# 一、资源分配

此次openshift架构包括 2节点的负载均衡（haproxy）节点，3 节点的 master ，2节点的外部 registry，2个计算节点，3个基础架构节点，2个exrouter，2个 inrouter共16台服务器的Docker运行环境。具体的配置信息如下表格所示（OSE的全部环境都基于RHEL 7.3）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **节点** | **域名** | **IP地址** | **功能** | **配置** |
| haproxy-1 | SISVR02254 | 10.200.5.177 | 负载均衡 | 8c/16g(VM) |
| haproxy-2 | SISVR02255 | 10.200.5.178 | 负载均衡 | 8c/16g(VM) |
| Master1 | SISVR02251 | 10.203.65.53 | Master | 8c/16g(VM) |
| Master2 | SISVR02252 | 10.203.65.54 | Master | 8c/16g(VM) |
| Master3 | SISVR02253 | 10.203.65.55 | Master | 8c/16g(VM) |
| Node1 | SISHB00305 | 10.203.64.105 | 计算节点 | 20c/256g(PM) |
| Node2 | SISHB00401 | 10.203.64.121 | 计算节点 | 20c/256g(PM) |
| Node3 | SISHB00402 | 10.203.64.122 | 计算节点 | 20c/256g(PM) |
| Node4 | SISHB00403 | 10.203.64.123 | 计算节点 | 20c/256g(PM) |
| Infra-node01 | SISHB00501 | 10.203.64.141 | 基础架构节点 | 20c/256g(PM) |
| Infra-node02 | SISHB00502 | 10.203.64.142 | 基础架构节点 | 20c/256g(PM) |
| Infra-node03 | SISHB00503 | 10.203.64.143 | 基础架构节点 | 20c/256g(PM) |
| exrouter1 | SISHB00303 | 10.203.64.103 | router | 20c/256g(PM) |
| exrouter2 | SISHB00304 | 10.203.64.104 | router | 20c/256g(PM) |
| inrouter1 | SISHB00301 | 10.203.64.101 | router | 20c/256g(PM) |
| inrouter2 | SISHB00302 | 10.203.64.102 | router | 20c/256g(PM) |
| Registry1 | SISHB00404 | 10.203.64.124 | registry,YUM | 20c/256g(PM) |
| Registry2 | SISHB00405 | 10.203.64.125 | registry | 20c/256g(PM) |

存储配置：两块盘一块作为系统盘一块作为 docker 共享存储（sda/80G sdb/虚拟机-300G，物理机 sda3-700G）：

持久化存储：

1. 日志集群（3个es集群节点）挂载 emcpowera/666G；

2. 配置 NFS 存储 inregistry/400G + exregistry/100G。

相对应的各个集群环境由以下所配置的 VIP进行负载均衡，负载均衡高可用在防火墙策略上进行配置，当然也可使用HAproxy等应用配置相应的VIP负载均衡；:

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **VIP** | **Port** | **BackendIP & Role** |
| 1 |  | 443 | 10.200.5.177（DMZ1）  10.200.5.178（DMZ2） |
| 2 |  | 443/8443/9090 | 10.203.65.53（master1）  10.203.65.54（master2）  10.203.65.55（master3） |
| 3 |  | 80/443 | 10.203.64.103（exrouter1）  10.203.64.104（exrouter2） |
| 4 |  | 80/443/4443/5000 | 10.203.64.124（外部registry1）  10.203.64.125（外部registry2） |
| 5 |  | 80/443 | 10.203.64.101（inrouter1）  10.203.64.102（inrouter2） |
| 6 |  |  | 10.203.64.105（计算节点）  10.203.64.121（计算节点）  10.203.64.122（计算节点）  10.203.64.123（计算节点） |

## 二、基本环境配置

### 2.1 配置SELINUX及开启NetworkManager服务

在所有的节点中，必须开启NetworkManager及selinux服务

systemctl start NetworkManager

systemctl enable NetworkManager

/bin/sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config

setenforce 0

## 2.2 配置yum源仓库

Yum镜像为ose35.tar包，可解压并创建私有的yum仓库

## 2.3 安装相关程序包

所有server node均安装如下程序

yum -y install libcgroup sysstat bridge-utils gcc bind-utils libaio expect ntp libstdc++-devel curl dnsmasq autoconf automake gcc-c++ git libtool xinetd check openssl-devel zlib-devel krb5-devel net-tools libxml2 dos2unix wget telnet tar gzip unzip yum-utils ftp bash-completion lrzsz gpm-libs vim nfs-utils

在master节点安装Openshift

yum -y install atomic-openshift-utils

## 2.4 配置Iptables

Docker必须使用iptables，不能使用系统自带的firewalld

所有server node都需配置iptables

#!/bin/bash

systemctl stop firewalld

systemctl disable firewalld;

yum install iptables-services -y;

systemctl enable iptables

iptables -A INPUT -s 192.0.0.0/8 -j ACCEPT

iptables -A INPUT -s 172.0.0.0/8 -j ACCEPT

iptables -A INPUT -p tcp --sport 22 -j ACCEPT

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A INPUT -p tcp --sport 21 -j ACCEPT

iptables -A INPUT -p tcp --dport 21 -j ACCEPT

iptables -A INPUT -p tcp --sport 81 -j ACCEPT

iptables -A INPUT -p tcp --dport 81 -j ACCEPT

iptables -A INPUT -p tcp --sport 49942 -j ACCEPT

iptables -A INPUT -p tcp --dport 49942 -j ACCEPT

iptables -A INPUT -p udp --sport 20 -j ACCEPT

iptables -A INPUT -p udp --dport 20 -j ACCEPT

iptables -A INPUT -p udp --sport 123 -j ACCEPT

iptables -A INPUT -p udp --dport 123 -j ACCEPT

iptables -A INPUT -p udp --sport 53 -j ACCEPT

iptables -A INPUT -p udp --dport 53 -j ACCEPT

iptables -P INPUT ACCEPT; iptables -P OUTPUT ACCEPT; iptables -P FORWARD ACCEPT

iptables-save |grep -v DROP |grep -v REJECT >/etc/sysconfig/iptables

systemctl restart iptables

systemctl status iptables

## 2.5 配置DNS域名主机名解析

测试域名解析

vim ping.sh

#!/bin/sh

SERVERLIST='

sisvr02251.sgm.shanghaigm.com

sisvr02252.sgm.shanghaigm.com

sisvr02253.sgm.shanghaigm.com

sishb00301.sgm.shanghaigm.com

sishb00302.sgm.shanghaigm.com

sishb00303.sgm.shanghaigm.com

sishb00304.sgm.shanghaigm.com

sishb00305.sgm.shanghaigm.com

sishb00401.sgm.shanghaigm.com

sishb00402.sgm.shanghaigm.com

sishb00403.sgm.shanghaigm.com

sishb00404.sgm.shanghaigm.com

sishb00405.sgm.shanghaigm.com

sishb00501.sgm.shanghaigm.com

sishb00502.sgm.shanghaigm.com

sishb00503.sgm.shanghaigm.com

'

for node in ${SERVERLIST}

do

ping ${node} -c 3

done

#sh ping.sh

## 2.6 配置SSH无秘钥登录

在三台master节点上生成key文件

ssh-keygen

在所有master节点上将key文件复制到各个节点并测试无秘钥登录

#vim ssh.sh

#!/bin/sh

SERVERLIST='

sisvr02252.sgm.shanghaigm.com

sisvr02253.sgm.shanghaigm.com

sishb00301.sgm.shanghaigm.com

sishb00302.sgm.shanghaigm.com

sishb00303.sgm.shanghaigm.com

sishb00304.sgm.shanghaigm.com

sishb00305.sgm.shanghaigm.com

sishb00401.sgm.shanghaigm.com

sishb00402.sgm.shanghaigm.com

sishb00403.sgm.shanghaigm.com

sishb00404.sgm.shanghaigm.com

sishb00405.sgm.shanghaigm.com

sishb00501.sgm.shanghaigm.com

sishb00502.sgm.shanghaigm.com

sishb00503.sgm.shanghaigm.com

'

