

OpenShift 4.3

基于在线裸机环境  
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 | 066-4OL | 2020/07/17 | 刘晓宇 | 针对BM在线安装优化 |

Contents

[1 OpenShift集群说明和部署规划 1](#_Toc45919946)

[1.1 重要声明 1](#_Toc45919947)

[1.2 本文的OpenShift集群安装方式 1](#_Toc45919948)

[1.3 OpenShift集群部署过程和节点角色说明 1](#_Toc45919949)

[1.4 集群节点角色 1](#_Toc45919950)

[1.5 虚机节点资源配置规划 2](#_Toc45919951)

[1.6 虚机节点网络配置规划 2](#_Toc45919952)

[1.7 DNS域名解析规划 2](#_Toc45919953)

[2 创建Support虚机节点 3](#_Toc45919954)

[2.1 安装Support虚机节点操作系统 3](#_Toc45919955)

[2.2 配置在线YUM源 4](#_Toc45919956)

[2.3 安装工具软件 4](#_Toc45919957)

[3 获取安装文件 4](#_Toc45919958)

[3.1 确定安装版本和创建目录 5](#_Toc45919959)

[3.2 获得安装文件方法1 6](#_Toc45919960)

[3.3 获得安装文件方法2 7](#_Toc45919961)

[3.3.1 下载并安装OCP客户端 7](#_Toc45919962)

[3.3.2 下载openshift-install 7](#_Toc45919963)

[3.3.3 下载CoreOS镜像 7](#_Toc45919964)

[4 配置Support节点的服务 7](#_Toc45919965)

[4.1 设置防火墙和SELinux 7](#_Toc45919966)

[4.1.1 关闭firewalld防火墙 7](#_Toc45919967)

[4.1.2 设置selinux 8](#_Toc45919968)

[4.2 设置环境变量 8](#_Toc45919969)

[4.3 安装HTTP服务 8](#_Toc45919970)

[4.4 安装配置DNS服务 8](#_Toc45919971)

[4.4.1 安装BIND服务 9](#_Toc45919972)

[4.4.2 设置BIND配置文件 9](#_Toc45919973)

[4.4.3 配置Zone区域 9](#_Toc45919974)

[4.4.4 重启BIND服务 12](#_Toc45919975)

[4.4.5 测试正反向DNS解析 12](#_Toc45919976)

[4.5 部署NTP服务 13](#_Toc45919977)

[4.5.1 设置正确的时区 13](#_Toc45919978)

[4.5.2 安装chrony 13](#_Toc45919979)

[4.5.3 配置chrony 13](#_Toc45919980)

[4.5.4 检查chrony服务端启动 14](#_Toc45919981)

[4.5.5 验证chrony服务同步 15](#_Toc45919982)

[4.6 部署HAProxy负载均衡服务 15](#_Toc45919983)

[5 定制化安装文件 18](#_Toc45919984)

[5.1 设置基础环境变量 18](#_Toc45919985)

[5.2 安装oc客户端 18](#_Toc45919986)

[5.3 准备CoreOS启动文件及其HTTP访问服务 18](#_Toc45919987)

[5.3.1 准备CoreOS启动文件的HTTP访问服务 18](#_Toc45919988)

[5.3.2 准备CoreOS Metal Raw文件 19](#_Toc45919989)

[5.3.3 准备CoreOS ISO文件 19](#_Toc45919990)

[5.3.4 验证可访问到CorsOS的ISO和RWS启动文件 21](#_Toc45919991)

[5.4 准备Ignition引导文件 21](#_Toc45919992)

[5.4.1 安装openshift-install 21](#_Toc45919993)

[5.4.2 准备install-config.yaml文件 21](#_Toc45919994)

[5.4.3 创建manifest文件 24](#_Toc45919995)

[5.4.4 创建Ignition引导文件 25](#_Toc45919996)

[5.4.5 创建Ignition引导文件下载目录 25](#_Toc45919997)

[6 创建Bootstrap、Master、Worker虚拟机节点 25](#_Toc45919998)

[7 安装OCP集群 26](#_Toc45919999)

[7.1 第一阶段：部署bootstrap阶段 26](#_Toc45920000)

[7.2 第二阶段：部署master阶段 27](#_Toc45920001)

[7.3 第三阶段：配置master阶段 33](#_Toc45920002)

[7.4 第四阶段：部署worker阶段 34](#_Toc45920003)

[8 集群初始化和功能验证 36](#_Toc45920004)

[8.1 用户管理 36](#_Toc45920005)

[8.1.1 新建集群管理员 36](#_Toc45920006)

[8.1.2 新建普通用户 37](#_Toc45920007)

[8.1.3 删除kubeadmin 37](#_Toc45920008)

[8.2 部署BusyBox应用 38](#_Toc45920009)

[8.2.1 导入BusyBox的应用镜像 38](#_Toc45920010)

[8.2.2 部署BusyBox应用 38](#_Toc45920011)

[8.3 部署NFS服务 39](#_Toc45920012)

[8.3.1 安装NFS服务 39](#_Toc45920013)

[8.3.2 验证NFS共享目录 39](#_Toc45920014)

[8.4 配置OpenShift内部镜像库的存储 39](#_Toc45920015)

[8.4.1 创建内部镜像库使用的NFS目录 39](#_Toc45920016)

[8.4.2 创建PV 40](#_Toc45920017)

[8.4.3 创建PVC 40](#_Toc45920018)

[8.4.4 指定内部镜像库使用PVC 40](#_Toc45920019)

[8.5 配置NFS StorageClass 41](#_Toc45920020)

[8.5.1 创建NFS 目录 41](#_Toc45920021)

[8.5.2 创建NFS StorageClass部署配置 41](#_Toc45920022)

[8.5.3 执行NFS StorageClass部署配置 44](#_Toc45920023)

[8.5.4 部署测试应用验证NFS存储 45](#_Toc45920024)

[8.5.5 附：删除已部署的NFS StorageClass 48](#_Toc45920025)

[9 安装常见错误 49](#_Toc45920026)

[9.1 bootstrap网卡名错误导致启动失败 49](#_Toc45920027)

[9.1.1 现象 49](#_Toc45920028)

[9.1.2 确认 50](#_Toc45920029)

[9.1.3 修复错误 50](#_Toc45920030)

[9.2 无法下载文件 50](#_Toc45920031)

[9.2.1 现象 50](#_Toc45920032)

[9.2.1 确认 50](#_Toc45920033)

[9.3 无法证书过期 51](#_Toc45920034)

[9.3.1 现象 51](#_Toc45920035)

[9.3.1 修复错误 51](#_Toc45920036)

# OpenShift集群说明和部署规划

## 重要声明

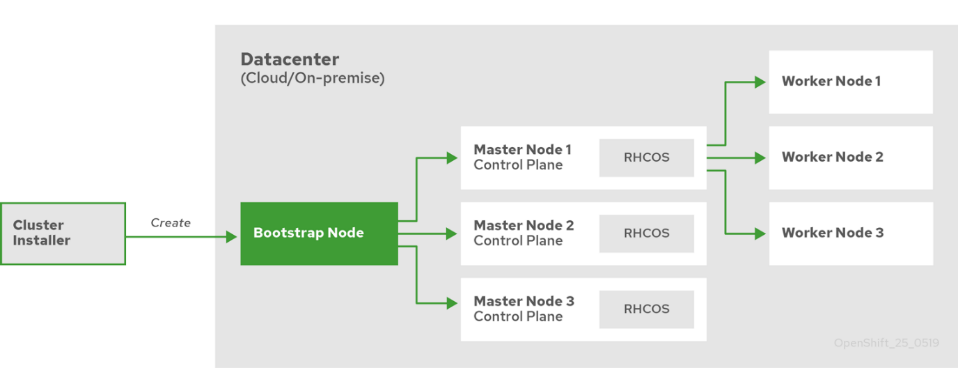
1. 本文内容只适用于在线安装OpenShift 4.3的通用测试环境。如果安装生产环境，请根据本文和实际环境定制安装。

## 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的离线安装方式部署一个精简的OpenShift 4.3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。

## 集群节点角色

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | 节点类型 | 功能角色 | 本文标识 |
| bootstrap | VM+CoreOS | 安装初始化，当安装完master节点后就可以关闭bootstrap | [Bootstrap] |
| support | VM+RHEL | DNS 服务  NTP服务  NFS Storage服务  HAProxy Load Balance  Cluster Installer（运行安装操作） | [Support] |
| master-0 | VM+CoreOS | OCP Cluster-Master  OCP Cluster -Etcd | [Master] |
| worker-0 | VM+CoreOS | OCP Cluster -Worker Node | [Worker] |
| worker-1 | VM+CoreOS | OCP Cluster -Worker Node |

