:

Kind: ClusterRole

Name: cluster-admin

Subjects:

Kind Name Namespace

---- ---- ---------

Group system:masters

User admin

用admin登陆

oc login https://api.${OCP\_CLUSTER\_ID}.${DOMAIN}:6443 -u admin -p P@ssw0rd

oc whoami

admin

### 新建普通用户

htpasswd -b users.htpasswd user1 P@ssw0rd

cat users.htpasswd

admin:$2y$05$hDddUJfAR1PMTf6QgEsfE.Q6pxDEDj6cN0d1h/OtkzX.5WqFJN6IG

user1:$apr1$5kQD/0.H$Dzt1HEGhn0gnmIZGLTnOu1

更新用户认证库secret

oc create secret generic htpass-secret --from-file=htpasswd=users.htpasswd -n openshift-config --dry-run -o yaml | oc apply -f -

添加授权

oc adm policy add-cluster-role-to-user admin user1

登陆验证

oc login https://api.${OCP\_CLUSTER\_ID}.${DOMAIN}:6443 -u user1 -p P@ssw0rd

oc get identity

NAME IDP NAME IDP USER NAME USER NAME USER UID

my\_htpasswd\_provider:admin my\_htpasswd\_provider admin admin 039fef4f-c21f-443f-97fd-3967f350f1fd

my\_htpasswd\_provider:user1 my\_htpasswd\_provider user1 user1 4f27f69e-1135-4e1e-9233-8501ac786d3c

### 删除kubeadmin

在上述步骤完成后，特别是添加了具有cluster-admin role的用户后，即可删除kubeadmin用户

oc delete secrets kubeadmin -n kube-system

|  |  |  |  |
| --- | --- | --- | --- |
|  | 🔥 IMPORTANT | |  |
|  |  | 特别要注意，上述命令不可逆，如果之前没有创建cluster-admin role的用户替换kubeadmin，那么整个集群需要重装！ |  |
| https://docs.openshift.com/container-platform/4.5/authentication/remove-kubeadmin.html |
|  |

## 部署BusyBox应用

### 导入BusyBox的应用镜像

setVAR BUSYBOX\_IMG\_PATH ${OCP\_PATH}/app-image/thirdparty/busybox

skopeo copy --dest-creds=openshift:redhat docker-archive:${BUSYBOX\_IMG\_PATH}/busybox\_1.31.1.tar.gz \

docker://${REG\_DOMAIN}/apps/busybox:1.31.1

skopeo copy --dest-creds=openshift:redhat docker-archive:${BUSYBOX\_IMG\_PATH}/busybox\_1.31.1.tar.gz \

docker://${REG\_DOMAIN}/apps/busybox:latest

skopeo inspect --creds=openshift:redhat docker://${REG\_DOMAIN}/apps/busybox:latest

{

"Name": "registry.example.internal:5000/apps/busybox",

"Digest": "sha256:46fa766829563a5df50f5ddd8e8334e369faf79fc019a43a78fff81dc8f3e635",

"RepoTags": [

"1.31.1",

"latest"

"Created": "2020-03-10T00:19:33.019716493Z",

"DockerVersion": "18.09.7",

"Labels": null,

"Architecture": "amd64",

"Os": "linux",

"Layers": [

"sha256:c72287a86f7bef1ed4720698b73b32ba197b517dee46a4e1066096a3f0c3540f"

],

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

]

}

### 部署BusyBox应用

oc new-project busybox

cat << EOF | oc apply -f -

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: busybox

namespace: busybox

labels:

app: busybox

spec:

replicas: 1

selector:

matchLabels:

app: busybox

template:

metadata:

labels:

app: busybox

spec:

containers:

- name: pod-backend

image: ${REG\_DOMAIN}/apps/busybox:latest

command: ["sleep"]

args: ["1000"]

EOF

oc get pod

NAME READY STATUS RESTARTS AGE

pod/busybox-74b54595cf-wbd7d 1/1 Running 0 3m3s

oc rsh $(oc get pod | grep busybox | awk '{print $1}') echo HelloWorld

HelloWorld

## 部署NFS服务

### 安装NFS服务

yum -y install nfs-utils

systemctl enable nfs-server --now

systemctl status nfs-server

● nfs-server.service - NFS server and services

Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)

Active: active (exited) since Tue 2020-05-05 16:49:01 CST; 1min 13s ago

Process: 6247 ExecStartPost=/bin/sh -c if systemctl -q is-active gssproxy; then systemctl reload gssproxy ; fi (code=exited, status=0/SUCCESS)

Process: 6230 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)

Process: 6227 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)

Main PID: 6230 (code=exited, status=0/SUCCESS)

CGroup: /system.slice/nfs-server.service

创建NFS共享目录

mkdir -p /data/nfs/openshift

echo /data/nfs/openshift \*'(rw,root\_squash)' >> /etc/exports.d/openshift.exports

chown -R nfsnobody.nfsnobody /data/nfs

exportfs -rav

exporting \*:/data/nfs/openshift

### 验证NFS共享目录

[Bastion]

[root@bastion ~]#

安装NFS客户端

yum -y install nfs-utils

ll /sbin/mount\*

-rwsr-xr-x. 1 root root 117504 Sep 27 2018 /sbin/mount.nfs

lrwxrwxrwx. 1 root root 9 Aug 17 15:58 /sbin/mount.nfs4 -> mount.nfs

-rwxr-xr-x. 1 root root 41563 Sep 27 2018 /sbin/mountstats

验证NFS服务

showmount -e nfs.example.internal

Export list for nfs.example.internal:

/data/nfs/openshift \*

mount -t nfs nfs.example.internal:/data/nfs/openshift /mnt

df /mnt

Filesystem 1K-blocks Used Available Use% Mounted on

nfs.example.internal:/data/nfs/openshift 254802944 32799744 222003200 13% /mnt

## 配置OpenShift内部镜像库的存储

### 创建内部镜像库使用的NFS目录

mkdir -p /data/nfs/registry

echo /data/nfs/registry \*'(rw,sync,no\_wdelay,no\_root\_squash,insecure,fsid=0)' >> /etc/exports.d/registry.exports

cat /etc/exports.d/registry.exports

chown -R nfsnobody.nfsnobody /data/nfs

exportfs -rav | grep registry

exporting \*:/data/nfs/registry

### 创建PV

cat << EOF | oc create -f -

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv-registry

spec:

capacity:

storage: 50Gi

accessModes:

- ReadWriteMany

persistentVolumeReclaimPolicy: Retain

nfs:

path: /data/nfs/registry

server: nfs.${DOMAIN}

readOnly: false

EOF

oc get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

pv-registry 50Gi RWX Retain Available 3s

### 创建PVC

oc project openshift-image-registry

cat << EOF | oc create -f -

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-registry

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 50Gi

EOF

oc get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

pvc-registry Bound pv-registry 50Gi RWX 3s

### 指定内部镜像库使用PVC

oc get pod -n openshift-image-registry

NAME READY STATUS RESTARTS AGE

cluster-image-registry-operator-5d7d64d769-xpxhc 2/2 Running 0 7d22h

oc patch configs.imageregistry.operator.openshift.io cluster --type merge \

--patch '{"spec":{"storage":{"pvc":{"claim":"pvc-registry"}}}}'

oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState": "Managed"}}'

oc get configs.imageregistry.operator.openshift.io -o json | jq -r '.items[].spec |.managementState,.storage'

Managed

{

"pvc": {

"claim": "pvc-registry"

}

}

oc get pod -n openshift-image-registry

NAME READY STATUS RESTARTS AGE

cluster-image-registry-operator-5d7d64d769-czmss 2/2 Running 2 24h

image-registry-65cd459795-h4p2h 1/1 Running 0 3m18s

node-ca-nvkv4 1/1 Running 0 3m21s

node-ca-rrt2h 1/1 Running 0 3m21s

node-ca-z6gl8 1/1 Running 0 3m21s

## 配置NFS StorageClass

为了OpenShift应用能够使用存储，本文将以NFS为例说明如何为OpenShift添加StorageClass。

### 创建NFS 目录

mkdir -p /data/nfs/userfile

chown -R nfsnobody.nfsnobody /data/nfs/userfile

chmod -R 777 /data/nfs

echo /data/nfs/userfile \*'(rw,sync,no\_wdelay,no\_root\_squash,insecure)' > /etc/exports.d/userfile.exports

systemctl restart nfs-server

exportfs -rav | grep userfile

exporting \*:/data/nfs/userfil

showmount -e | grep userfile

/data/nfs/userfile \*

### 创建NFS StorageClass部署配置

#### 导入NFS Client镜像

setVAR NFS\_NAMESPACE csi-nfs

skopeo copy --dest-creds=openshift:redhat \

docker-archive:${OCP\_PATH}/csi/nfs/nfs-client-provisioner\_v3.1.0-k8s1.11.tar.gz \

docker://${REG\_DOMAIN}/${NFS\_NAMESPACE}/nfs-client-provisioner:v3.1.0-k8s1.11

skopeo copy --dest-creds=openshift:redhat \

docker-archive:${OCP\_PATH}/csi/nfs/nfs-client-provisioner\_v3.1.0-k8s1.11.tar.gz \

docker://${REG\_DOMAIN}/${NFS\_NAMESPACE}/nfs-client-provisioner:latest

curl -u openshift:redhat https://${REG\_DOMAIN}/v2/\_catalog

{"repositories":["csi-nfs/nfs-client-provisioner","ocp4/openshift4"]}

skopeo inspect \

--creds=openshift:redhat \

docker://${REG\_DOMAIN}/${NFS\_NAMESPACE}/nfs-client-provisioner:latest

{

"Name": "registry.example.internal:5000/csi-nfs/nfs-client-provisioner",

"Digest": "sha256:ded8d7cbd32ecf65f75cb515bd7889470ce0320d823ac14d74041d213594953d",