for node in ${SERVERLIST}

do

ssh-copy-id -i ~/.ssh/id\_rsa.pub ${node}

ssh ${node} 'hostname'

done

## 2.7 配置NTP时间同步

在所有server node上配置ntpd

yum install -y ntp

sed -i '/^server.\*/d' /etc/ntp.conf

echo "server 10.203.238.254 iburst" >>/etc/ntp.conf

systemctl enable ntpd

systemctl restart ntpd

systemctl is-enabled ntpd

enabled

## 2.8 安装docker并配置

在所有server node节点安装docker

yum -y install docker

systemctl enable docker

配置docker选项，registry.jqp.c.saic-gm.net为registry私有仓库的域名

# sed -i s/".\*OPTIONS=.\*"/"OPTIONS=' --selinux-enabled --log-driver=journald --insecur

e-registry 172.30.0.0\/16'"/g /etc/sysconfig/docker

sed -i 's/registry.access.redhat.com/registry.jqp.c.saic-gm.net/g' /etc/sysconfig/docker

echo "BLOCK\_REGISTRY='--block-registry public --block-registry registry.access.redhat.com

' ">> /etc/sysconfig/docker

[root@sisvr02251 ~]# vim /etc/sysconfig/docker

ADD\_REGISTRY='--add-registry registry.jqp.c.saic-gm.net --add-registry registry.access.red

hat.com'

BLOCK\_REGISTRY='--block-registry public --block-registry registry.access.redhat.com'

INSECURE\_REGISTRY='--insecure-registry registry.jqp.c.saic-gm.net'

配置registry.jqp.c.saic-gm.net证书

mkdir -p /etc/docker/certs.d/registry.jqp.c.saic-gm.net \\新建证书存放目录

cat SGMCA2048.pem >> /etc/rhsm/ca/redhat-uep.pem \\将CA证书文件重定向为redhat官方证书文件etc/rhsm/ca/redhat-uep.pem

ln -s /etc/rhsm/ca/redhat-uep.pem /etc/docker/certs.d/registry.jqp.c.saic-gm.net/registry.crt \\将redhat官方CA证书路径链接到自定义的证书路径/etc/docker/certs.d/registry.jqp.c.saic-gm.net/registry.crt

# 将CA证书追加到redhat官方CA文件路径中则可避免修改相关配置文件的CA证书文件的配置选项

[root@sisvr02251 ~]# ll /etc/docker/certs.d/registry.jqp.c.saic-gm.net/registry.crt

lrwxrwxrwx. 1 root root 27 Mar 12 15:19 /etc/docker/certs.d/registry.jqp.c.saic-gm.net/registry.crt -> /etc/rhsm/ca/redhat-uep.pem

配置docker存储

systemctl stop docker

rm -rf /var/lib/docker/\*

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

虚拟机：

cat <<EOF >/etc/sysconfig/docker-storage-setup

DEVS=/dev/sdb

VG=docker-vg

SETUP\_LVM\_THIN\_POOL=yes

EOF

物理机：

//有外挂存储的机子，如3个 es 节点，需要更改 filter

# vim /etc/lvm/lvm.conf

…

filter = [ "a|/dev/sda[0-9]\*|", "a|/dev/emcpower.\*|", "r|.\*|" ]

cat <<EOF >/etc/sysconfig/docker-storage-setup

VG=vg01

SETUP\_LVM\_THIN\_POOL=yes

EOF

lvmconf --disable-cluster

docker-storage-setup

systemctl restart docker

# 三、配置Docker Registry

## 3,1 安装Registry

Option 1 安装单节点registry

在registry(10.203.64.124)上安装配置Docker Registry

|  |
| --- |
| yum -y install docker-distribution  systemctl enable docker-distribution  systemctl start docker  systemctl start docker-distribution |

Option 2 配置单点Harbor

搭建 VMware 的开源镜像库 Harbor 作为Openshift 平台外部 registry高可用方案。

准备基本环境及修改配置

安装 docker，docker-compose 作为基本环境，准备好 harbor 离线安装包、registry.jqp.c.saic-gm.net.crt & key，SGMCA.pem：

[root@sishb00404 ~]# docker --version

Docker version 1.12.6, build 1398f24/1.12.6

[root@sishb00404 ~]# docker-compose version

docker-compose version 1.17.1, build 6d101fb

docker-py version: 2.5.1

CPython version: 2.7.13

OpenSSL version: OpenSSL 1.0.1t 3 May 2016

准备好registry域名的证书及key文件

[root@sishb00404 ~]# ll /data/docker2/harbor/certs/

total 102

-rwxr-xr-x. 1 root root 2454 Mar 12 17:16 registry.jqp.c.saic-gm.net.cer

-rw-r--r--. 1 root root 1704 Mar 12 16:54 registry.jqp.c.saic-gm.net.key

-rw-r--r--. 1 root root 1602 Mar 12 17:17 SGMCA2048.pem

准备好harbor离线安装包

[root@sishb00404 ~]# ll /var/opt/harbor

-rw-r--r--. 1 root root 539885476 Oct 20 16:56 harbor.v1.2.2.tar.gz

制作 registry.crt 用于 harbor ssl:

[root@sishb00404 ~]# cd /data/docker2/harbor/certs/

[root@sishb00404 ~]# cat registry.jqpqa.c.saic-gm.net.cer SGMCA2048 > registry.crt