## 虚机节点资源配置规划

以下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。

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 |
| support | RHEL 7.6/CentOS7.6 | 2 | 4 | 200 |  |
| bootstrap | RHCOS | 2 | 10 | 100 |  |
| master-0 | RHCOS | 4 | 16 | 200 |  |
| worker-0 | RHCOS | 4 | 16-32 | 200 |  |
| worker-1 | RHCOS | 4 | 16-32 | 200 |  |

## 虚机节点网络配置规划

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hostname | IP Address | Gateway | NETMASK | DNS |
| support | 10.101.31.159 | 10.101.31.1 | 255.255.255.0 | 10.101.31.159 |
| bootstrap | XXXXXXXX |
| master-0 | 10.101.31.160 |
| worker-0 | 10.101.31.161 |
| worker-1 | 10.101.31.162 |

## DNS域名解析规划

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

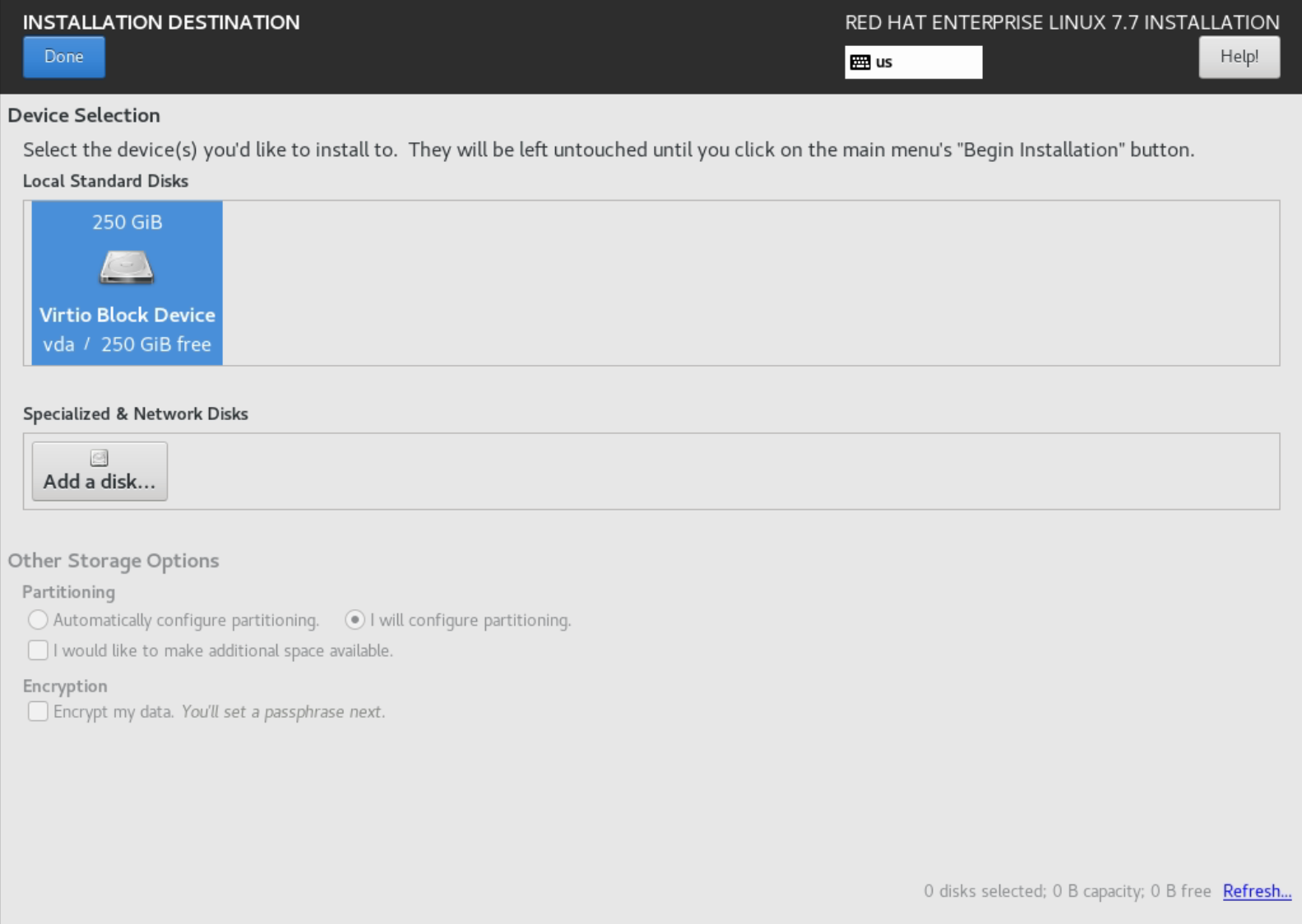
|  |  |
| --- | --- |
| DNS Name | IP |
| support.example.internal  dns.example.internal  nfs.example.internal  yum.example.internal  registry.example.internal  nfs.example. internal  lb.ocp4-1.example.internal  api.ocp4-1.example.internal  api-int.ocp4-1.example.internal  \*.apps.ocp4-1.example.internal | 10.101.31.159 |
| bootstrap.ocp4-1.example.internal |  |
| master-0.ocp4-1.example.internal | 10.101.31.160 |
| worker-0.ocp4-1.example.internal | 10.101.31.161 |
| worker-1.ocp4-1.example.internal | 10.101.31.162 |