"RepoTags": [

"v3.1.0-k8s1.11",

"latest"

],

"Created": "2018-08-30T16:09:44.430156478Z",

"DockerVersion": "17.09.0-ce",

"Labels": null,

"Architecture": "amd64",

"Os": "linux",

"Layers": [

"sha256:7334f0d97ab2fac7cdfbab641a615d41a0b4c779b58e070d714a844a6a3b8c37",

"sha256:4e73054bc5d94e9ca5026d3addc9cfe038fe483768e9dcd4b6a13686277c9c5f",

"sha256:31c9d9043fe24796b67271cac778658dcc92206f0eea47f84e9b6b7ee881be62"

],

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

]

}

#### 从Docker Registry中删除镜像（可选）

1. 先允许删除镜像

sed -i 's/enabled: false/enabled: true/g' /etc/docker-distribution/registry/config.yml

1. 重启Docker Registry

systemctl restart docker-distribution

1. 删除容器镜像

skopeo delete --creds=openshift:redhat docker://${REG\_DOMAIN}/${NFS\_NAMESPACE}/nfs-client-provisioner:latest

1. 删除blobs/layers中的镜像垃圾

registry garbage-collect /etc/docker-distribution/registry/config.yml

1. 重启Docker Registry，释放缓存

systemctl restart docker-distribution

#### 创建配置文件目录

setVAR NFS\_DEPLOY\_PATH ${OCP\_PATH}/csi/nfs/deploy

mkdir -p ${NFS\_DEPLOY\_PATH}

#### 创建rbac.yaml文件

cat << EOF > ${NFS\_DEPLOY\_PATH}/rbac.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: ${NFS\_NAMESPACE}

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: nfs-client-provisioner-runner

rules:

- apiGroups: [""]

resources: ["persistentvolumes"]

verbs: ["get", "list", "watch", "create", "delete"]

- apiGroups: [""]

resources: ["persistentvolumeclaims"]

verbs: ["get", "list", "watch", "update"]

- apiGroups: ["storage.k8s.io"]

resources: ["storageclasses"]

verbs: ["get", "list", "watch"]

- apiGroups: [""]

resources: ["events"]

verbs: ["create", "update", "patch"]

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: run-nfs-client-provisioner

subjects:

- kind: ServiceAccount

name: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: ${NFS\_NAMESPACE}

roleRef:

kind: ClusterRole

name: nfs-client-provisioner-runner

apiGroup: rbac.authorization.k8s.io

---

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: leader-locking-nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: ${NFS\_NAMESPACE}

rules:

- apiGroups: [""]

resources: ["endpoints"]

verbs: ["get", "list", "watch", "create", "update", "patch"]

---

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: leader-locking-nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: ${NFS\_NAMESPACE}

subjects:

- kind: ServiceAccount

name: nfs-client-provisioner

# replace with namespace where provisioner is deployed

namespace: ${NFS\_NAMESPACE}

roleRef:

kind: Role

name: leader-locking-nfs-client-provisioner

apiGroup: rbac.authorization.k8s.io

EOF

#### 创建deployment.yaml文件

setVAR NFS\_PATH "/data/nfs/userfile"

setVAR NFS\_DOMAIN nfs.${DOMAIN}

cat << EOF > ${NFS\_DEPLOY\_PATH}/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nfs-client-provisioner

labels:

app: nfs-client-provisioner

spec:

replicas: 1

strategy:

type: Recreate

selector:

matchLabels:

app: nfs-client-provisioner

template:

metadata:

labels:

app: nfs-client-provisioner

spec:

serviceAccountName: nfs-client-provisioner

containers:

- name: nfs-client-provisioner

image: ${REG\_DOMAIN}/${NFS\_NAMESPACE}/nfs-client-provisioner:latest

volumeMounts:

- name: nfs-client-root

mountPath: /persistentvolumes

env:

- name: PROVISIONER\_NAME

value: kubernetes-nfs

- name: NFS\_SERVER

value: ${NFS\_DOMAIN}

- name: NFS\_PATH

value: ${NFS\_PATH}

volumes:

- name: nfs-client-root

nfs:

server: ${NFS\_DOMAIN}

path: ${NFS\_PATH}

EOF

#### 创建storageclass.yaml文件

cat << EOF > ${NFS\_DEPLOY\_PATH}/class.yaml

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: csi-nfs-sc

provisioner: kubernetes-nfs

parameters:

archiveOnDelete: "false"

EOF

|  |  |  |  |
| --- | --- | --- | --- |
|  | 💡 NOTE | |  |
|  |  | archiveOnDelete： "false" 删除PVC时不会保留数据，"true"将保留PVC数据 |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 🔥 IMPORTANT | |  |
|  |  | provisioner参数 必须与deployment .yaml中的PROVISIONER\_NAME相同 |  |

### 执行NFS StorageClass部署配置

#### 部署NFS StorageClass配置

oc new-project ${NFS\_NAMESPACE}

oc apply -f ${NFS\_DEPLOY\_PATH}/rbac.yaml -n ${NFS\_NAMESPACE}

oc get clusterrole,clusterrolebinding,role,rolebinding -n ${NFS\_NAMESPACE} | grep nfs

clusterrole.rbac.authorization.k8s.io/nfs-client-provisioner-runner 22s

clusterrolebinding.rbac.authorization.k8s.io/run-nfs-client-provisioner 22s

role.rbac.authorization.k8s.io/leader-locking-nfs-client-provisioner 22s

rolebinding.rbac.authorization.k8s.io/leader-locking-nfs-client-provisioner 22s

oc describe scc hostmount-anyuid -n ${NFS\_NAMESPACE}

Name: hostmount-anyuid

Priority: <none>

Access:

Users: system:serviceaccount:openshift-infra:pv-recycler-controller

Groups: <none>

...

为serviceaccount 用户nfs-client-provisioner添加hostmount-anyuid 的scc

oc adm policy add-scc-to-user hostmount-anyuid system:serviceaccount:${NFS\_NAMESPACE}:nfs-client-provisioner

oc describe scc hostmount-anyuid -n ${NFS\_NAMESPACE}

Name: hostmount-anyuid

Priority: <none>

Access:

Users: system:serviceaccount:openshift-infra:pv-recycler-controller,system:serviceaccount:csi-nfs:nfs-client-provisioner

Groups: <none>

...

oc apply -f ${NFS\_DEPLOY\_PATH}/deployment.yaml -n ${NFS\_NAMESPACE}

oc get pod -n ${NFS\_NAMESPACE}

NAME READY STATUS RESTARTS AGE

nfs-client-provisioner-7f79d5787b-lkwdm 1/1 Running 0 5s

oc apply -f ${NFS\_DEPLOY\_PATH}/class.yaml -n ${NFS\_NAMESPACE}

oc get storageclass -n ${NFS\_NAMESPACE}

NAME PROVISIONER AGE

csi-nfs-sc kubernetes-nfs 5s

#### 配置为默认存储类

oc patch storageclass csi-nfs-sc -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "true"}}}' \

-n ${NFS\_NAMESPACE}

oc get storageclass -n ${NFS\_NAMESPACE}

NAME PROVISIONER AGE

csi-nfs-sc (default) kubernetes-nfs 47h

### 部署测试应用验证NFS存储

#### 导入NGINX的应用镜像

setVAR NIGNX\_IMG\_PATH ${OCP\_PATH}/app-image/thirdparty/nginx

skopeo copy --dest-creds=openshift:redhat \

docker-archive:${NIGNX\_IMG\_PATH}/nginx\_1.17.9.tar.gz docker://${REG\_DOMAIN}/apps/nginx:1.17.9

skopeo copy --dest-creds=openshift:redhat \

docker-archive:${NIGNX\_IMG\_PATH}/nginx\_1.17.9.tar.gz docker://${REG\_DOMAIN}/apps/nginx:latest

skopeo inspect --creds=openshift:redhat docker://${REG\_DOMAIN}/apps/nginx:latest

{

"Name": "registry.example.internal:5000/apps/nginx",

"Digest": "sha256:f147d6a97e4924af872168abcce355d468154e139630e7457e45147665f470f1",

"RepoTags": [

"1.17.9",

"latest"

],

"Created": "2020-03-04T17:31:55.614610625Z",

"DockerVersion": "18.09.7",

"Labels": {

"maintainer": "NGINX Docker Maintainers \u003cdocker-maint@nginx.com\u003e"