[root@sishb00404 ~]# ll /data/docker2/harbor/certs/

total 102

-rw-r--r--. 1 root root 4017 Mar 12 16:54 registry.crt

-rwxr-xr-x. 1 root root 2454 Mar 12 17:16 registry.jqp.c.saic-gm.net.cer

-rw-r--r--. 1 root root 1704 Mar 12 16:54 registry.jqp.c.saic-gm.net.key

-rw-r--r--. 1 root root 1602 Mar 12 17:17 SGMCA2048.pem

将本地的离线harbor122pub.tgz解压到/var/opt/目录下，然后解压

[root@sishb00404 /]# cd /var/opt/

[root@sishb00404 opt]# ls

harbor122jqp.tgz harbormysql.tar harborredis.tar images yum

[root@sishb00404 opt]# tar -zxvf harbor122pub.tgz

[root@sishb00404 opt]# ls

harbor harbor122jqp.tgz harbormysql.tar harborredis.tar images yum

registry连接nfs存储

[root@sishb00404 opt]# cat /etc/fstab

#

# /etc/fstab

# Created by anaconda on Tue Mar 6 13:16:03 2018

#

# Accessible filesystems, by reference, are maintained under '/dev/disk'

# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info

#

/dev/mapper/vg00-rootvol / ext4 defaults 1 1

UUID=94a892f9-1fa6-42f1-a192-37dff8be1600 /boot ext4 defaults 1 2

/dev/mapper/vg00-homevol /home ext4 defaults 1 2

/dev/mapper/vg00-tmpvol /tmp ext4 defaults 1 2

/dev/mapper/vg00-varvol /var ext4 defaults 1 2

/dev/mapper/vg00-swapvol swap swap defaults 0 0

10.203.22.83:/ifs/docker2 /data/docker2 nfs defaults 0 0

[root@sishb00404 opt]# vgs

VG #PV #LV #SN Attr VSize VFree

vg00 1 5 0 wz--n- 120.00g 6.00g

vg01 1 1 0 wz--n- 717.84g 430.71g

[root@sishb00404 opt]# df -Th

Filesystem Type Size Used Avail Use% Mounted on

/dev/mapper/vg00-rootvol ext4 20G 4.1G 15G 23% /

devtmpfs devtmpfs 126G 0 126G 0% /dev

tmpfs tmpfs 126G 128K 126G 1% /dev/shm

tmpfs tmpfs 126G 747M 126G 1% /run

tmpfs tmpfs 126G 0 126G 0% /sys/fs/cgroup

/dev/sda1 ext4 477M 142M 306M 32% /boot

/dev/mapper/vg00-homevol ext4 2.0G 9.9M 1.8G 1% /home

/dev/mapper/vg00-tmpvol ext4 9.8G 37M 9.2G 1% /tmp

/dev/mapper/vg00-varvol ext4 50G 12G 36G 26% /var

tmpfs tmpfs 26G 16K 26G 1% /run/user/42

tmpfs tmpfs 26G 0 26G 0% /run/user/0

10.203.22.83:/ifs/docker2 nfs 100G 3.7G 97G 4% /data/docker2

10.203.22.83:/ifs/docker1 nfs 400G 0 400G 0% /data/docker1

[root@sishb00404 ~]# mkdir -p /data/docker2/harbor/data/notary-db

[root@sishb00404 ~]# chown root /data/docker2/harbor –R

[root@sishb00404 ~]# chmod 777 /data/docker2/harbor/ -R

[root@sishb00404 ~]# ll /data/docker2/harbor/data/ -d

drwxrwxrwx. 8 root root 151 Mar 12 17:18 /data/docker2/harbor/data/

[root@sishb00404 ~]# ll /data/docker2/harbor/ -d

drwxrwxrwx. 5 root root 68 Mar 20 15:45 /data/docker2/harbor/

配置Harbor文件

[root@sishb00404 notary-db]# cd /var/opt/harbor/

[root@sishb00404 harbor]# ll

total 527680

drwxr-xr-x. 4 root root 4096 Jan 15 20:00 common

drwxr-xr-x. 4 root root 4096 Jan 15 20:01 data

-rw-r--r--. 1 root root 1163 Oct 20 16:52 docker-compose.clair.yml

-rw-r--r--. 1 root root 2008 Mar 12 16:27 docker-compose.notary.yml

-rw-r--r--. 1 root root 3405 Mar 12 16:49 docker-compose.yml

-rw-r--r--. 1 root root 4304 Oct 20 16:52 harbor\_1\_1\_0\_template

-rw-r--r--. 1 root root 4437 Mar 12 16:59 harbor.cfg

-rw-r--r--. 1 root root 539885476 Oct 20 16:56 harbor.v1.2.2.tar.gz

-rwxr-xr-x. 1 root root 5332 Oct 20 16:52 install.sh

-rw-r--r--. 1 root root 371640 Oct 20 16:52 LICENSE

drwxr-xr-x. 17 root root 4096 Apr 13 08:00 log

-rw-r--r--. 1 root root 482 Oct 20 16:52 NOTICE

-rwxr-xr-x. 1 root root 17592 Oct 20 16:52 prepare

-rwxr-xr-x. 1 root root 4550 Oct 20 16:52 upgrade

修改docker-compose.notary.yml配置

[root@sishb00404 harbor]# vim docker-compose.notary.yml

…

notary-db:

image: vmware/harbor-notary-db:mariadb-10.1.10

container\_name: notary-db

restart: always

networks:

notary-mdb:

aliases:

- mysql

volumes:

- ./common/config/notary/mysql-initdb.d:/docker-entrypoint-initdb.d

- /data/docker2/harbor/data/notary-db:/var/lib/mysql

environment:

- TERM=dumb

…

修改docker-compose.yml配置，将mysql模块注释

[root@sishb00404 harbor]# vim docker-compose.yml

// 将 harbor-db 模块（mysql） 独立出来用于共享 registry 的目录等数据，将session 数据存于外部 redis

version: '2'

services:

log:

image: vmware/harbor-log:v1.2.2

container\_name: harbor-log

restart: always

volumes:

- /var/opt/harbor/log/:/var/log/docker/:z

ports:

- 127.0.0.1:1514:514

networks:

- harbor

registry:

image: vmware/registry:2.6.2-photon

container\_name: registry

restart: always

volumes:

- /data/exregistry/harbor/data/registry:/storage

- ./common/config/registry/:/etc/registry/:z

networks:

- harbor

environment:

- GODEBUG=netdns=cgo

command:

["serve", "/etc/registry/config.yml"]

depends\_on:

- log

logging:

driver: "syslog"

options:

syslog-address: "tcp://127.0.0.1:1514"

tag: "registry"

# mysql:

# image: vmware/harbor-db:v1.2.2

# container\_name: harbor-db

# restart: always

# volumes:

# - /data/database:/var/lib/mysql:z

# networks:

# - harbor

# env\_file:

# - ./common/config/db/env

# depends\_on:

# - log

# logging:

# driver: "syslog"

# options:

# syslog-address: "tcp://127.0.0.1:1514"

# tag: "mysql"

adminserver:

image: vmware/harbor-adminserver:v1.2.2

container\_name: harbor-adminserver

env\_file:

- ./common/config/adminserver/env

restart: always

volumes:

- /data/exregistry/harbor/data/config/:/etc/adminserver/config/

- /data/exregistry/harbor/data/secretkey:/etc/adminserver/key

- /data/exregistry/harbor/data/:/data/

networks:

- harbor

depends\_on:

- log

logging:

driver: "syslog"

options:

syslog-address: "tcp://127.0.0.1:1514"

tag: "adminserver"

ui:

image: vmware/harbor-ui:v1.2.2

container\_name: harbor-ui

env\_file:

- ./common/config/ui/env

restart: always

volumes:

- ./common/config/ui/app.conf:/etc/ui/app.conf:z

- ./common/config/ui/private\_key.pem:/etc/ui/private\_key.pem:z

- /data/exregistry/harbor/data/secretkey:/etc/ui/key

- /data/exregistry/harbor/data/ca\_download/:/etc/ui/ca/

- /data/exregistry/harbor/data/psc/:/etc/ui/token/

networks:

- harbor

depends\_on:

- log

- adminserver

- registry

logging:

driver: "syslog"

options:

syslog-address: "tcp://127.0.0.1:1514"

tag: "ui"

jobservice:

image: vmware/harbor-jobservice:v1.2.2

container\_name: harbor-jobservice

env\_file:

- ./common/config/jobservice/env

restart: always

volumes:

- /var/opt/harbor/data/job\_logs:/var/log/jobs:z

- ./common/config/jobservice/app.conf:/etc/jobservice/app.conf:z

- /var/opt/harbor/data/secretkey:/etc/jobservice/key:z

networks:

- harbor

depends\_on:

- ui

- adminserver

logging:

driver: "syslog"

options:

syslog-address: "tcp://127.0.0.1:1514"

tag: "jobservice"

proxy:

image: vmware/nginx-photon:1.11.13

container\_name: nginx

restart: always

volumes:

- ./common/config/nginx:/etc/nginx:z

networks:

- harbor

ports:

- 80:80

- 443:443

- 4443:4443

depends\_on:

# - mysql

- registry

- ui

- log

logging:

driver: "syslog"

options:

syslog-address: "tcp://127.0.0.1:1514"

tag: "proxy"

networks:

harbor:

external: false

修改harbor.cfg文件

[root@sishb00404 harbor]# vim harbor.cfg

## Configuration file of Harbor

#The IP address or hostname to access admin UI and registry service.

#DO NOT use localhost or 127.0.0.1, because Harbor needs to be accessed by external clients.

hostname = registry.jqp.c.saic-gm.net

#The protocol for accessing the UI and token/notification service, by default it is http.

#It can be set to https if ssl is enabled on nginx.

ui\_url\_protocol = https

#The password for the root user of mysql db, change this before any production use.

db\_password = root123

#Maximum number of job workers in job service

max\_job\_workers = 3

#Determine whether or not to generate certificate for the registry's token.