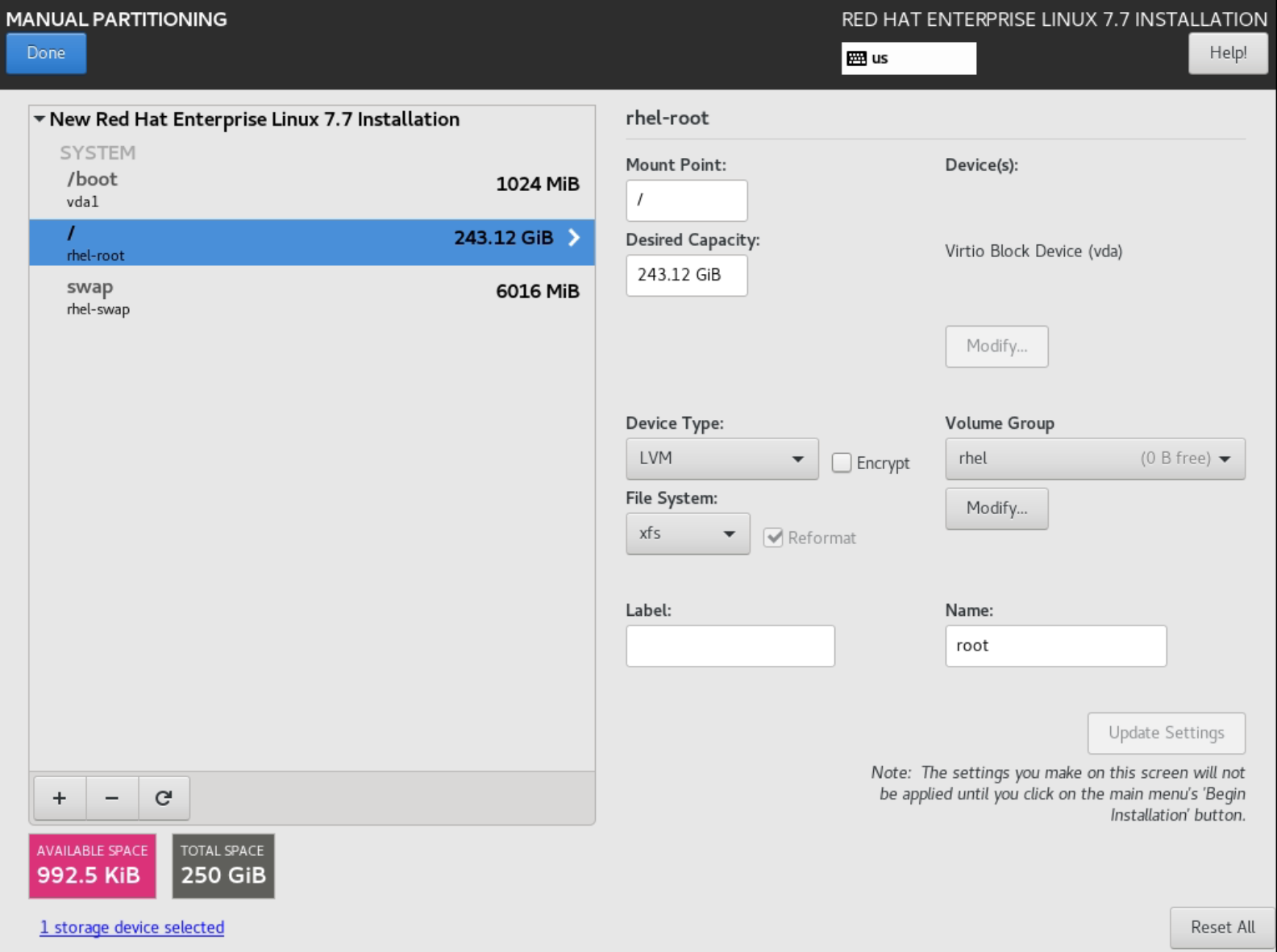
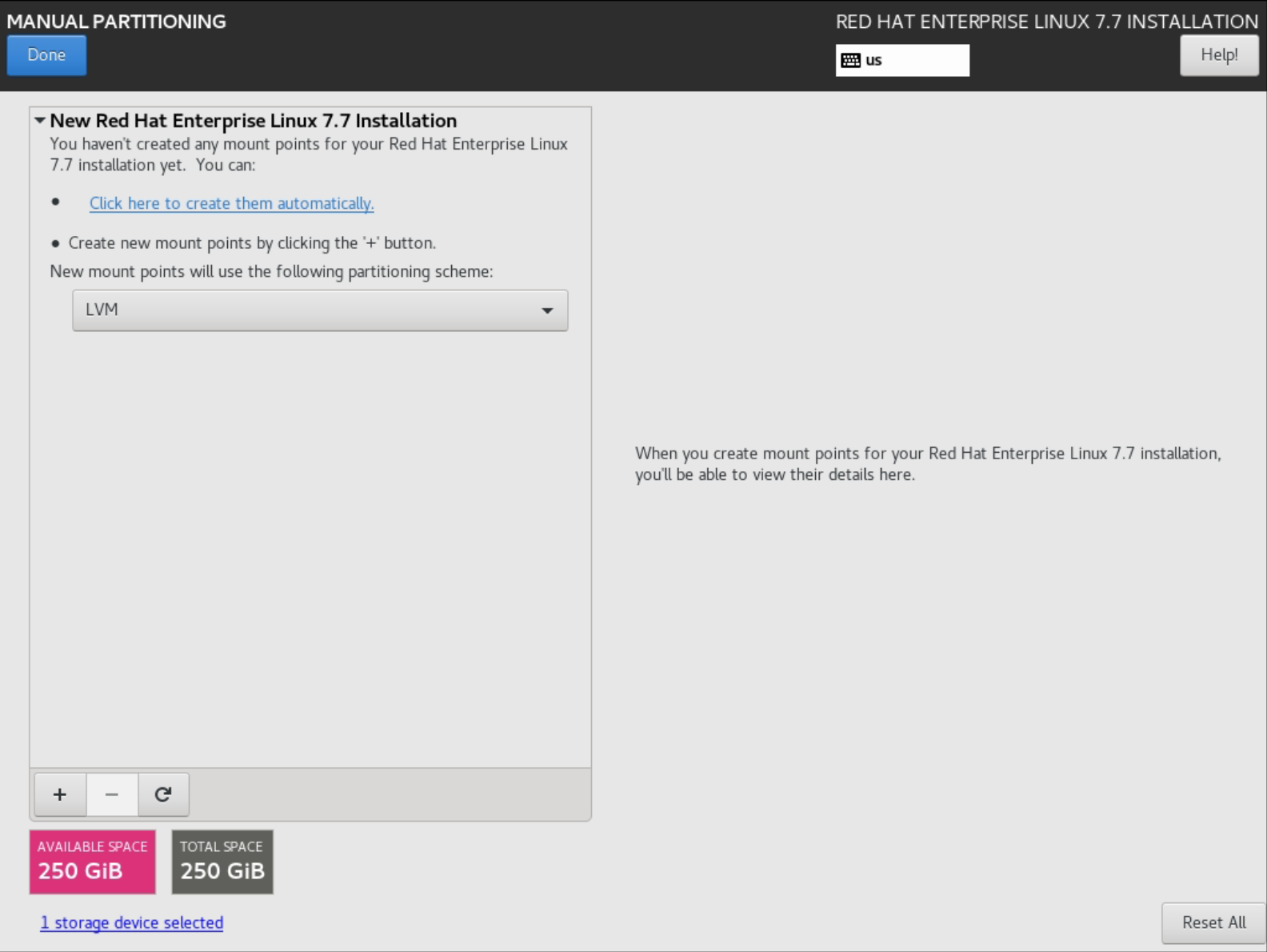
# 创建Support虚机节点

## 安装Support虚机节点操作系统

按以下要求创建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Support虚拟机。
2. 最小化安装RHEL7.6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”





## 配置在线YUM源

## 安装工具软件

yum -y install wget git tree openssl curl vim jq

# 获取安装文件

**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 ~]#

## 确定安装版本和创建目录

如果打算采用以下方法1获得安装文件，请执行：

export OCP\_VER=4.3.18

如果打算采用以下方法2获得安装文件，请执行：

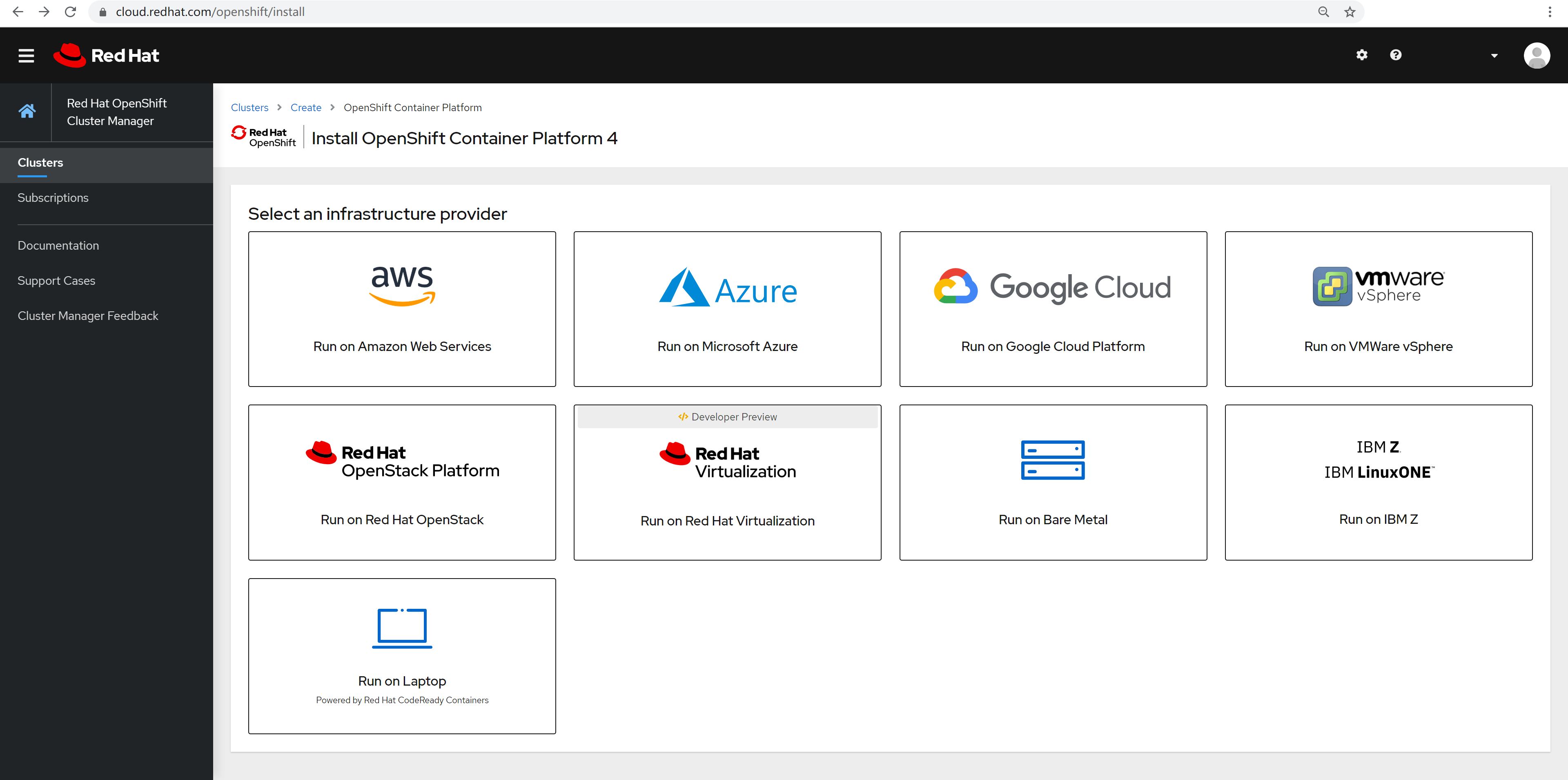
export OCP\_VER=$(curl -s https://mirror.openshift.com/pub/openshift-v4/clients/ocp/stable-4.3/release.txt | \grep 'Name:' | awk '{print $NF}')