},

"Architecture": "amd64",

"Os": "linux",

"Layers": [

"sha256:04bb872ddc28cb7ef8e5efd739be9ad2be3777d7b9eaa662c02d549108dab4e3",

"sha256:f16b73f76de49fc7a4c97b344af1aaa7afd6ca0d9adc9fc5ba233ea0284491b7",

"sha256:4e96a6e60281cc14508e873d0e2ae848329c8c622d96c23d19ff6c326cf1b62a"

],

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",

"NGINX\_VERSION=1.17.9",

"NJS\_VERSION=0.3.9",

"PKG\_RELEASE=1~buster"

]

}

#### 部署验证应用

创建验证应用使用的PVC资源

oc new-project pv-demo

cat << EOF | oc apply -f -

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-busybox

labels:

app: busybox

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 5Gi

storageClassName: csi-nfs-sc

EOF

oc get pv,pvc -n pv-demo

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

persistentvolume/pv-registry 50Gi RWX Retain Available 176m

persistentvolume/pvc-a0d63ef3-2584-4cc0-a896-5dea836b7aa9 5Gi RWX Delete Bound pv-demo/pvc-busybox csi-nfs-sc 5m32s

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

persistentvolumeclaim/pvc-busybox Bound pvc-a0d63ef3-2584-4cc0-a896-5dea836b7aa9 5Gi RWX csi-nfs-sc 80m

基于busybox部署backend应用

cat << EOF | oc apply -f -

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend

labels:

app: busybox

tier: backend

spec:

replicas: 2

selector:

matchLabels:

app: busybox

tier: backend

template:

metadata:

labels:

app: busybox

tier: backend

spec:

containers:

- name: pod-backend

image: ${REG\_DOMAIN}/apps/busybox:latest

command: ["/bin/sh"]

args: ["-c", "while true; do date >> /mnt/index.html; hostname >> /mnt/index.html; sleep $(($RANDOM % 5 + 5)); done"]

volumeMounts:

- name: volume-backend

mountPath: /mnt

volumes:

- name: volume-backend

persistentVolumeClaim:

claimName: pvc-busybox

EOF

oc get pod -n pv-demo

NAME READY STATUS RESTARTS AGE

backend-567564874c-nv47z 1/1 Running 0 12s

backend-567564874c-v6pl6 1/1 Running 0 12s

为前端Nginx应用添加访问授权

oc describe scc anyuid

Name: anyuid

Priority: 10

Access:

Users: <none>

Groups: system:cluster-admins

Settings:

Allow Privileged: false

Allow Privilege Escalation: true

Default Add Capabilities: <none>

Required Drop Capabilities: MKNOD

...

oc adm policy add-scc-to-user anyuid system:serviceaccount:pv-demo:default

oc describe scc anyuid

Name: anyuid

Priority: 10

Access:

Users: system:serviceaccount:pvdemo:default

Groups: system:cluster-admins

Settings:

Allow Privileged: false

Allow Privilege Escalation: true

Default Add Capabilities: <none>

Required Drop Capabilities: MKNOD

...

基于NGINX部署frontend应用

cat << EOF | oc create -f -

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend

labels:

app: nginx

tier: frontend

spec:

replicas: 2

selector:

matchLabels:

app: nginx

tier: frontend

template:

metadata:

labels:

app: nginx

tier: frontend

spec:

containers:

- name: pod-frontend

image: ${REG\_DOMAIN}/apps/nginx:latest

ports:

- name: port-frontend

containerPort: 80

volumeMounts:

- name: volume-frontend

mountPath: "/usr/share/nginx/html"

volumes:

- name: volume-frontend

persistentVolumeClaim:

claimName: pvc-busybox

EOF

创建frontend应用的service

cat << EOF | oc create -f -

apiVersion: v1

kind: Service

metadata:

name: svc-frontend

labels:

app: busybox

tier: frontend

spec:

selector:

app: busybox

tier: frontend

ports:

- name: port-frontend

protocol: TCP

port: 80

targetPort: 80

EOF

创建frontend应用的route，然后查看创建结果

cat << EOF | oc create -f -

apiVersion: v1

kind: Route

metadata:

name: route-frontend

labels:

app: busybox

tier: frontend

spec:

to:

kind: Service

name: svc-frontend

EOF

oc get all -n pv-demo

NAME READY STATUS RESTARTS AGE

pod/backend-74b7644746-7sf7c 1/1 Running 0 30m

pod/backend-74b7644746-hjwlf 1/1 Running 0 30m

pod/frontend-79df5488fc-87zhj 1/1 Running 0 69m

pod/frontend-79df5488fc-9ms67 1/1 Running 0 69m

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/svc-frontend ClusterIP 172.30.213.158 <none> 80/TCP 55m

NAME READY UP-TO-DATE AVAILABLE AGE

deployment.apps/backend 2/2 2 2 30m

deployment.apps/frontend 2/2 2 2 69m

NAME DESIRED CURRENT READY AGE

replicaset.apps/backend-74b7644746 2 2 2 30m

replicaset.apps/frontend-79df5488fc 2 2 2 69m

NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

route.route.openshift.io/route-frontend route-frontend-pvdemo.apps.ocp4-1.example.internal svc-frontend <all> None

#### 验证结果

curl -s $(oc get route route-frontend -n pv-demo --template='{{.spec.host}}') | tail -n 4

Thu Mar 12 02:32:21 UTC 2020

backend-74b7644746-hjwlf

Thu Mar 12 02:32:23 UTC 2020

backend-74b7644746-7sf7c

查看nfs服务器上的目录文件

mount -t nfs nfs.example.internal:/data/nfs/userfile /mnt

df /mnt

Filesystem 1K-blocks Used Available Use% Mounted on

nfs.example.internal:/data/nfs/userfile 471625216 132712960 338912256 29% /mnt

ll /mnt

drwxrwxrwx 2 nfsnobody nfsnobody 24 Apr 24 14:25 pvdemo-pvc-busybox-pvc-8bcbd7c3-7e78-48dd-92c9-2b7b02632033

ll /mnt/pvdemo-pvc-busybox-pvc-8bcbd7c3-7e78-48dd-92c9-2b7b02632033/

-rw-r--r-- 1 1000570000 nfsnobody 7560 Apr 24 2020 index.html

umount /mnt

#### 删除应用

删除应用部署

oc delete all --selector app=busybox -n pv-demo

删除PV

oc delete pvc --all -n pv-demo

移除授权

oc adm policy remove-scc-from-user anyuid system:serviceaccount:pv-demo:default

删除项目

oc delete project -n pv-demo

### 附：删除已部署的NFS StorageClass

oc delete -f ${NFS\_DEPLOY\_PATH}/rbac.yaml -n ${NFS\_NAMESPACE}

oc delete -f ${NFS\_DEPLOY\_PATH}/class.yaml -n ${NFS\_NAMESPACE}

oc delete all --selector app=nfs-client-provisioner -n ${NFS\_NAMESPACE}

oc adm policy remove-scc-from-user hostmount-anyuid system:serviceaccount:${NFS\_NAMESPACE}:nfs-client-provisioner

oc delete project -n ${NFS\_NAMESPACE}

## 导入ImageStream并配置Sample Operator

### 上传容器镜像

#### 解压容器镜像包，然后删除

for dir in $(ls --indicator-style=none ${OCP\_PATH}/app-image/redhat-app/images/v2); do

for file in $(ls ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir}/\*.gz); do

tar -xvf ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir}/${file} -C ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir}  
 rm -f ${OCP\_PATH}/app-image/redhat-app/images/v2/${dir}/${file}

done

done

#### 上传容器镜像包

cat ${OCP\_PATH}/app-image/redhat-app/app-images.txt | while read line; do

oc image mirror -a ${REG\_SECRET} ${REG\_DOMAIN}/$(echo $line | cut -d '/' -f 2-) --dir=${OCP\_PATH}/app-image/redhat-app/images \ file://$(echo $line | cut -d '/' -f2)/$(echo $line | cut -d '/' -f3)

done

#### 验证上传结果

查看镜像库目录

curl -s -u openshift:redhat https://${REG\_DOMAIN}/v2/\_catalog | jq .repositories[]

"3scale-amp21/apicast-gateway"

"3scale-amp22/apicast-gateway"

"3scale-amp23/apicast-gateway"

"3scale-amp24/apicast-gateway"

"3scale-amp25/apicast-gateway"

"3scale-amp26/apicast-gateway"

"apps/busybox"

"apps/cicdtools/gogs"

"apps/nginx"

"devtools/go-toolset-rhel7"

"dotnet/dotnet-21-rhel7"

...