#If the value is on, the prepare script creates new root cert and private key

#for generating token to access the registry. If the value is off the default key/cert will be used.

#This flag also controls the creation of the notary signer's cert.

customize\_crt = on

#The path of cert and key files for nginx, they are applied only the protocol is set to https

ssl\_cert = /data/docker2/harbor/certs/registry.crt

ssl\_cert\_key = /data/docker2/harbor/certs/registry.jqp.c.saic-gm.net.key

#The path of secretkey storage

secretkey\_path = /data/docker2/harbor/data/secret

#Admiral's url, comment this attribute, or set its value to NA when Harbor is standalone

admiral\_url = NA

#The password of the Clair's postgres database, only effective when Harbor is deployed with Clair.

#Please update it before deployment, subsequent update will cause Clair's API server and Harbor unable to access Clair's database.

clair\_db\_password = password

#NOTES: The properties between BEGIN INITIAL PROPERTIES and END INITIAL PROPERTIES

#only take effect in the first boot, the subsequent changes of these properties

#should be performed on web ui

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BEGIN INITIAL PROPERTIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Email account settings for sending out password resetting emails.

#Email server uses the given username and password to authenticate on TLS connections to host and act as identity.

#Identity left blank to act as username.

email\_identity =

email\_server = smtp.mydomain.com

email\_server\_port = 25

email\_username = sample\_admin@mydomain.com

email\_password = abc

email\_from = admin <sample\_admin@mydomain.com>

email\_ssl = false

##The initial password of Harbor admin, only works for the first time when Harbor starts.

#It has no effect after the first launch of Harbor.

#Change the admin password from UI after launching Harbor.

harbor\_admin\_password = passw0rd

##By default the auth mode is db\_auth, i.e. the credentials are stored in a local database.

#Set it to ldap\_auth if you want to verify a user's credentials against an LDAP server.

auth\_mode = db\_auth

#The url for an ldap endpoint.

ldap\_url = ldaps://ldap.mydomain.com

#A user's DN who has the permission to search the LDAP/AD server.

#If your LDAP/AD server does not support anonymous search, you should configure this DN and ldap\_search\_pwd.

#ldap\_searchdn = uid=searchuser,ou=people,dc=mydomain,dc=com

#the password of the ldap\_searchdn

#ldap\_search\_pwd = password

#The base DN from which to look up a user in LDAP/AD

ldap\_basedn = ou=people,dc=mydomain,dc=com

#Search filter for LDAP/AD, make sure the syntax of the filter is correct.

#ldap\_filter = (objectClass=person)

# The attribute used in a search to match a user, it could be uid, cn, email, sAMAccountName or other attributes depending on your LDAP/AD

ldap\_uid = uid

#the scope to search for users, 1-LDAP\_SCOPE\_BASE, 2-LDAP\_SCOPE\_ONELEVEL, 3-LDAP\_SCOPE\_SUBTREE

ldap\_scope = 3

#Timeout (in seconds) when connecting to an LDAP Server. The default value (and most reasonable) is 5 seconds.

ldap\_timeout = 5

#Turn on or off the self-registration feature

self\_registration = off

#The expiration time (in minute) of token created by token service, default is 30 minutes

token\_expiration = 30

#The flag to control what users have permission to create projects

#The default value "everyone" allows everyone to creates a project.

#Set to "adminonly" so that only admin user can create project.

project\_creation\_restriction = adminonly

#Determine whether the job service should verify the ssl cert when it connects to a remote registry.

#Set this flag to off when the remote registry uses a self-signed or untrusted certificate.

verify\_remote\_cert = off

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END INITIAL PROPERTIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

修改配置文件将 db 和 session 存储到外面的 mysql 和 redis 进行共享，前期为部署 OCP，先搭建单点 Harbor，将 mysql，redis 容器先部署在本机

提前手动将 harbor 镜像导出

[root@sishb00404 harbor]# pwd

/var/opt/harbor

[root@sishb00404 harbor]# ll harbor.v1.2.2.tar.gz

-rw-r--r--. 1 root root 539885476 Oct 20 16:56 harbor.v1.2.2.tar.gz

[root@sishb00404 harbor]# docker load -i harbor.v1.2.2.tar.gz

手动将redis及mysql容器镜像导出

[root@sishb00404 opt]# cd /var/opt/

[root@sishb00404 opt]# ll

total 1118668

drwxr-xr-x. 5 root root 4096 Apr 13 09:35 harbor

-rw-r--r--. 1 root root 536057684 Mar 20 15:38 harbor122jqp.tgz

-rw-------. 1 root root 379904000 Mar 7 15:11 harbormysql.tar

-rw-------. 1 root root 229535232 Mar 7 15:11 harborredis.tar

[root@sishb00404 harbor]# docker load -i harbormysql.tar

[root@sishb00404 harbor]# docker load -i harborredis.tar

部署单点

[root@sishb00404 harbor]# docker run -p 31001:6379 --name harbor\_redis -d registry.jqpqa.c.saic-gm.net:5000/harborredis

[root@sishb00404 harbor]# docker run -p 31002:3306 --name harbor\_db -e MYSQL\_USR=root -e MYSQL\_ROOT\_PASSWORD=root123 -d vmware/harbor-db:v1.2.2

部署高可用，部署在OCP上

将 harbormysql，harborredis 部署到 default 中，并将 harbormysql 数据持久化，将2个应用用 nodeport 暴露访问

# oc project default

// dc中注意更改route及镜像地址成对应环境名称

# oc create -f harbormysql.yaml

# oc create -f harborredis.yaml

# oc create -f harbormysql-pv.yaml

# oc create -f harbormysql-pvc.yaml

//因这两个应用将部署在inrouter节点上，把harbormysql.tar，harborredis.tar传到 inrouter 节点 load 出并根据需要 tag 改成对应镜像库的名字