创建存放安装文件的目录（**说明**：本文使用的是4.3.18版本）

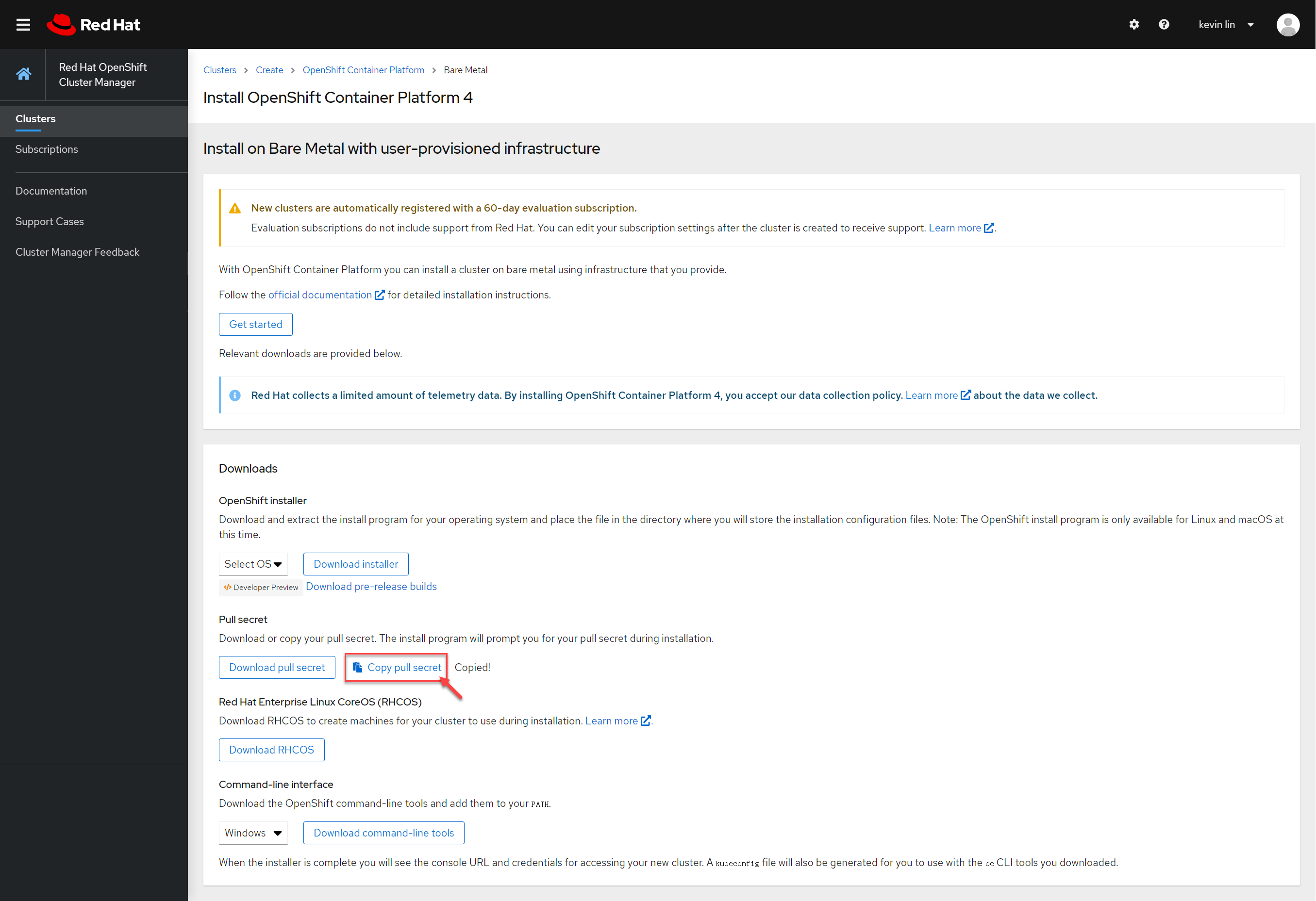
export OCP\_PATH=/data/OCP-${OCP\_VER}/ocp

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

使用订阅账号登陆如下网站 https://cloud.redhat.com/openshift/install



然后进入Bare Metal，将下图中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"

}

}

}

## 获得安装文件方法1

请下载网盘OCP-4.3.18/ocp目录下的csi、ocp-client, ocp-installer, rhcos四个目录到support节点的对应目录即可。**注意**：网盘中是基于OpenShift 4.3.18的介质。

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

## 获得安装文件方法2

### 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: 4.3.18

### 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-4.3.18.tar.gz

### 下载CoreOS镜像

#### 获取CoreOS 4.3最新版本信息

RHCOS\_VER=$(curl -s https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.3/latest/sha256sum.txt | \

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

echo ${RHCOS\_VER}

4.3.18

#### 下载CoreOS 4.3镜像文件

mkdir -p ${OCP\_PATH}/rhcos

wget https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.3/${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/4.3/${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 ~]#

## 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

## 安装HTTP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

chmod -R 705 /data

## 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 SUPPORT\_IP 10.101.31.159

setVAR DNS\_IP 10.101.31.159

setVAR NTP\_IP 10.101.31.159

setVAR YUM\_IP 10.101.31.159

setVAR REGISTRY\_IP 10.101.31.159

setVAR NFS\_IP 10.101.31.159

setVAR LB\_IP 10.101.31.159

setVAR BOOTSTRAP\_IP XXXXXXXXX

setVAR MASTER0\_IP 10.101.31.160

setVAR WORKER0\_IP 10.101.31.161

setVAR WORKER1\_IP 10.101.31.162

#### 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 |
| 101.10.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 "101.10.in-addr.arpa" IN {

type master;

file "159.31.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}.

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.

support IN A 10.101.31.159

dns IN A 10.101.31.159

ntp IN A 10.101.31.159

yum IN A 10.101.31.159

registry IN A 10.101.31.159

nfs IN A 10.101.31.159

#### 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 10.101.31.159

lb IN A 10.101.31.159

api IN A 10.101.31.159

api-int IN A 10.101.31.159

\*.apps IN A 10.101.31.159

bootstrap IN A XXXXXXXXX

master-0 IN A 10.101.31.160

etcd-0 IN A 10.101.31.160

worker-0 IN A 10.101.31.161

worker-1 IN A 10.101.31.162

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

#### 创建159.31.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/159.31.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}.

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

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

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

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

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

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

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

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

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

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

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

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

162.31.101.10.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.

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

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

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

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

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

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

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

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

159.31.101.10.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

10.101.31.159

dig yum.${DOMAIN} +short

10.101.31.159

dig registry.${DOMAIN} +short

10.101.31.159

dig ntp.${DOMAIN} +short

10.101.31.159

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

10.101.31.159

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

10.101.31.159

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

10.101.31.159

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

XXXXXXXXX

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

10.101.31.160

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

10.101.31.160

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

10.101.31.161

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

10.101.31.162

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

10.101.31.159

1. 反向解析测试

dig -x 10.101.31.159 +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 XXXXXXXXX +short

bootstrap.ocp4-1.example.internal.

dig -x 10.101.31.160 +short

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

dig -x 10.101.31.161 +short

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

dig -x 10.101.31.162 +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 10.101.0.0\/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))