验证镜像信息

cat ${OCP\_PATH}/app-image/redhat-app/app-images.txt | while read line; do

echo $(echo $line | cut -d '/' -f3) && oc image info -a ${REG\_SECRET} ${REG\_DOMAIN}/$(echo $line | cut -d '/' -f2-) | \

grep ERROR -A 3

done

cat ${OCP\_PATH}/app-image/redhat-app/app-images.txt | while read line; do

echo $(echo $line | cut -d '/' -f3) && oc image info -a ${REG\_SECRET} ${REG\_DOMAIN}/$(echo $line | cut -d '/' -f2-) | \

head -n 5 && echo ------------------------------------------------------

done

apicast-gateway:1.4-2

Name: registry.example.internal:5000/sample/3scale-amp21/apicast-gateway:1.4-2

Digest: sha256:2942b0b354b6e8504302c6e01e28ed15ebacd7e9479f25a38736d5cbafa55f49

Media Type: application/vnd.docker.distribution.manifest.v2+json

Created: 2y ago

Image Size: 97.96MB in 3 layers

------------------------------------------------------

apicast-gateway:1.8

Name: registry.example.internal:5000/sample/3scale-amp22/apicast-gateway:1.8

Digest: sha256:0d2126523035eaef3bb25732e0c098d8454f305e8cda8859551f039939978cbf

Media Type: application/vnd.docker.distribution.manifest.v2+json

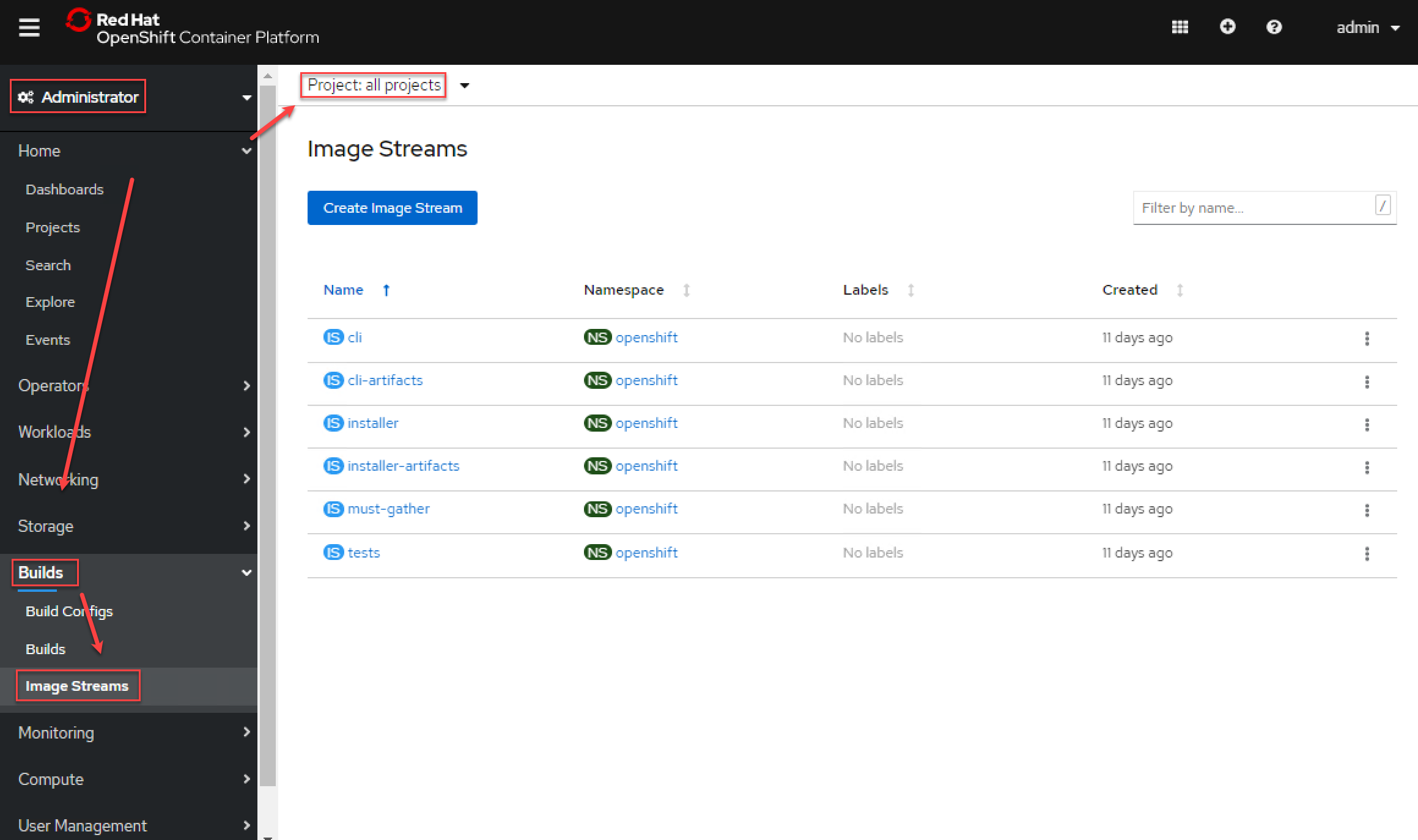
Created: 288d ago

Image Size: 100.9MB in 3 layers

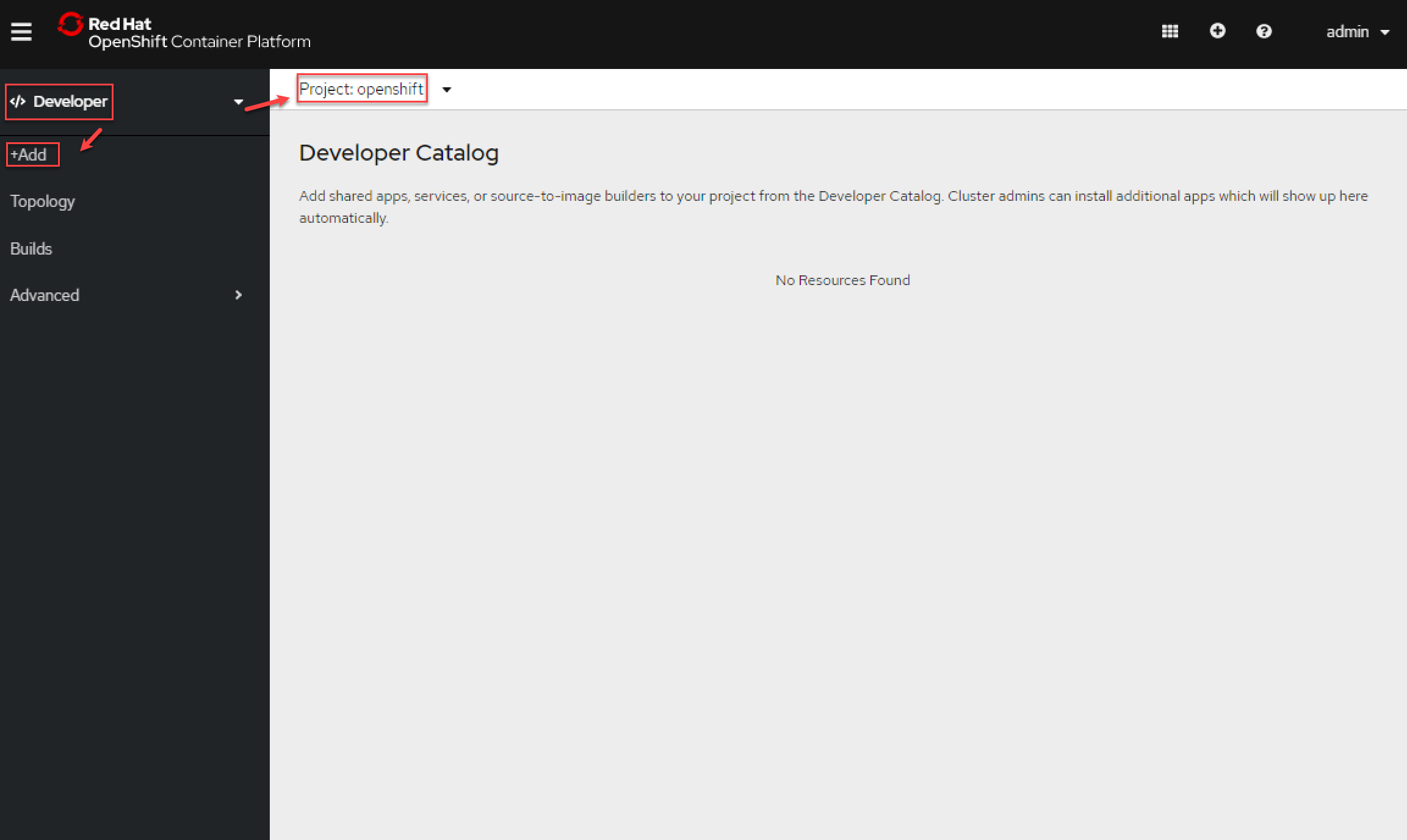
...