# docker load -i harbormysql.tar

# docker load -i harborredis.tar

//页面上发布这2个应用，并将pvc加到harbormysql的add storage将/var/lib/mysql/data持久化

//将两个应用的 nodeport 暴露

# oc patch svc harborredis -p '{"spec":{"type": "NodePort"}}'

# oc patch svc harbormysql -p '{"spec":{"type": "NodePort"}}'

# oc get svc harborredis -o json | sed 's/"nodePort": 3..../"nodePort": 31001/g' | oc replace -f -

# oc get svc harborredis -o json | sed 's/"nodePort": 3..../"nodePort": 31002/g' | oc replace -f -

// 配置 Harbor

[root@sishb00404 adminserver]# pwd

/var/opt/harbor/common/templates/adminserver

[root@sishb00404 adminserver]# vim common/templates/adminserver/env

…

DATABASE\_TYPE=mysql

MYSQL\_HOST=10.203.64.121

MYSQL\_PORT=31002

MYSQL\_USR=root

MYSQL\_PWD=$db\_password

MYSQL\_DATABASE=registry

…

RESET=true

[root@sishb00404 ui]# pwd

/var/opt/harbor/common/templates/ui

[root@sishb00404 ui]# vim env

LOG\_LEVEL=debug

CONFIG\_PATH=/etc/ui/app.conf

UI\_SECRET=$ui\_secret

JOBSERVICE\_SECRET=$jobservice\_secret

GODEBUG=netdns=cgo

\_REDIS\_URL=10.203.64.121:31001

//将对应域名的证书放到 nfs 上

# cat registry.jqp.c.saic-gm.net.cer SGMCA2048.pem > registry.crt

[root@sishb00404 ~]# ll /data/docker2/harbor/certs/

total 102

-rw-r--r--. 1 root root 4017 Mar 12 16:54 registry.crt

-rwxr-xr-x. 1 root root 2454 Mar 12 17:16 registry.jqp.c.saic-gm.net.cer

-rw-r--r--. 1 root root 1704 Mar 12 16:54 registry.jqp.c.saic-gm.net.key

-rw-r--r--. 1 root root 1602 Mar 12 17:17 SGMCA2048.pem

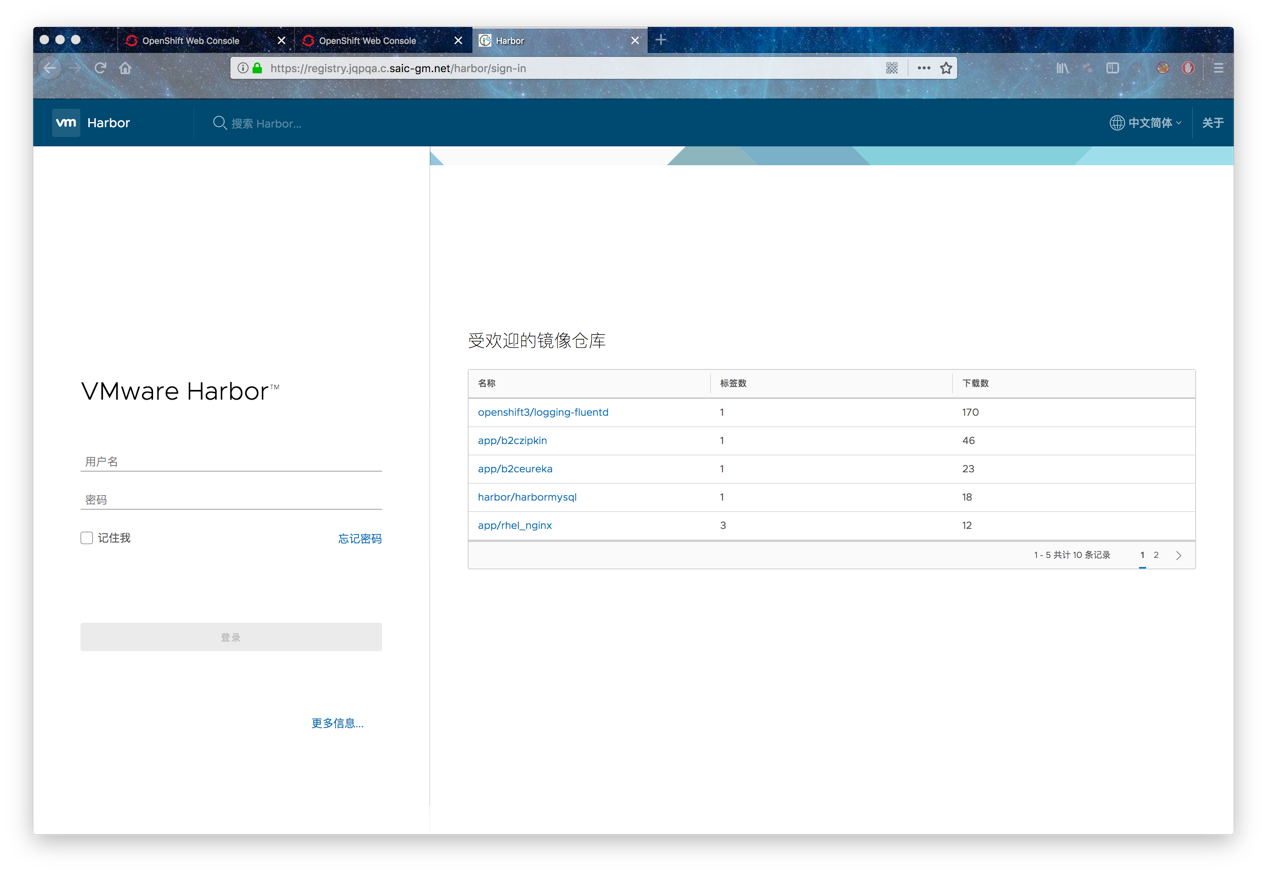
启用Harbor服务

# cd /var/opt/harbor/

# ./install.sh --with-notary

//更改本地 hosts 文件域名指向10.203.64.124

用 admin/password 登录可以正常访问



将服务停下并打包传到 registry02（10.203.64.125）并进行访问验证（改 hosts）

停止harbor服务

# docker-compose -f docker-compose.yml -f docker-compose.notary.yml down –v

# tar -zcvf harbor122.tgz /var/opt/harbor

# scp harbor122.tgz root@10.203.103.106:/var/opt

// 在 registry02上

# cd /var/opt

# tar -zxvf harbor122.tgz

# cd harbor

启动harbor服务

# docker-compose -f docker-compose.yml -f docker-compose.notary.yml up -d

//更改本地 hosts 文件域名指向10.203.64.125 并进行登录功能访问验证

高可用验证

回到 registry01上将 harbor 启动：

# cd /var/opt/harbor

# docker-compose -f docker-compose.yml -f docker-compose.notary.yml up -d

改hosts文件在registry01上登录harbor后创建project并上传镜像，再改 hosts文件为registry02，刷新页面可以看到可以保持登录状态，并且可以看到上传的镜像，并且可以pull成功，高可用harbor搭建完毕。

## 3.2 导入image

OpenShift 3.5对应的镜像包有3个：

ose\_core\_v3.5.5.31.tar

ose\_logging\_3.5.0.tar

ose\_metrics\_3.5.0.tar

上传到registry服务器上，例如保存在/var/opt/images下，然后使用如下命令导入：

[root@sishb00404 images]# pwd

/var/opt/images

[root@sishb00404 images]# ll

total 5433928

-rwxr-xr-x. 1 root root 1510175744 Mar 7 15:08 ose\_core\_v3.5.5.31.tar

-rwxr-xr-x. 1 root root 1367774720 Mar 7 15:08 ose\_logging\_3.5.0.tar

-rwxr-xr-x. 1 root root 2686373888 Mar 7 15:08 ose\_metrics\_3.5.0.tar

[root@sishb00404 images]# docker load -i ose\_core\_v3.5.5.31.tar

[root@sishb00404 images]# docker load -i ose\_metrics\_3.5.0.tar

[root@sishb00404 images]# docker load -i ose\_logging\_3.5.0.tar

把上述镜像更改对应镜像库标签后推送到本地Registry：

[root@sishb00404 ~]# vim openshift.sh

#!/bin/bash

#REDHAT\_REG="registry.access.redhat.com"

REDHAT\_REG="registry.jqpqa.c.saic-gm.net"

PRIVATE\_REG="registry.jqp.c.saic-gm.net"

for i in $(docker images|grep $REDHAT\_REG|awk '{print $1":"$2}') ; do docker tag $i "$PRIVATE\_REG$(echo $i|awk -F 'com' {'print $2'})" ; done;

#for i in `docker images|grep $PRIVATE\_REG|awk '{print $1}'` ; do docker push $i; done;

//Harbor，在 harbor上新建openshift3项目并设为公开，将上面脚本 PRIVATE\_REG 改为 registry.jqp.c.saic-gm.net 后上传镜像

# 四、安装Openshift 3.5

## 3.1 准备Ansible hosts文件

使用Ansible安装配置OpenShift 3.5，脚本如下（在master1上准备）

[root@sisvr02251 ~]# cat << EOF >/etc/ansible/hosts

# Logging

openshift\_hosted\_logging\_deploy=true

openshift\_logging\_image\_prefix=registry.jqp.c.saic-gm.net/openshift3/

openshift\_logging\_es\_cluster\_size=3

openshift\_logging\_image\_version=3.5.0

openshift\_logging\_es\_nodeselector={"region":"infra"}

# enable ntp on masters to ensure proper failover

openshift\_clock\_enabled=true

# Router

openshift\_hosted\_router\_selector="env=inrouter"

openshift\_hosted\_router\_replicas=2

# Registry

openshift\_hosted\_registry\_selector="env=inrouter"

openshift\_hosted\_registry\_replicas=2

[OSEv3:children]

masters

nodes

etcd

[masters]

sisvr02251.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02252.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02253.sgm.shanghaigm.com host\_zone=cn-east-sh