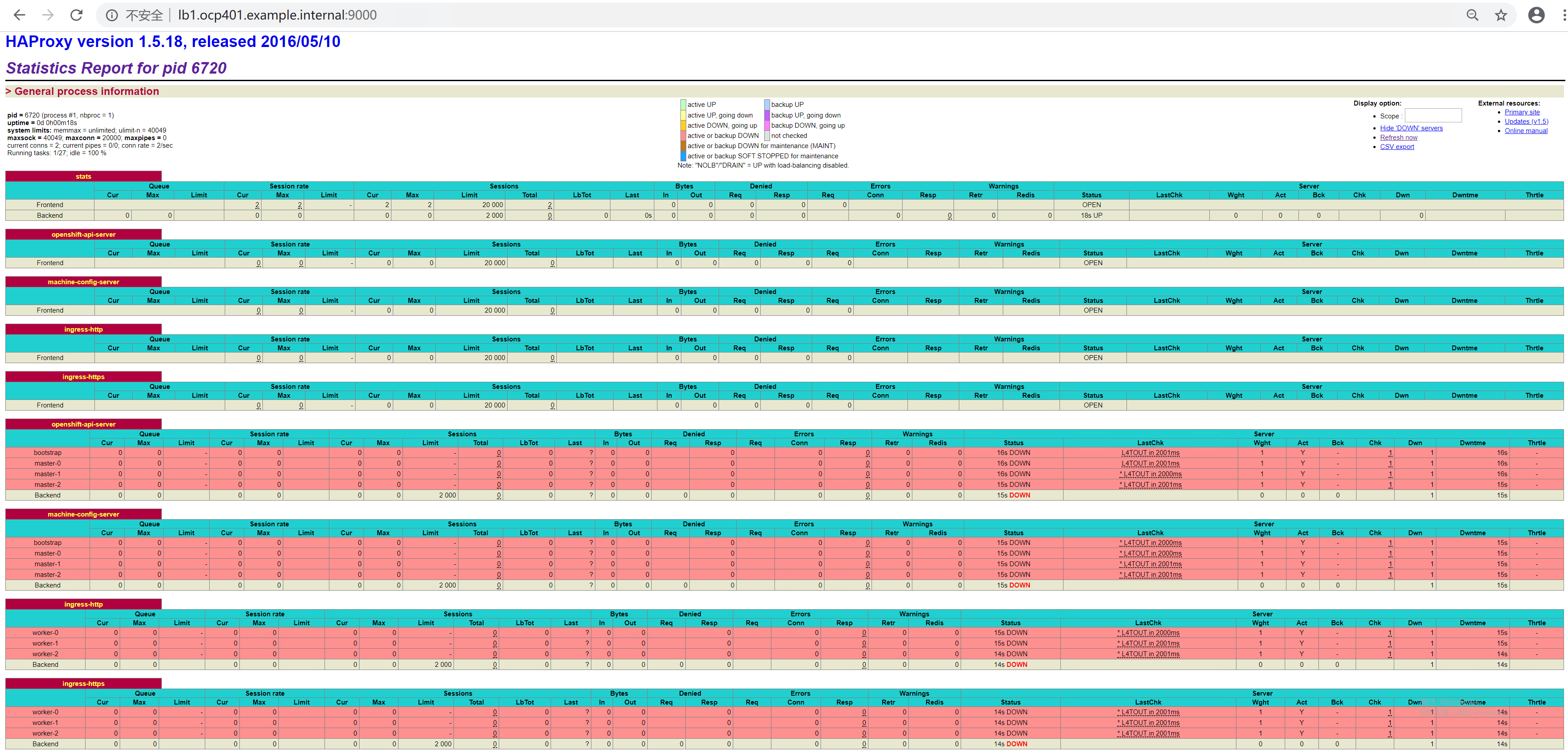
LISTEN 0 128 \*:6443 \*:\* users:(("haproxy",pid=5968,fd=7))

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}

4.3.18,4.3.8,/data/OCP-4.3.18/ocp/secret/registry-secret.json,registry.example.internal:5000,ocp4/openshift4

## 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: 4.3.18

## 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。注意：不同的IaaS软件使用的名称不一样，KVM中缺省使用ens3

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-4.3.8-x86\_64-metal.x86\_64.raw.gz coreos.inst.ignition\_url=http://yum.example.internal:8080/ignition/ocp4-1/bootstrap.ign ip=XXXXXXXXX::192.168.1.1:255.255.255.0:bootstrap.ocp4-1.example.internal:ens3:none nameserver=10.101.31.159

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-4.3.8-x86\_64-metal.x86\_64.raw.gz coreos.inst.ignition\_url=http://yum.example.internal:8080/ignition/ocp4-1/bootstrap.ign ip=XXXXXXXXX::192.168.1.1:255.255.255.0:bootstrap.ocp4-1.example.internal:ens3:none nameserver=10.101.31.159

#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/work-0.iso

curl -I -s http://${YUM\_DOMAIN}/rhcos-iso/ocp4-1/work-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 v4.3.18

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。

**注意**：将硬盘设置位虚拟机启动顺序的最高优先级。每个虚拟机挂在对应名称的iso文件。

# 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

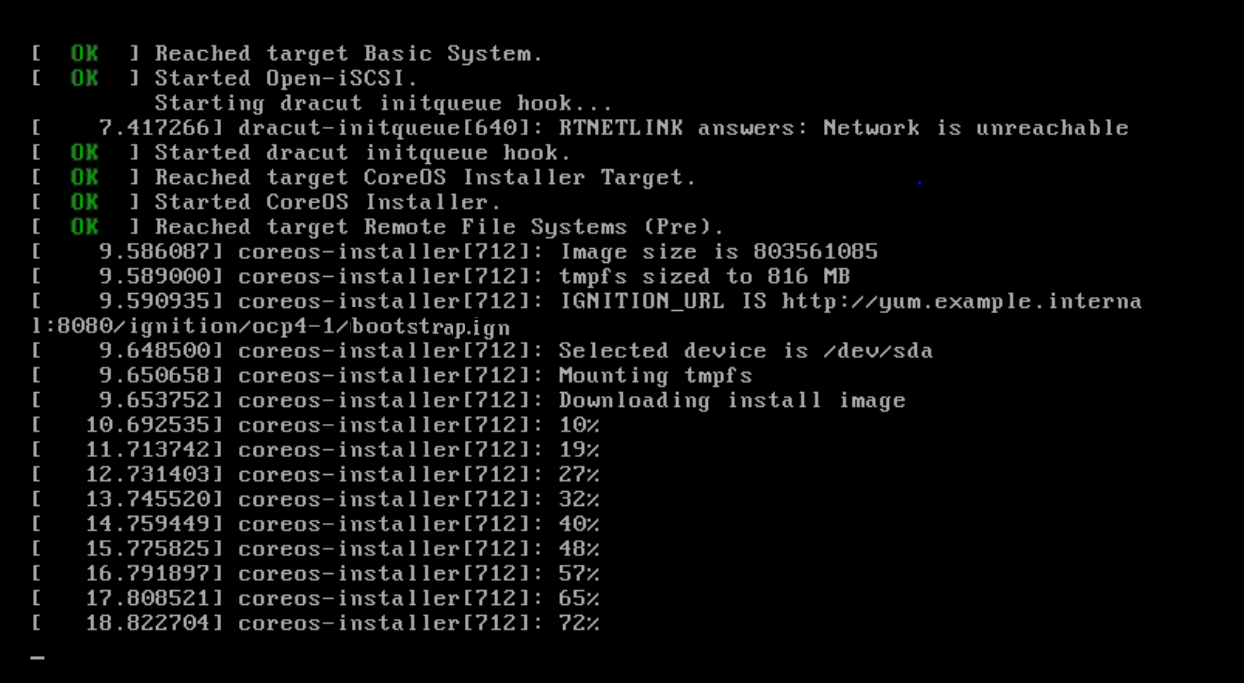
curl -I -s http://${YUM\_DOMAIN}/ignition/${OCP\_CLUSTER\_ID}/bootstrap.ign

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. 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. 检查是否能够拉取镜像库的镜像。

podman pull ${REG\_DOMAIN}/${REPO\_NAME}:4.3.18-jenkins

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)

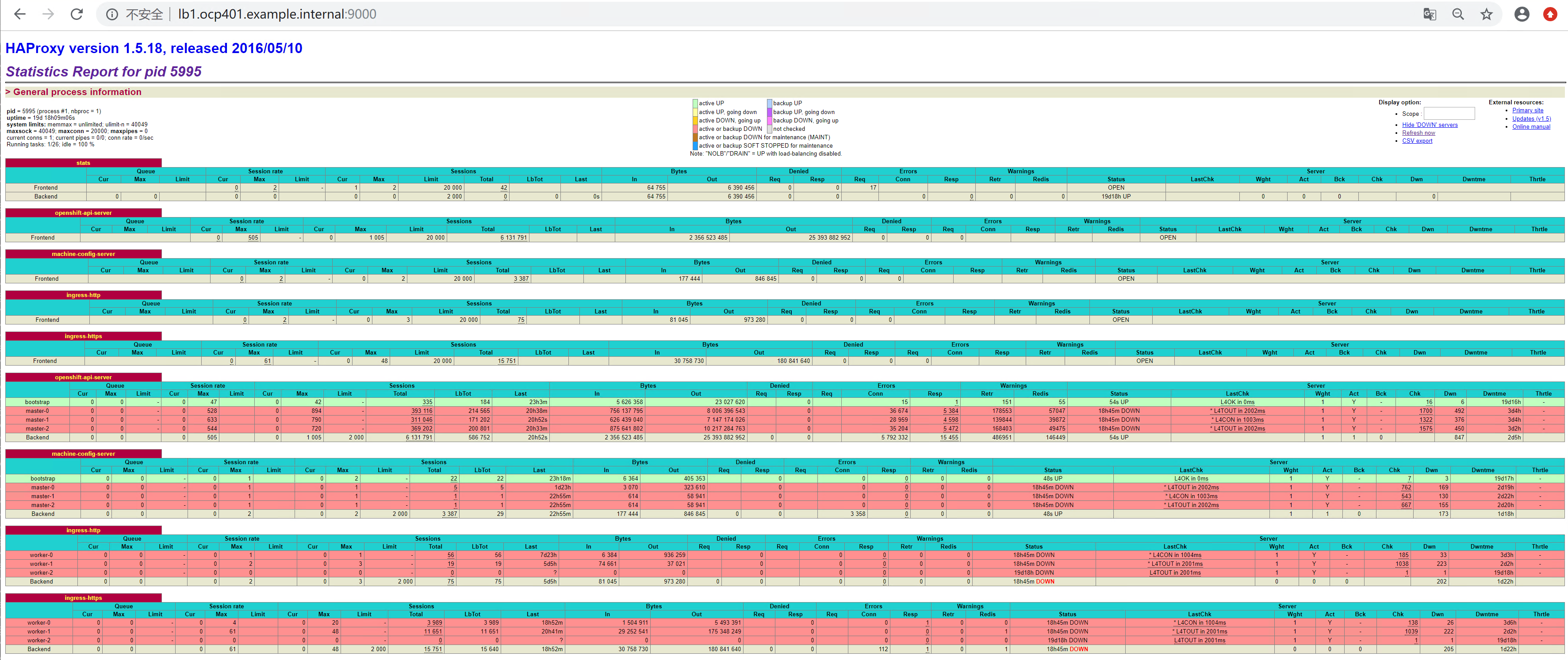
Mar 24 08:55:18 bootstrap.ocp4-1.example.internal bootkube.sh[2424]: {"level":"warn","ts":"2020-03-24T08:55:18.436Z","caller":"clientv3/retry\_interceptor.go:61","msg":"retrying of unary invoker failed","target":"endpoint://client-d7c98144-0f92-4ea8-ac8a-f33ef2e57d60/etcd-0.ocp4-1.example.internal:2379","attempt":0,"error":"rpc error: code = DeadlineExceeded desc = latest connection error: connection error: desc = \"transport: Error while dialing dial tcp 10.101.31.160:2379: 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