### 修改OCP配置

默认情况，系统仅有以下几个imagestream



而由于Sample Operator默认没有启用，因此openshift项目下看不到任何imagestream



私有镜像库使用私有 PKI 签发的证书提供 https 服务, 需要让 samples operator 信任根 CA

oc create configmap registry-crt --from-file=/etc/pki/ca-trust/source/anchors/registry.crt -n openshift-config

oc describe configmap registry-crt -n openshift-config

Name: registry-crt

Namespace: openshift-config

Labels: <none>

Annotations: <none>

Data

====

registry.crt:

----

-----BEGIN CERTIFICATE-----

MIIGAzCCA+ugAwIBAgIJALmEUieT70akMA0GCSqGSIb3DQEBCwUAMIGXMQswCQYD

VQQGEwJDTjERMA8GA1UECAwIU0hBTkdIQUkxETAPBgNVBAcMCFNIQU5HSEFJMQ8w

DQYDVQQKDAZSRURIQVQxCzAJBgNVBAsMAlNBMSIwIAYDVQQDDBlyZWdpc3RyeS5l

eGFtcGxlLmludGVybmFsMSAwHgYJKoZIhvcNAQkBFhFhZG1pbkBleGFtcGxlLmNv

bTAeFw0yMDAzMzAwODU2MDRaFw0yMTAzMzAwODU2MDRaMIGXMQswCQYDVQQGEwJD

TjERMA8GA1UECAwIU0hBTkdIQUkxETAPBgNVBAcMCFNIQU5HSEFJMQ8wDQYDVQQK

DAZSRURIQVQxCzAJBgNVBAsMAlNBMSIwIAYDVQQDDBlyZWdpc3RyeS5leGFtcGxl

LmludGVybmFsMSAwHgYJKoZIhvcNAQkBFhFhZG1pbkBleGFtcGxlLmNvbTCCAiIw

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

-----END CERTIFICATE-----

Events: <none>

oc get image.config.openshift.io -o yaml

apiVersion: v1

items:

- apiVersion: config.openshift.io/v1

kind: Image

metadata:

annotations:

release.openshift.io/create-only: "true"

creationTimestamp: "2020-03-30T10:05:11Z"

generation: 1

name: cluster

resourceVersion: "4025024"

selfLink: /apis/config.openshift.io/v1/images/cluster

uid: 8e8c5aa0-43e7-42c9-b6a3-cd9fbb9fdb27

spec: {}

status:

internalRegistryHostname: image-registry.openshift-image-registry.svc:5000

kind: List

metadata:

resourceVersion: ""

selfLink: ""

oc patch image.config.openshift.io cluster --patch '{"spec":{"additionalTrustedCA":{"name":"registry-crt"}}}' --type=merge

oc get image.config.openshift.io -o yaml

apiVersion: v1

items:

- apiVersion: config.openshift.io/v1

kind: Image

metadata:

annotations:

release.openshift.io/create-only: "true"

creationTimestamp: "2020-03-30T10:05:11Z"

generation: 2

name: cluster

resourceVersion: "5372452"

selfLink: /apis/config.openshift.io/v1/images/cluster

uid: 8e8c5aa0-43e7-42c9-b6a3-cd9fbb9fdb27

spec:

additionalTrustedCA:

name: registry-crt

status:

internalRegistryHostname: image-registry.openshift-image-registry.svc:5000

kind: List

metadata:

resourceVersion: ""

selfLink: ""

清除现有的 imagestreams

oc patch configs.samples.operator.openshift.io cluster \

--patch '[{"op": "replace", "path": "/spec/managementState", "value":"Removed"}]' --type=json

oc patch configs.samples.operator.openshift.io cluster \

--patch '{"spec":{"samplesRegistry":null,"skippedImagestreams":null}}' --type=merge

oc get is -n openshift

NAME IMAGE REPOSITORY TAGS UPDATED

cli image-registry.openshift-image-registry.svc:5000/openshift/cli latest

cli-artifacts image-registry.openshift-image-registry.svc:5000/openshift/cli-artifacts latest

installer image-registry.openshift-image-registry.svc:5000/openshift/installer latest

installer-artifacts image-registry.openshift-image-registry.svc:5000/openshift/installer-artifacts latest

must-gather image-registry.openshift-image-registry.svc:5000/openshift/must-gather latest

tests image-registry.openshift-image-registry.svc:5000/openshift/tests latest

再重新创建出来

oc get pod -n openshift-cluster-samples-operator

NAME READY STATUS RESTARTS AGE

cluster-samples-operator-6b5459b7b9-t67bw 2/2 Running 0 11d

oc patch configs.samples.operator.openshift.io cluster \

--patch '[{"op": "replace", "path": "/spec/managementState", "value":"Managed"}]' --type=json

添加samplesRegistry为私有镜像库地址

oc patch configs.samples.operator.openshift.io cluster --patch '{"spec":{"samplesRegistry":"${REG\_DOMIAN}",\

"skippedImagestreams":["jenkins","jenkins-agent-nodejs","jenkins-agent-maven"]}}' --type=merge

oc get configs.samples.operator.openshift.io -o json |jq -r .items[].spec

{

"architectures": [

"x86\_64"

],

"managementState": "Managed",

"samplesRegistry": "registry.example.internal:5000",

"skippedImagestreams": [

"jenkins",

"jenkins-agent-nodejs",

"jenkins-agent-maven"

]

}

oc get imagestreams -n openshift -o json | jq -r '.items[].metadata.name' | \

grep -v 'cli\|cli-artifacts\|installer\|installer-artifacts\|must-gather\|tests'

apicast-gateway

apicurito-ui

busybox

dotnet

dotnet-runtime

eap-cd-openshift

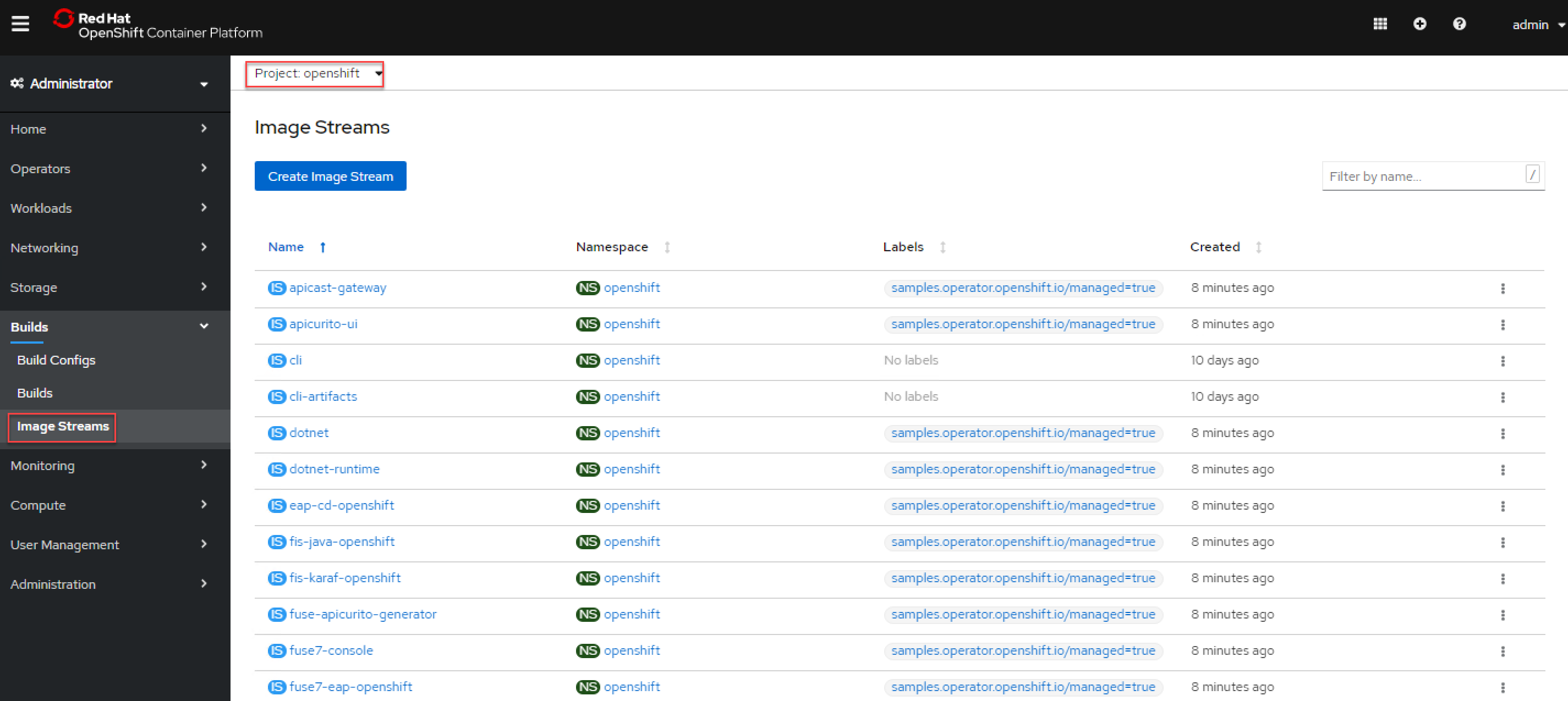
fis-java-openshift

fis-karaf-openshift

fuse-apicurito-generator

fuse7-console

...



oc get template -n openshift -o name

template.template.openshift.io/3scale-gateway

template.template.openshift.io/amq63-basic

template.template.openshift.io/amq63-persistent

template.template.openshift.io/amq63-persistent-ssl

template.template.openshift.io/amq63-ssl

template.template.openshift.io/apicurito

...