# host group for etcd

[etcd]

sisvr02251.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02252.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02253.sgm.shanghaigm.com host\_zone=cn-east-sh

# host group for nodes, includes region info

[nodes]

## master

sisvr02251.sgm.shanghaigm.com openshift\_hostname=sisvr02251.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

sisvr02252.sgm.shanghaigm.com openshift\_hostname=sisvr02252.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

sisvr02253.sgm.shanghaigm.com openshift\_hostname=sisvr02253.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

## These are regular nodes

sishb00301.sgm.shanghaigm.com openshift\_hostname=sishb00301.sgm.shanghaigm.com openshift\_node\_labels="{'env':'inrouter', 'zone': 'cn-east-sh'}"

sishb00302.sgm.shanghaigm.com openshift\_hostname=sishb00302.sgm.shanghaigm.com openshift\_node\_labels="{'env':'inrouter', 'zone': 'cn-east-sh'}"

sishb00303.sgm.shanghaigm.com openshift\_hostname=sishb00303.sgm.shanghaigm.com openshift\_node\_labels="{'env':'exrouter', 'zone': 'cn-east-sh'}"

sishb00304.sgm.shanghaigm.com openshift\_hostname=sishb00304.sgm.shanghaigm.com openshift\_node\_labels="{'env':'exrouter', 'zone': 'cn-east-sh'}"

## infra nodes

sishb00501.sgm.shanghaigm.com openshift\_hostname=sishb00501.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

sishb00502.sgm.shanghaigm.com openshift\_hostname=sishb00502.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

sishb00503.sgm.shanghaigm.com openshift\_hostname=sishb00503.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

## compute nodes

sishb00305.sgm.shanghaigm.com openshift\_hostname=sishb00305.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00401.sgm.shanghaigm.com openshift\_hostname=sishb00401.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00402.sgm.shanghaigm.com openshift\_hostname=sishb00402.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00403.sgm.shanghaigm.com openshift\_hostname=sishb00403.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

EOF

## 4.2 准备认证文件

在masters（10.203.65.53,54,55）上执行：

yum -y install httpd-tools

mkdir -p /etc/origin/master/htpasswd

htpasswd -c /etc/origin/master/htpasswd/user admin

[root@sisvr02251 ~]# ll /etc/origin/master/htpasswd/user

-rw-r--r--. 1 root root 45 Mar 13 18:44 /etc/origin/master/htpasswd/user

将所需配置的证书（console.jqp.c.saic-gm.net.cer&key, inapps.jqp.c.saic-gm.net.cer&key, SGMCA2048.pem）对应放在 /root/certs 下，对应 ansible hosts文件中的目录配置。

## 4.3 安装openshift

卸载ansible脚本安装操作

ansible-playbook -i /etc/ansible/hosts /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml

安装ansible脚本

ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/config.yml

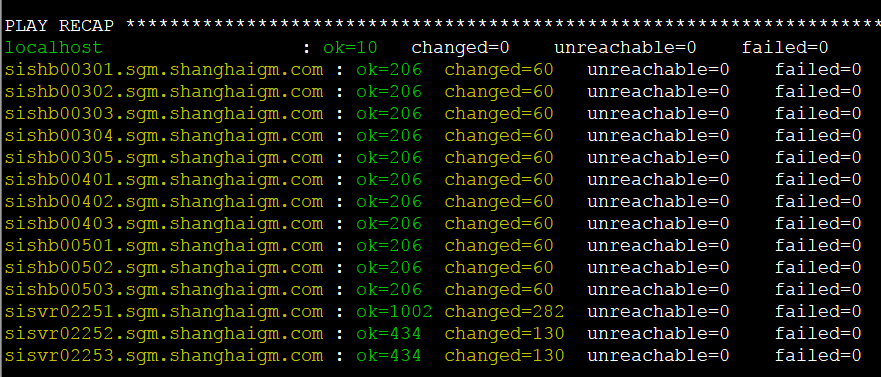
**在master1（10.203.65.53）上执行：**

# 先测试一下通讯

[root@sisvr02251 ~]# ansible all -m ping

[root@sisvr02251 ~]#ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/config.yml

安装过程15~30分钟，最后结果应该全部节点failed=0



# 五、配置Openshift 3.5

## 5．1 配置 Docker 并拉取 Images

修改所有节点（含master，因为安装openshift的时候会修改）的Docker选项。

[root@sisvr02251 ~]# cp /etc/sysconfig/docker /etc/sysconfig/docker.bak.$(date "+%Y%m%d%H%M%S")

确认/etc/sysconfig/docker文件是否发生改变

[root@sisvr02251 ~]# vim /etc/sysconfig/docker

ADD\_REGISTRY='--add-registry registry.jqp.c.saic-gm.net --add-registry registry.access.redhat.com'

BLOCK\_REGISTRY='--block-registry public --block-registry registry.access.redhat.com'

INSECURE\_REGISTRY='--insecure-registry registry.jqp.c.saic-gm.net'

如果需要修改，则必须重新启动docker

systemctl restart docker

在其他Node节点测试拉取Registry的Images：

docker pull registry.jqp.c.saic-gm.net/openshift3/ose-pod:v3.5.5.31

docker pull registry.jqp.c.saic-gm.net/openshift3/ose-deployer:v3.5.5.31

## 5.2 配置用户

在三台Master配置用户并赋予权限：

[root@sisvr02251 ~]# yum -y install httpd-tools

[root@sisvr02251 ~]# htpasswd -cb /etc/origin/master/htpasswd/user redhat passw0rd

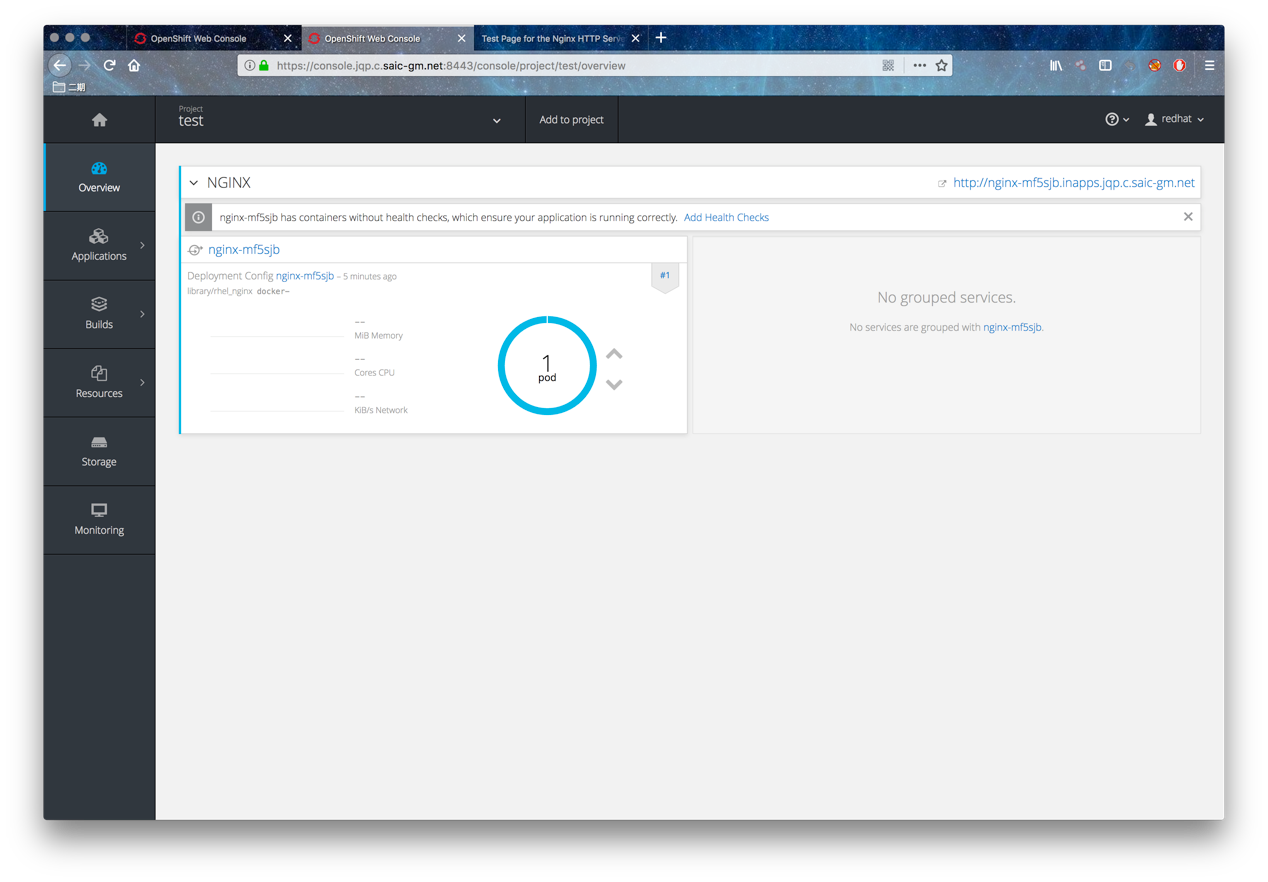
[root@sisvr02251 ~]# oadm policy add-cluster-role-to-user cluster-admin redhat