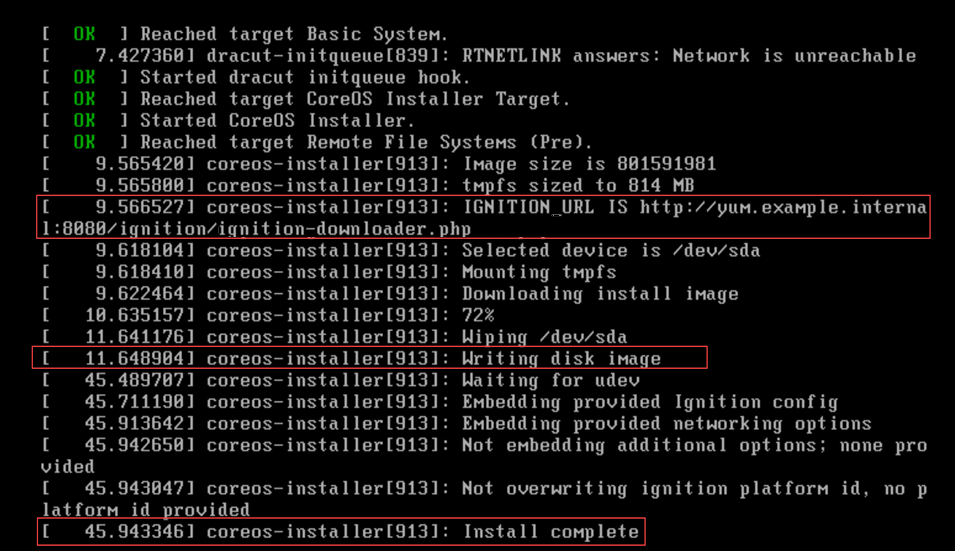
访问如下地址<http://lb.ocp4-1.example.internal:9000/>，结果和下图一致。



## 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"]}

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 v4.3.18"

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 v1.16.2 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 4.3.18

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。

mkdir ~/.kube

cp ${IGN\_PATH}/auth/kubeconfig ~/.kube/config

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 v1.16.2

检查master配置过程

oc get clusteroperators

NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE

authentication Unknown Unknown True 8m32s

cloud-credential 4.3.18 True False False 15m

cluster-autoscaler 4.3.18 True False False 6m28s

console 4.3.18 Unknown True False 7m28s

dns 4.3.18 True False False 10m

image-registry 4.3.18 True False False 7m6s

ingress unknown False True True 7m1s

insights 4.3.18 True False False 12m

kube-apiserver 4.3.18 True False False 9m41s

kube-controller-manager 4.3.18 True False False 9m46s

kube-scheduler 4.3.18 True False False 9m28s

machine-api 4.3.18 True False False 10m

machine-config 4.3.18 True False False 9m57s

marketplace 4.3.18 True False False 6m25s

monitoring False True True 2m

network 4.3.18 True False False 11m

node-tuning 4.3.18 True False False 8m18s

openshift-apiserver 4.3.18 True False False 7m50s

openshift-controller-manager 4.3.18 True False False 9m36s

openshift-samples 4.3.18 True False False 3m6s

operator-lifecycle-manager 4.3.18 True False False 11m

operator-lifecycle-manager-catalog 4.3.18 True False False 11m

operator-lifecycle-manager-packageserver 4.3.18 True False False 8m14s

service-ca 4.3.18 True False False 11m

service-catalog-apiserver 4.3.18 True False False 8m33s

service-catalog-controller-manager 4.3.18 True False False 8m33s

storage 4.3.18 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 4.3.18: 99% complete, waiting on authentication, console, ingress, monitoring

oc get clusterversion

NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version False True 17m Unable to apply 4.3.18: 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 REQUESTOR CONDITION

csr-2jl6p 4m40s system:node:worker-1.ocp4-1.example.internal Pending

csr-7f62t 58m system:node:master-0.ocp4-1.example.internal Pending

csr-f9l8b 9m45s system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Pending

csr-jj785 10m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Pending

csr-mb9zf 58m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Pending

csr-s9szw 4m38s system:node:worker-0.ocp4-1.example.internal Pending

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 REQUESTOR CONDITION

csr-2srwk 71m system:node:master-0.ocp4-1.example.internal Approved,Issued

csr-44wcw 7m45s system:node:worker-0.ocp4-1.example.internal Approved,Issued

csr-7lrkt 19m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-cctqv 34m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-gfkg2 7m41s system:node:worker-1.ocp4-1.example.internal Approved,Issued

csr-hnqsr 71m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-tlvlk 34m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

csr-vfnqn 19m system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Approved,Issued

oc get node

NAME STATUS ROLES AGE VERSION

master-0.ocp4-1.example.internal Ready master 56m v1.16.2

worker-0.ocp4-1.example.internal NotReady worker 39s v1.16.2

worker-1.ocp4-1.example.internal NotReady worker 33s v1.16.2

oc get node

NAME STATUS ROLES AGE VERSION

master-0.ocp4-1.example.internal Ready master 58m v1.16.2

worker-0.ocp4-1.example.internal Ready worker 108s v1.16.2

worker-1.ocp4-1.example.internal Ready worker 102s v1.16.2

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 False True 86m Unable to apply 4.3.18: some cluster operators have not yet rolled out

tail -f 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...

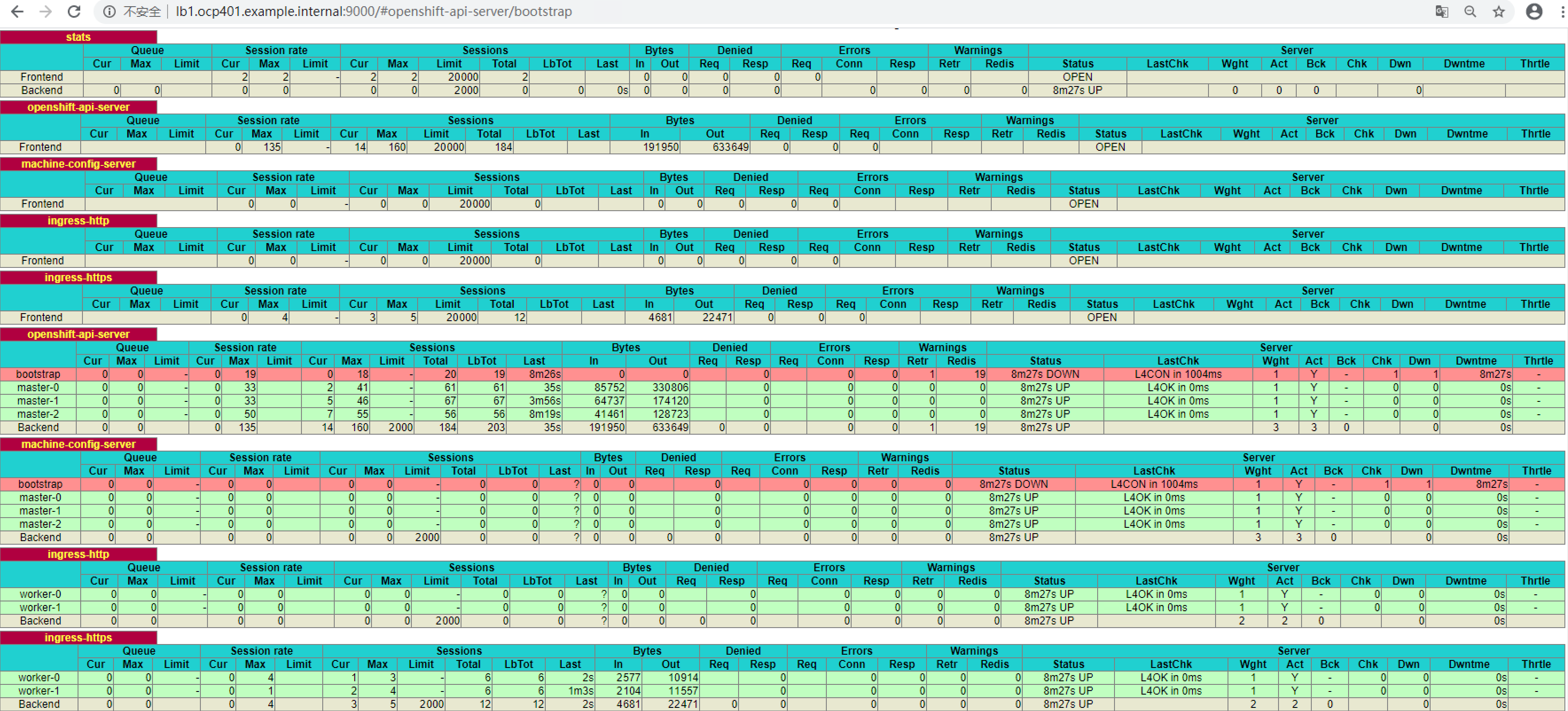
INFO Install complete!

INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/data/ocp/upi/baremetal/auth/kubeconfig'

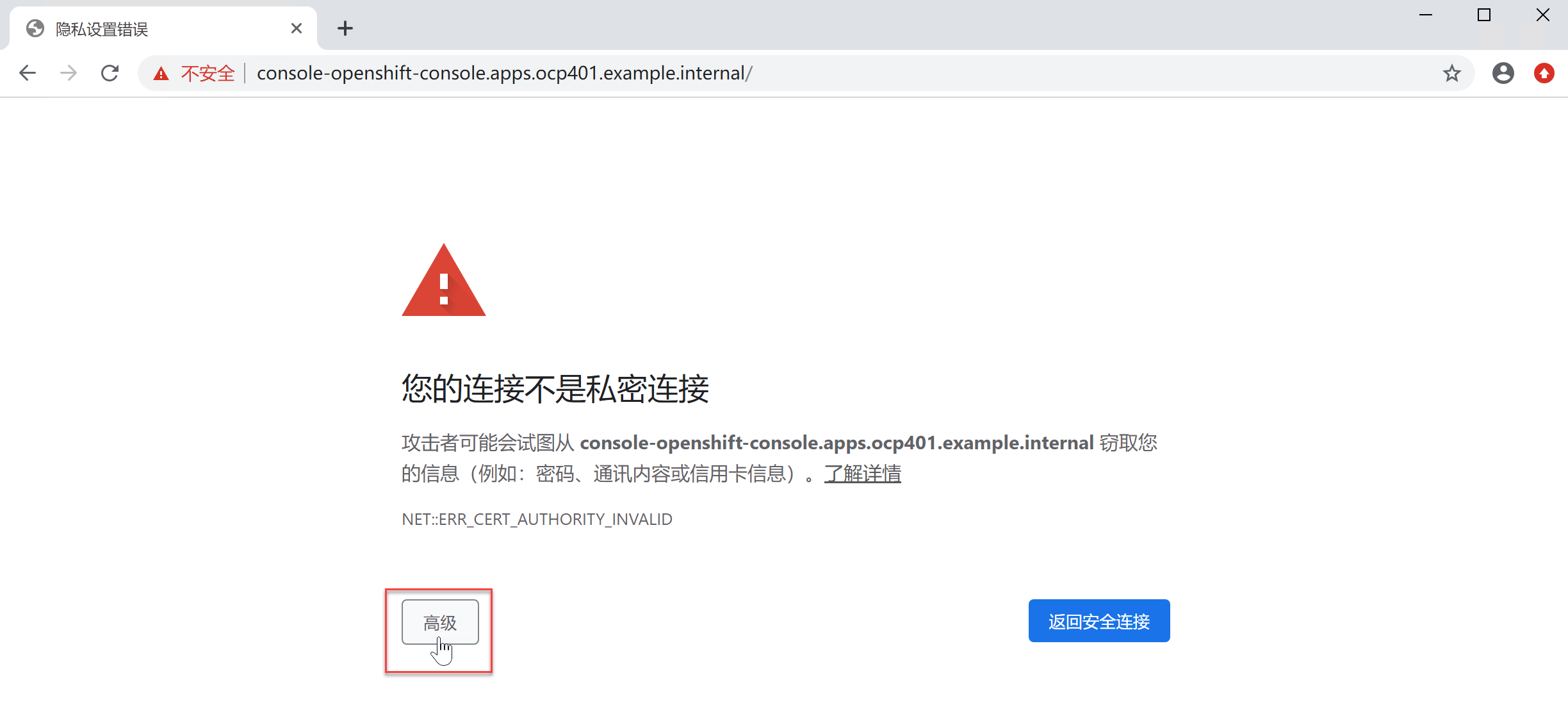
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp4-1.example.internal

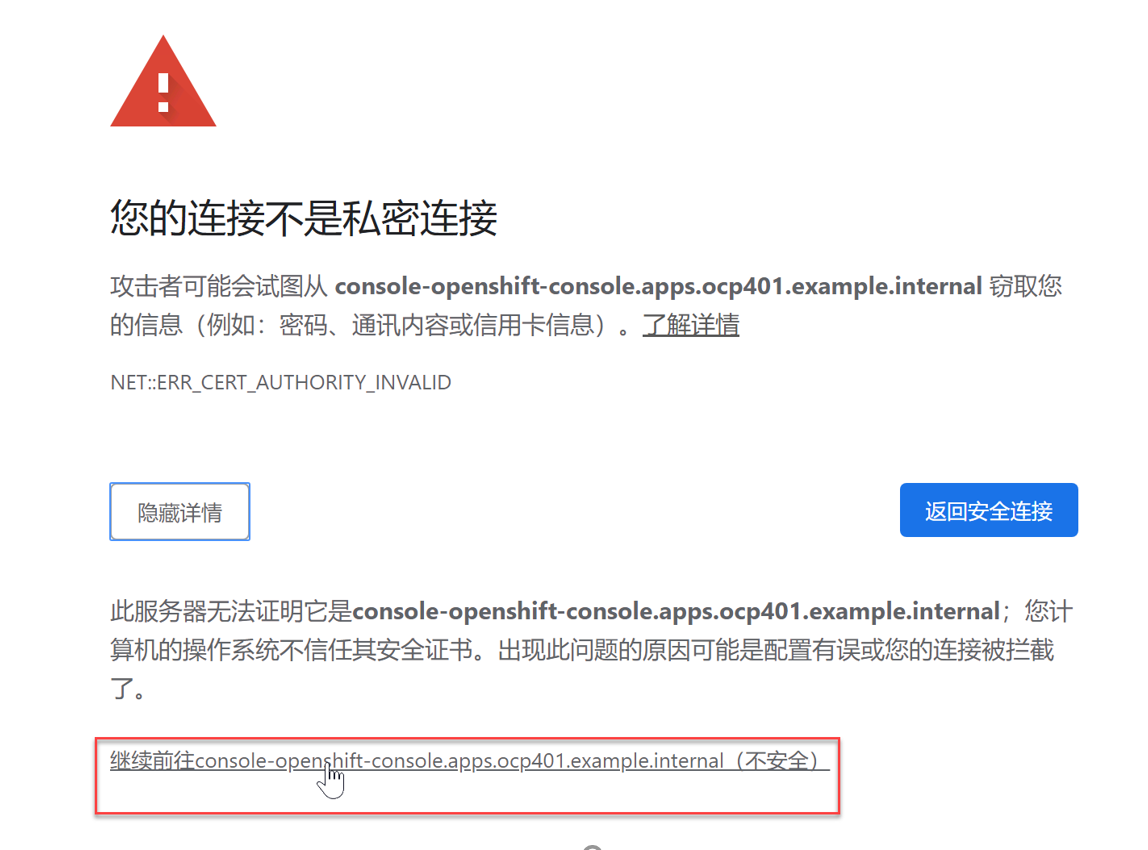
INFO Login to the console with user: kubeadmin, password: u26K4-pFVtr-a5WEZ-rYd2X