稍等几分钟后检查是否获取镜像的摘要信息

oc describe is -n openshift postgresql

oc describe is -n openshift postgresql

Name: postgresql

Namespace: openshift

Created: 7 seconds ago

Labels: samples.operator.openshift.io/managed=true

Annotations: openshift.io/display-name=PostgreSQL

openshift.io/image.dockerRepositoryCheck=2020-04-11T07:21:24Z

samples.operator.openshift.io/version=4.5.9

Image Repository: image-registry.openshift-image-registry.svc:5000/openshift/postgresql

Image Lookup: local=false

Unique Images: 2

Tags: 3

9.6

tagged from registry.example.internal:5000/rhscl/postgresql-96-rhel7:latest

prefer registry pullthrough when referencing this tag

Provides a PostgreSQL 9.6 database on RHEL 7. For more information about using this database image, including OpenShift considerations, see https://github.com/sclorg/postgresql-container/blob/master/README.md.

Tags: database, postgresql

\* registry.example.internal:5000/rhscl/postgresql-96-rhel7@sha256:b47373220099623dd448c2017a876484cfc80857aa9be203f755edc0086973f1

2 seconds ago

10 (latest)

tagged from registry.example.internal:5000/rhscl/postgresql-10-rhel7:latest

prefer registry pullthrough when referencing this tag

Provides a PostgreSQL 10 database on RHEL 7. For more information about using this database image, including OpenShift considerations, see https://github.com/sclorg/postgresql-container/blob/master/README.md.

Tags: database, postgresql

\* registry.example.internal:5000/rhscl/postgresql-10-rhel7@sha256:f2d87b72362cb30e60bbfbf2f65999c657c272de4ec0d84321ca23bca35fafe2

2 seconds ago

## 部署Gogs+PostgreSQL环境

### 准备Gogs镜像

[Bastion]

[root@bastion ~]#

1. 获得Gogs镜像到离线文件

mkdir -p ~/cicdtools && cd ~/cicdtools

**方法1**：在线下载Gogo镜像

skopeo copy docker://docker.io/wkulhanek/gogs:11.86 docker-archive:cicdtools/gogs\_11.86.tar.gz

**方法2**：从本教程《获取离线安装文件》中的网盘中获得Gogo镜像文件：gogs\_11.86.tar.gz

1. 向本地Docker Registry导入Gogs镜像

skopeo copy --dest-creds=openshift:redhat \

docker-archive:gogs\_11.86.tar.gz docker://registry.example.internal:5000/apps/cicdtools/gogs:11.86

### 使用Template结合StorageClass部署PostgreSQL数据库

oc new-project cicd-tool-chain

oc new-app -l app=gogs-db --template=postgresql-persistent \

--param DATABASE\_SERVICE\_NAME=gogs-db \

--param POSTGRESQL\_DATABASE=gogs \

--param POSTGRESQL\_USER=gogs \

--param POSTGRESQL\_PASSWORD=gogs \

--param VOLUME\_CAPACITY=4Gi \

### 部署Gogs应用

oc project cicd-tool-chain

oc create secret docker-registry secret-private-registry \

--docker-server=registry.example.internal:5000 --docker-username=openshift --docker-password=redhat --docker-email=unused

oc secrets link default secret-private-registry --for=pull

oc new-app --docker-image=registry.example.internal:5000/apps/cicdtools/gogs:11.86 -lapp=gogs --name=gogs

oc rollout pause dc gogs

oc patch dc gogs --patch='{ "spec": { "strategy": { "type": "Recreate" }}}'

oc set volume dc/gogs --add --overwrite --name=gogs-volume-1 --mount-path=/data/ \