# 六、发布应用进行平台基本功能验证

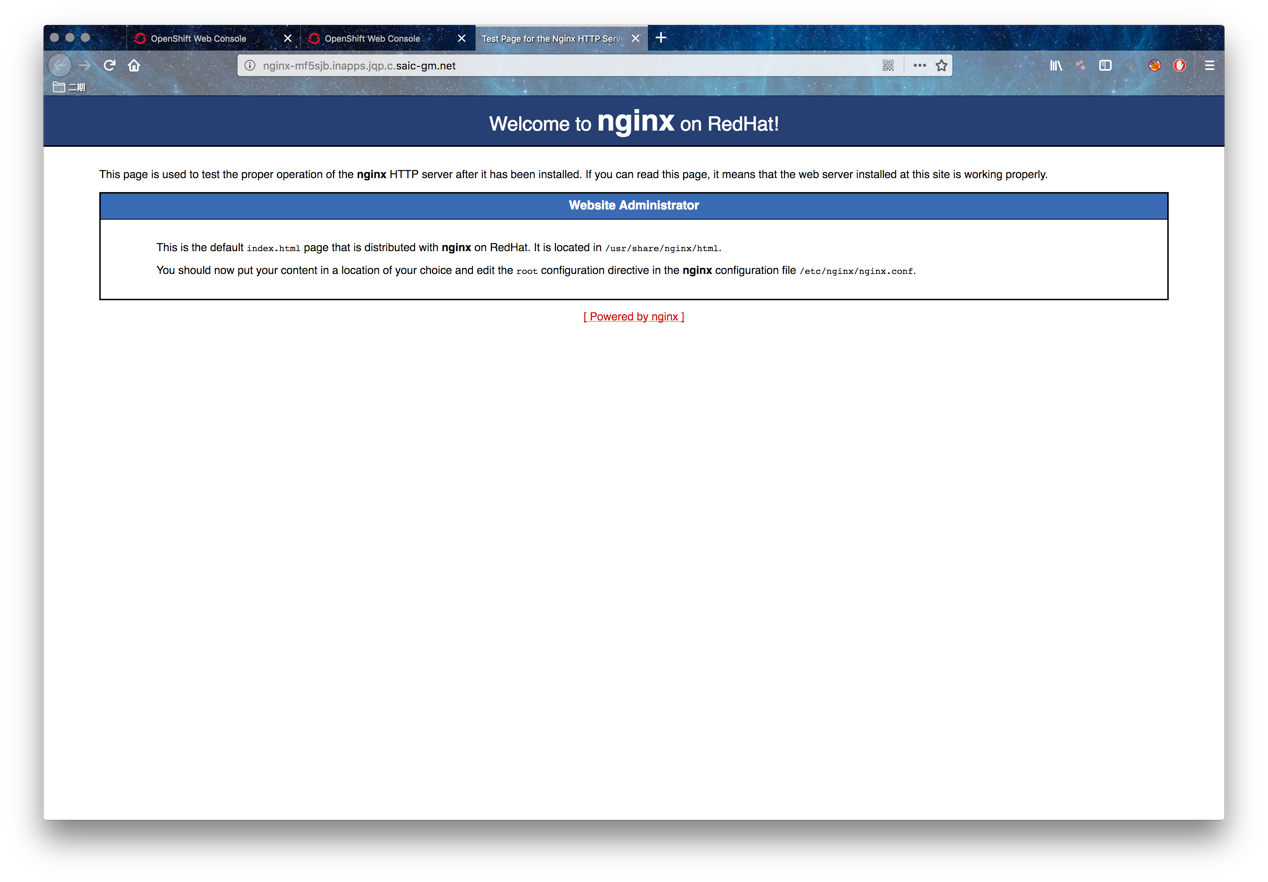
安装结束后，通过浏览器打开： https://console.jqp.c.saic-gm.net:8443/ 可以访问控制台。用户名redhat，密码passw0rd。

## 6.1 发布Nginx应用

登录Openshift控制台，选择并发布一个nginx应用进行功能验证，供参考：



部署完毕后，打开对应的Route，此处为：http://nginx-mf5sjb.inapps.jqp.c.saic-gm.net/，正确显示如下测试页面。



# 七、内外网router及内部registry部署

1.node节点打标签用于部署内网router

[root@sisvr02251 ~]# oc label node sishb00301.sgm.shanghaigm.com env=inrouter --overwrite=true

[root@sisvr02251 ~]# oc label node sishb00302.sgm.shanghaigm.com env=inrouter --overwrite=true

查看标签

[root@sisvr02251 ~]# oc get node --show-labels=true|grep inrouter

sishb00301.sgm.shanghaigm.com Ready 30d beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,env=inrouter,kubernetes.io/hostname=sishb00301.sgm.shanghaigm.com,logging-infra-fluentd=true,zone=cn-east-sh

sishb00302.sgm.shanghaigm.com Ready 30d beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,env=inrouter,kubernetes.io/hostname=sishb00302.sgm.shanghaigm.com,logging-infra-fluentd=true,zone=cn-east-sh

赋权限

[root@sisvr02251 ~]# oadm policy add-scc-to-user privileged -z inrouter

创建用于部署内网router

[root@sisvr02251 ~]# oc adm router inrouter --replicas=2 --service-account=inrouter --images='registry.jqp.c.saic-gm.net/openshift3/ose-${component}:${version}' --selector='env=inrouter'

2.node节点打标签用于部署公网router

[root@sisvr02251 ~]# oc label node sishb00303.sgm.shanghaigm.com env=exrouter --overwrite=true

[root@sisvr02251 ~]# oc label node sishb00304.sgm.shanghaigm.com env=exrouter --overwrite=true

查看标签

[root@sisvr02251 ~]# oc get node --show-labels=true|grep exrouter

sishb00303.sgm.shanghaigm.com Ready 30d beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,env=exrouter,kubernetes.io/hostname=sishb00303.sgm.shanghaigm.com,logging-infra-fluentd=true,zone=cn-east-sh

sishb00304.sgm.shanghaigm.com Ready 30d beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,env=exrouter,kubernetes.io/hostname=sishb00304.sgm.shanghaigm.com,logging-infra-fluentd=true,zone=cn-east-sh