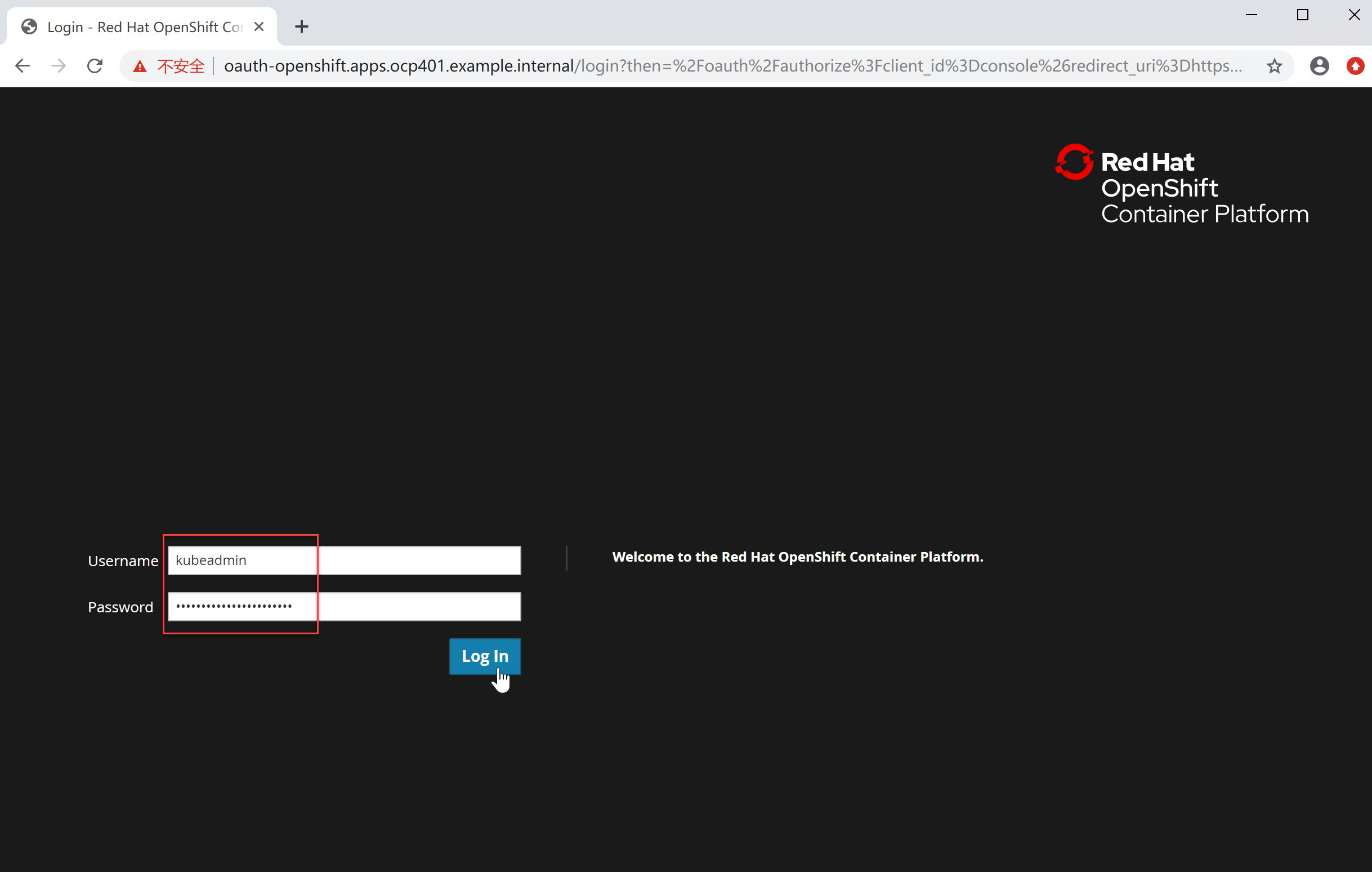
观察haproxy监控页面，如下所示：



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/4.3/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

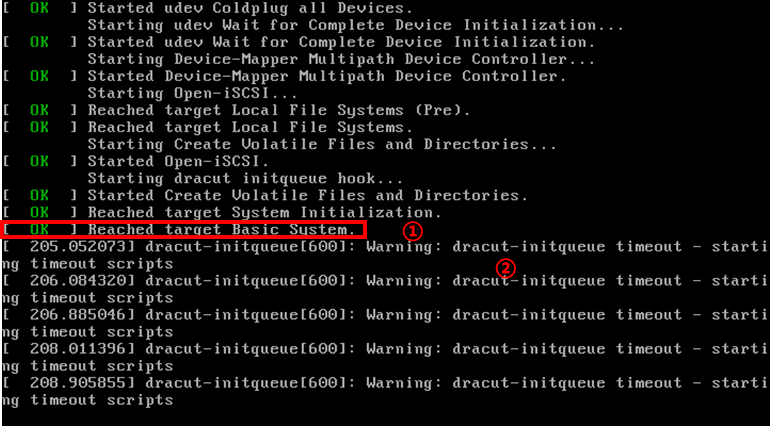
oc delete project -n ${NFS\_NAMESPACE}

# 安装常见错误

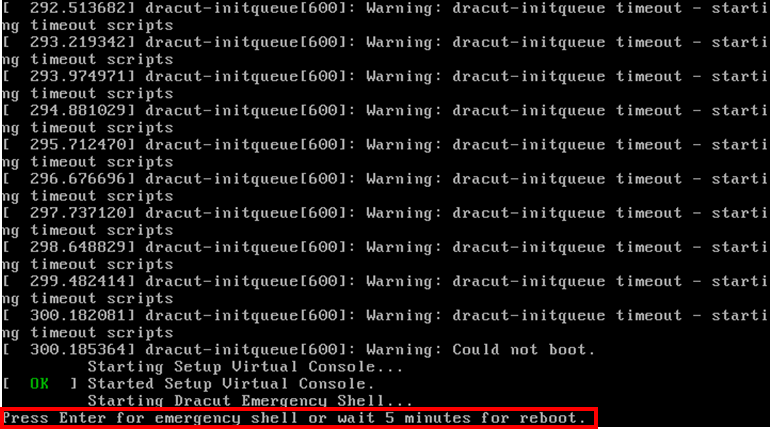
## bootstrap网卡名错误导致启动失败

### 现象

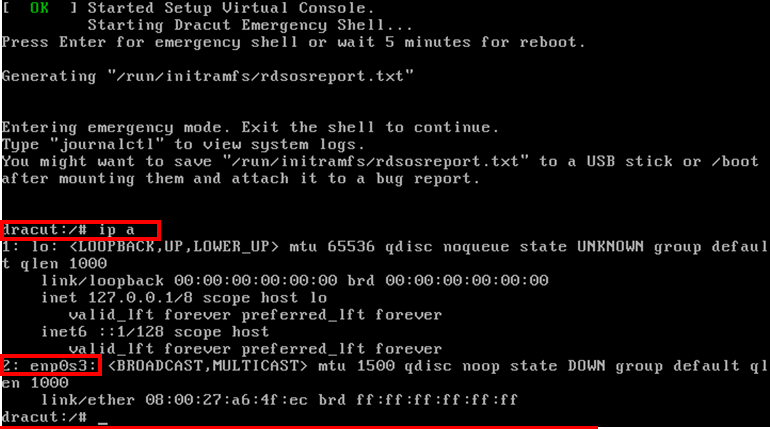
1. 在bootstrap启动后停留在下图中的**①**处约5分钟后，然后反复出现**②**的错误。



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

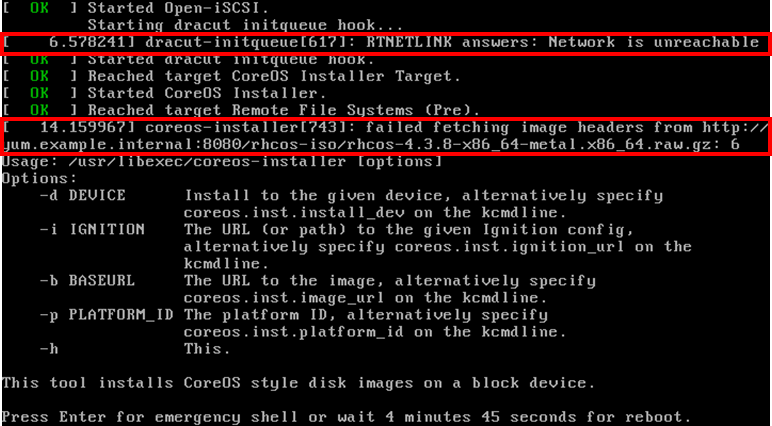
1. 确认bootstrap节点使用的网络接口名和support节点中的“NET\_IF\_NAME”环境变量设置的名称是一致的。如果两者不一致那么需要修复错误。

### 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-4.3.8-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-4.3.8-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。