--type persistentVolumeClaim --claim-name=gogs --claim-size=10Gi

oc rollout resume dc gogs

oc expose svc gogs

### Gogs应用初始化

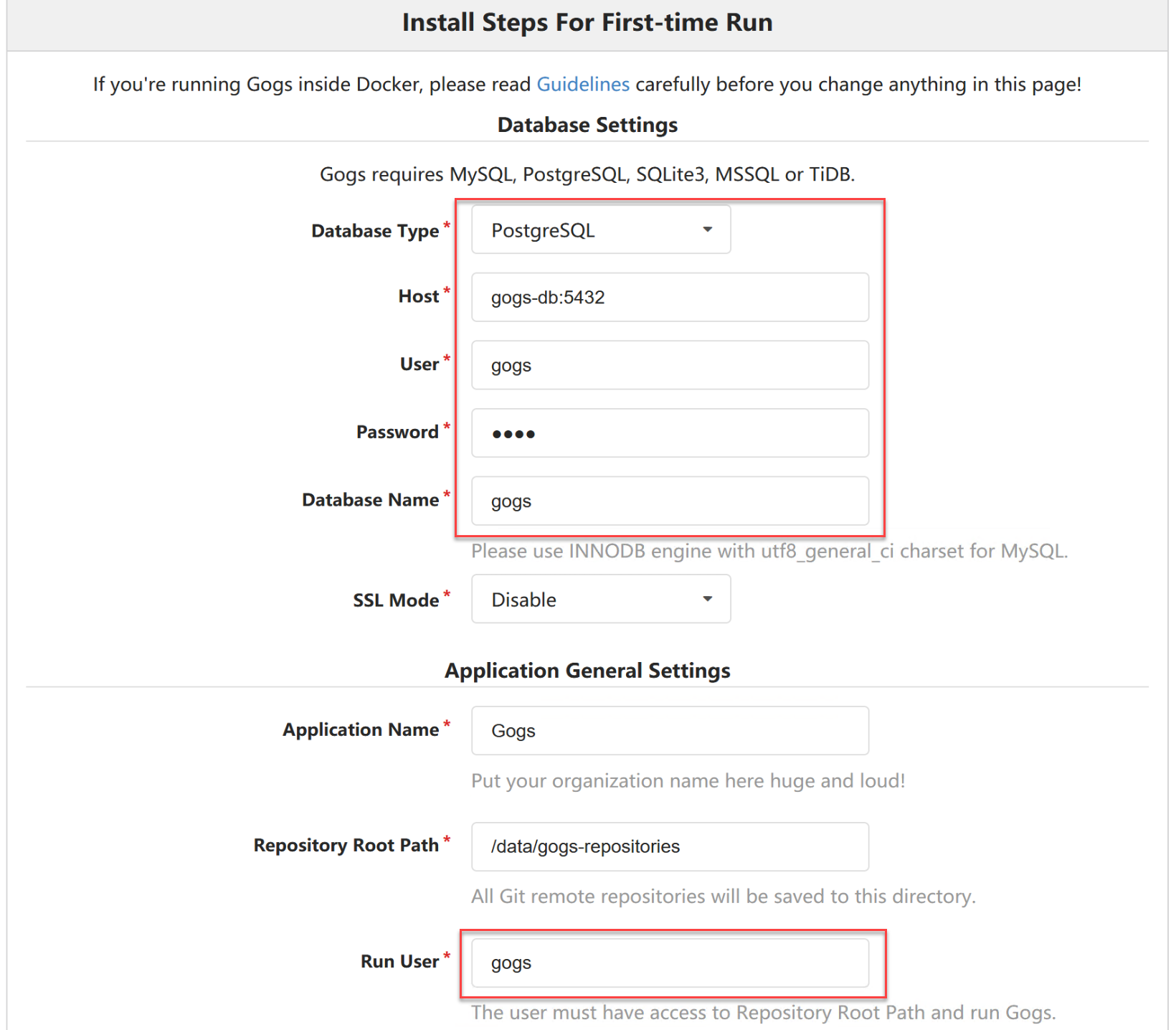
获取Gogs的访问地址

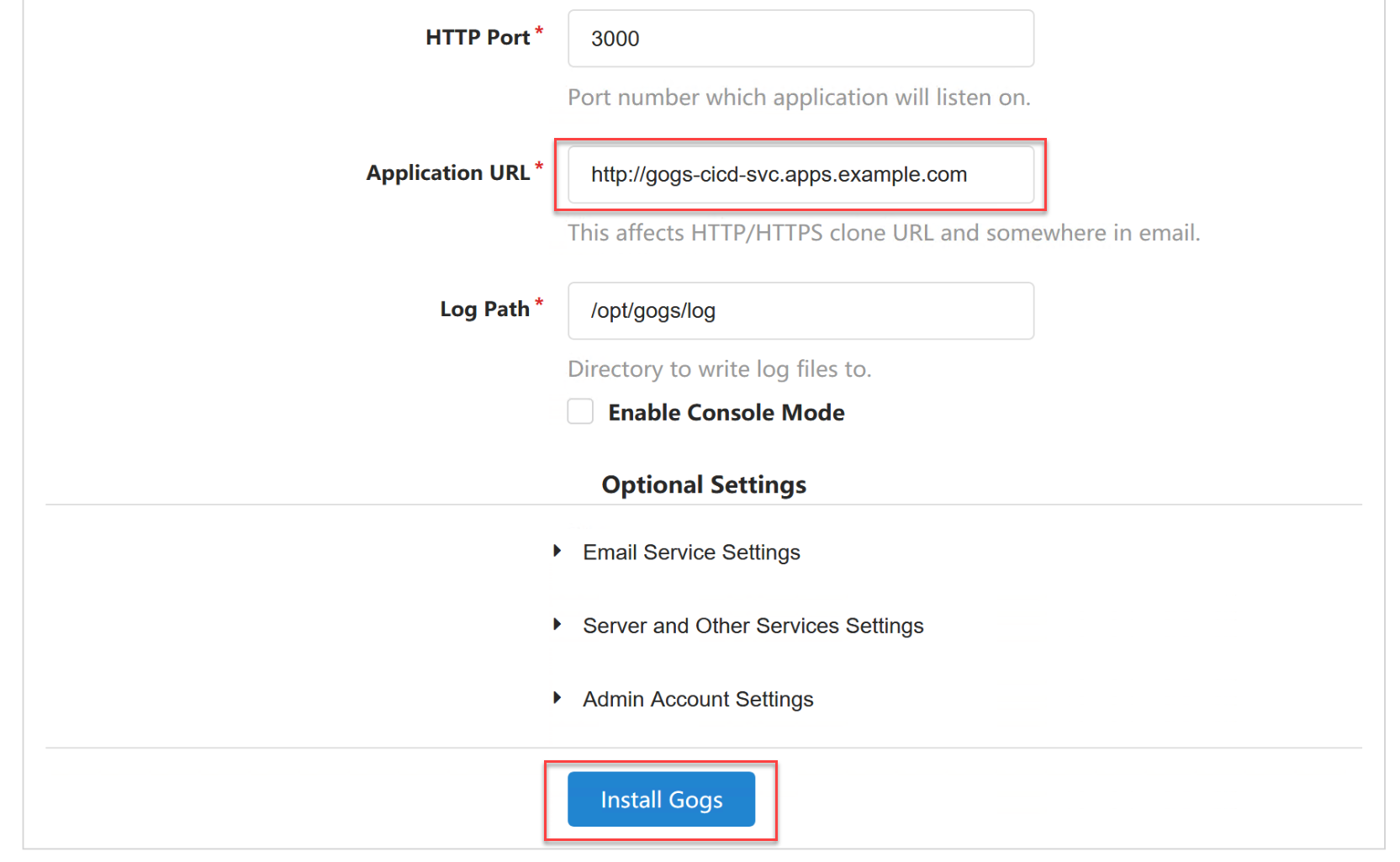
oc get route gogs --template '{{.spec.host}}'

gogs-cicd-tool-chain.apps.ocp4-1.example.internal

用浏览器访问http://gogs-cicd-tool-chain.apps.ocp4-1.example.internal，然后在Gogs中进行如下配置：

1. Database Type: PostgreSQL
2. Host: gogs-db:5432
3. User: gogs
4. Password: gogs
5. Database Name: gogs
6. Run User: gogs
7. Application URL: http://gogs-cicd-tool-chain.apps.ocp4-1.example.internal
8. 点击Install Gogs按钮





进入运行PostgreSQL数据库的Pod

sh-4.2$

以gogs用户登陆，然后列出所有数据库

psql -U gogs

psql (10.12)

Type "help" for help.

gogs-> \l

List of databases

Name | Owner | Encoding | Collate | Ctype | Access privileges

-----------+----------+----------+------------+------------+-----------------------

gogs | gogs | UTF8 | en\_US.utf8 | en\_US.utf8 |

postgres | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 |

template0 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | en\_US.utf8 | en\_US.utf8 | =c/postgres +

| | | | | postgres=CTc/postgres

(4 rows)

选择数据库gogs，然后查看数据库中的所有表

gogs-> \c gogs

You are now connected to database "gogs" as user "gogs".

gogs-> \dt

List of relations

Schema | Name | Type | Owner

--------+--------------------------+-------+-------

public | access | table | gogs

public | access\_token | table | gogs

public | action | table | gogs

public | attachment | table | gogs

public | collaboration | table | gogs

public | comment | table | gogs

public | deploy\_key | table | gogs

public | email\_address | table | gogs

public | follow | table | gogs

public | hook\_task | table | gogs

public | issue | table | gogs

public | issue\_label | table | gogs

public | issue\_user | table | gogs

public | label | table | gogs

public | login\_source | table | gogs

public | milestone | table | gogs

public | mirror | table | gogs

public | notice | table | gogs

public | org\_user | table | gogs

public | protect\_branch | table | gogs

public | protect\_branch\_whitelist | table | gogs

public | public\_key | table | gogs

public | pull\_request | table | gogs

public | release | table | gogs

public | repository | table | gogs

public | star | table | gogs

public | team | table | gogs

public | team\_repo | table | gogs

public | team\_user | table | gogs

public | two\_factor | table | gogs

public | two\_factor\_recovery\_code | table | gogs

public | upload | table | gogs

public | user | table | gogs

public | version | table | gogs

public | watch | table | gogs

public | webhook | table | gogs

(36 rows)

### Gogs配置持久化

通过获取当前正在运行的Gogs的配置文件app.ini（/opt/gogs/custom/conf/app.ini），并通过Configmap持久化配置，确保Gogs容器重启后，原来的配置依赖有效。

oc project cicd-tool-chain

获取Gogs的配置文件app.ini

oc cp $(oc get pod -l app=gogs -o custom-columns=NAME:.metadata.name --no-headers):opt/gogs/custom/conf/app.ini ~/app.ini

添加参数到app.ini中文件

[webhook]

SKIP\_TLS\_VERIFY = true

通过app.ini配置文件生成对应的Configmap，查看Configmap的内容同app.ini配置文件内容一致

oc create configmap conf-gogs --from-file=~/app.ini

oc describe configmap conf-gogs

Name: conf-gogs

Namespace: cicd-svc

Labels: <none>

Annotations: <none>

Data

====

app.ini:

----

APP\_NAME = Gogs

RUN\_USER = gogs

RUN\_MODE = prod

[database]

DB\_TYPE = postgres

HOST = gogs-db:5432

NAME = gogs

USER = gogs

PASSWD = gogs

SSL\_MODE = disable

PATH = data/gogs.db

<<OMIT>>

将Configmap配置挂载到容器的DC配置上，使配置持久化，然后查看Configmap挂载结果。

oc set volume dc/gogs --add --overwrite --name=config-volume -m /opt/gogs/custom/conf/ -t configmap --configmap-name=conf-gogs

oc set volume dc/gogs

deploymentconfigs/gogs

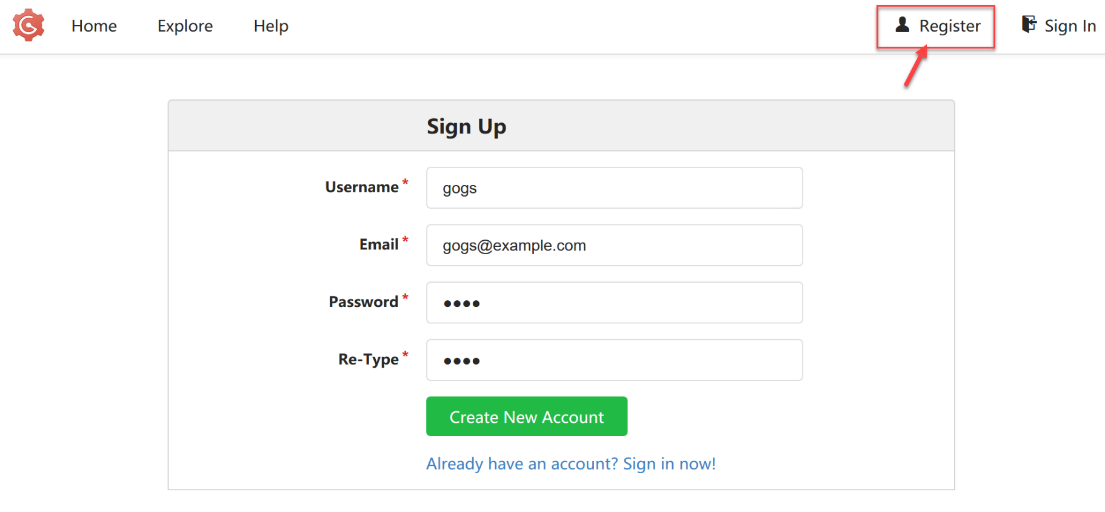
pvc/pvc-gogs (allocated 10GiB) as gogs-volume-1

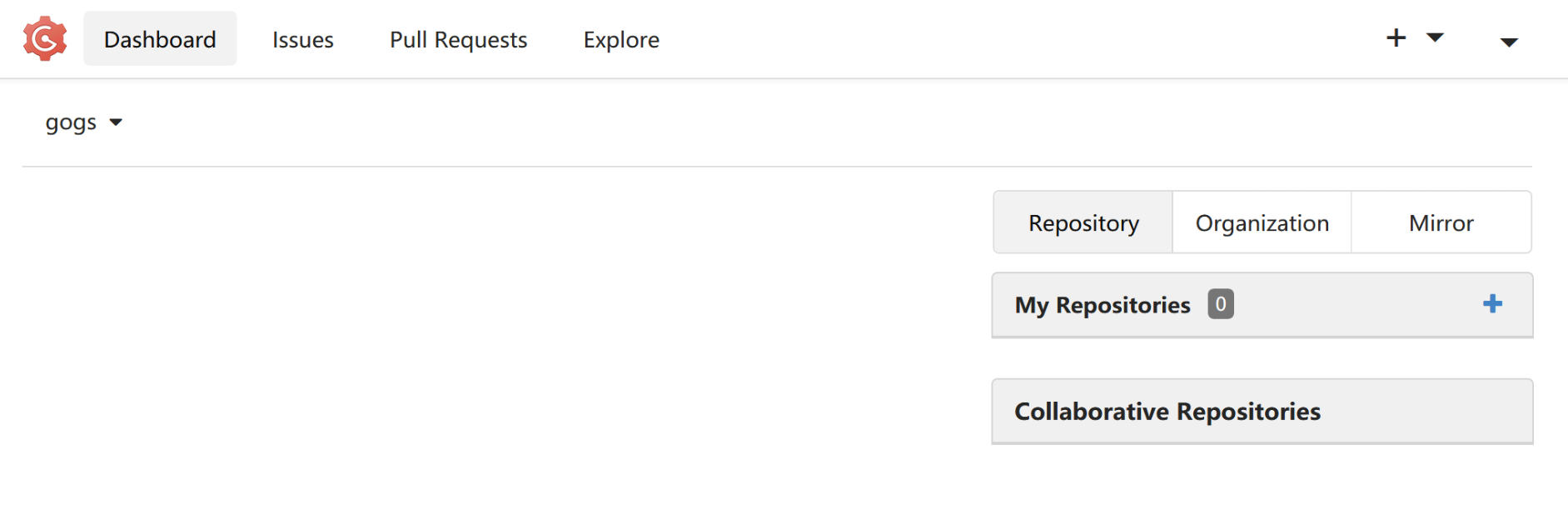
mounted at /data

configMap/conf-gogs as config-volume

mounted at /opt/gogs/custom/conf

等待容器重建完毕，然后再次访问gogs页面。创建新用户，默认情况，第一个创建用户为管理员。本示例中，用户名和密码均为gogs。注册完毕后，使用相应的用户名和密码登陆Gogs即可。



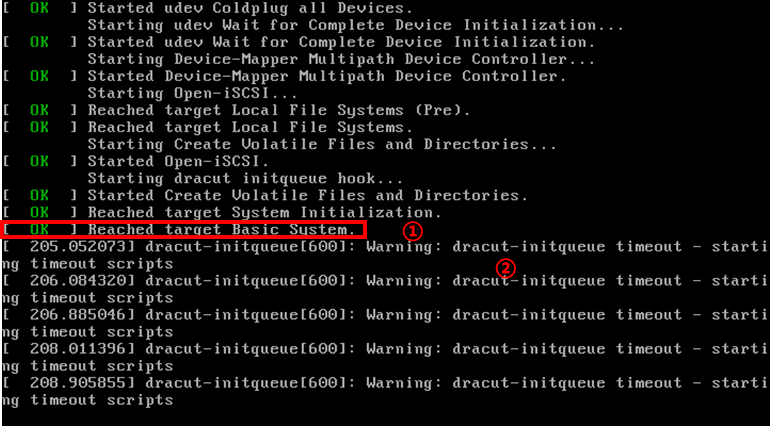


# 安装常见错误

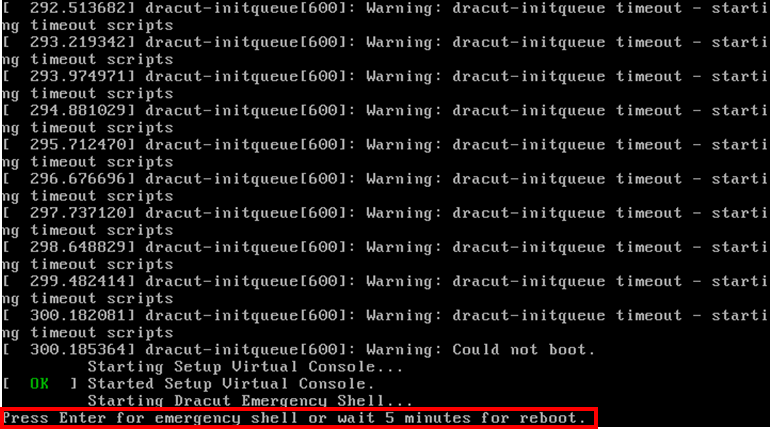
## bootstrap网卡名错误导致启动失败

### 现象

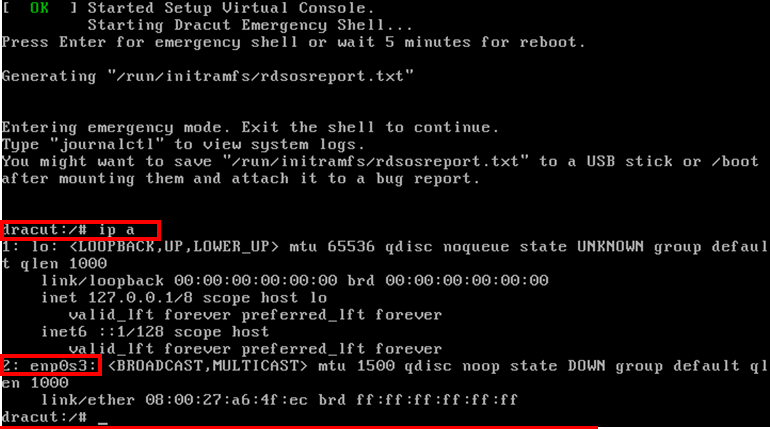
1. 在bootstrap启动后停留在下图中的**①**处约5分钟后，然后反复出现**②**的错误。



1. 然后界面显示下图提示“Could not boot”，说明bootstrap没有启动成功。此时按入回车键。



1. 界面显示如下图。然后通过下图中的命令查看bootstrap的网络接口名称，例如下图中的“enp0s3”。



### 确认

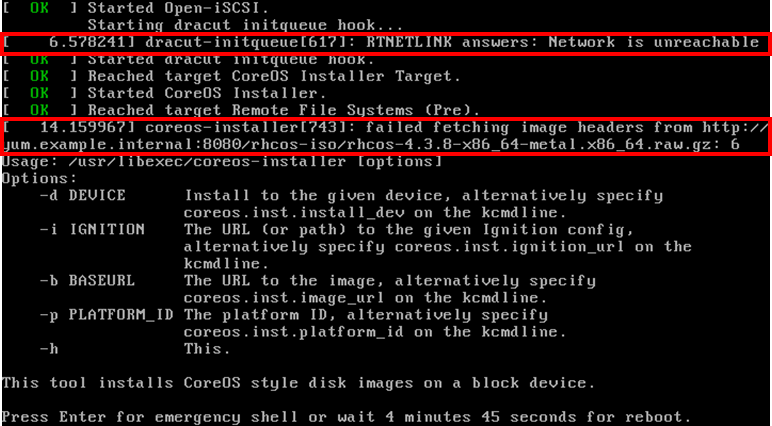
1. 确认bootstrap节点使用的网络接口名和support节点中的“NET\_IF\_NAME”环境变量设置的名称是一致的。如果两者不一致那么需要修复错误。

### 修复错误

1. 将support节点的“NET\_IF\_NAME”环境变量设置为上图中实际的网络接口名称，例如“enp0s3”。
2. 重新生成所有ISO文件，并使用新的ISO文件启动虚拟机。

## 无法下载文件

### 现象



### 确认

1. 确认support节点的防火墙已经关闭。
2. 在support节点执行以下命令，确认可下载该文件。

curl -I -s http://yum.example.internal:8080/rhcos-iso/rhcos-4.5.6-x86\_64-metal.x86\_64.raw.gz

1. 在bastion节点或其他节点执行以下命令，确认可下载该文件。

curl -I -s http://yum.example.internal:8080/rhcos-iso/rhcos-4.5.6-x86\_64-metal.x86\_64.raw.gz

## 无法证书过期

### 现象

在安装日志中有以下提示：证书过期或无效。

DEBUG Still waiting for the Kubernetes API: Get https://api.ocp4-1.example.internal:6443/version?timeout=32s: x509: certificate has expired or is not yet valid

### 修复错误

有以下两种情况可以导致安装过程中的证书过期错误：

1. 在生成证书后24小时还没有安装完OpenShift集群，则证书过期。

**解决方案**：重新执行“5.4章节”以后的操作，生成新证书和所有相关文件。然后重新创建现有OpenShift集群虚拟机的存储，并重新执行OpenShift集群安装过程。

1. 生成证书的节点和OpenShift集群中节点的系统时间差别较大，导致证书无效。

**解决方案**：修改生成证书的support节点的系统时间，使之保持和OpenShift集群的bootstrap、master节点时间相同。然后重新执行OpenShift集群安装过程。