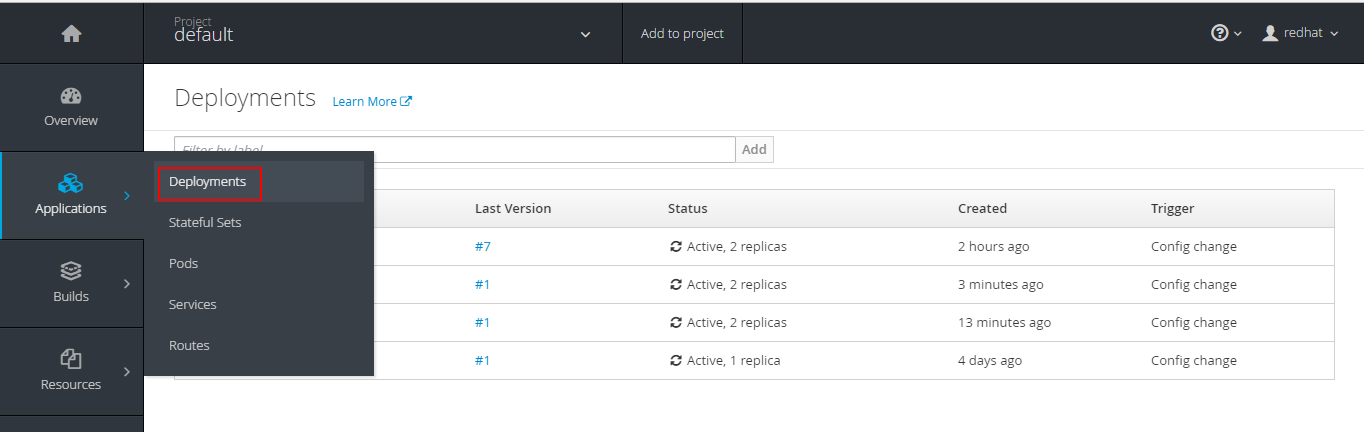
赋权限

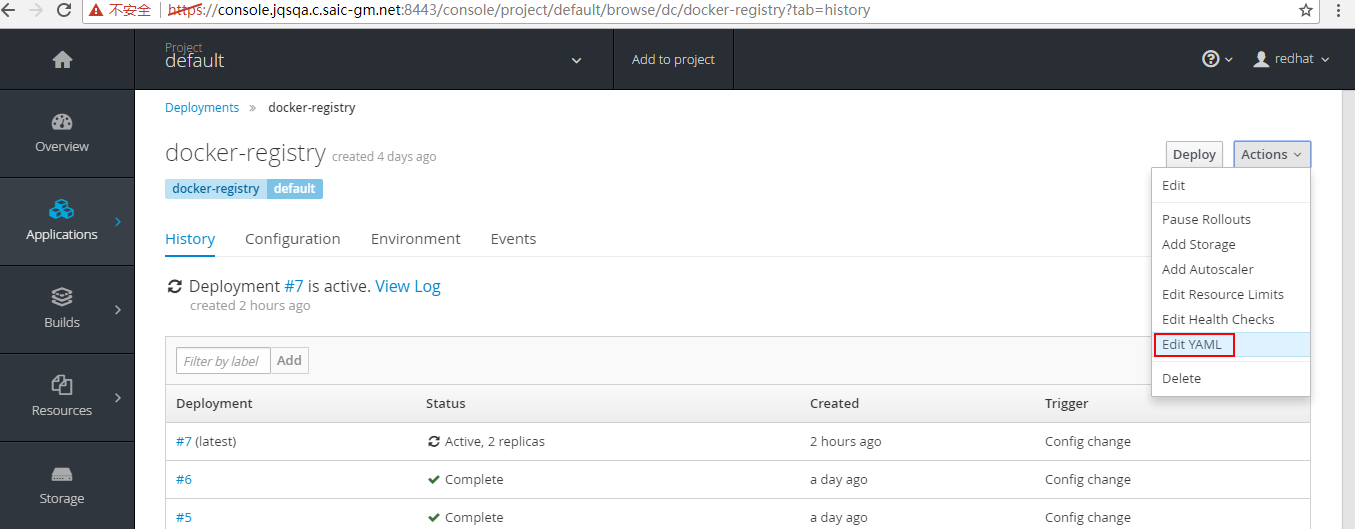
[root@sisvr02251 ~]# oadm policy add-scc-to-user privileged –z exrouter

创建用于部署公网router

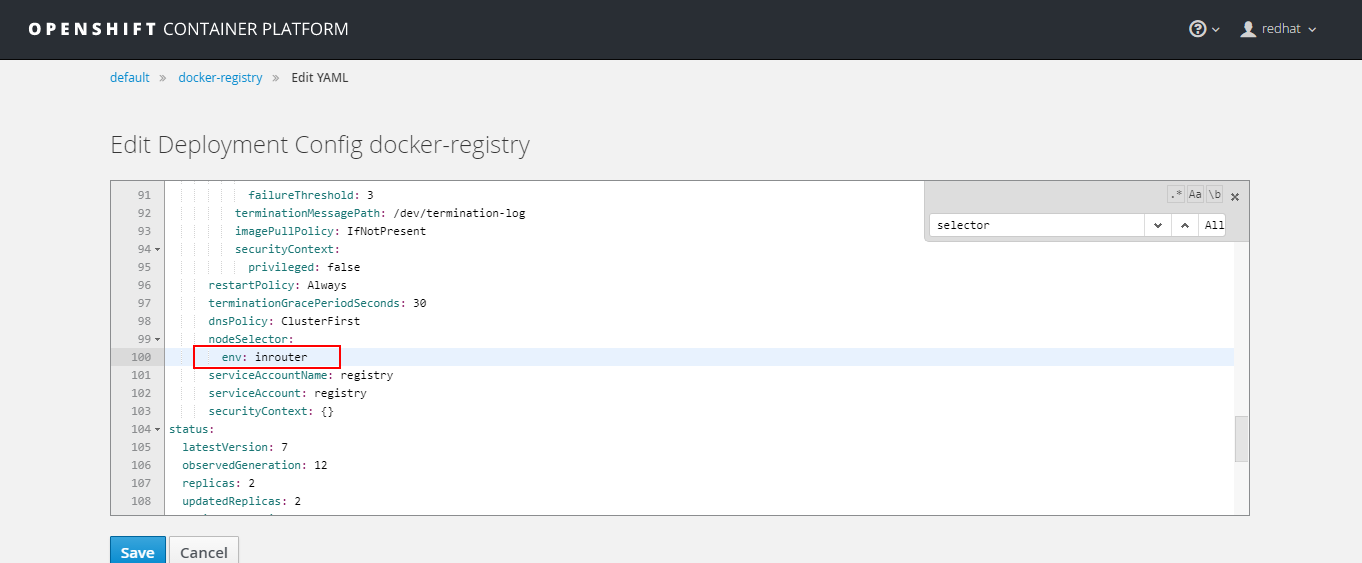
[root@sisvr02251 ~]# oc adm router exrouter --replicas=2 --service-account=exrouter --images='registry.jqp.c.saic-gm.net/openshift3/ose-${component}:${version}' --selector='env=exrouter'

3. 部署内部仓库在内网router





nodeSelector修改成部署inrouter的node节点标签

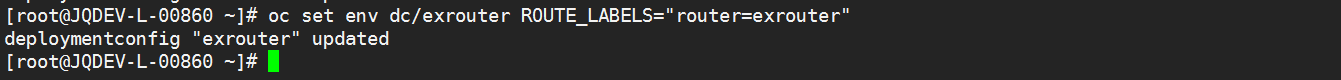


可选：验证 exrouter 发布应用

**给 exrouter 打标签**

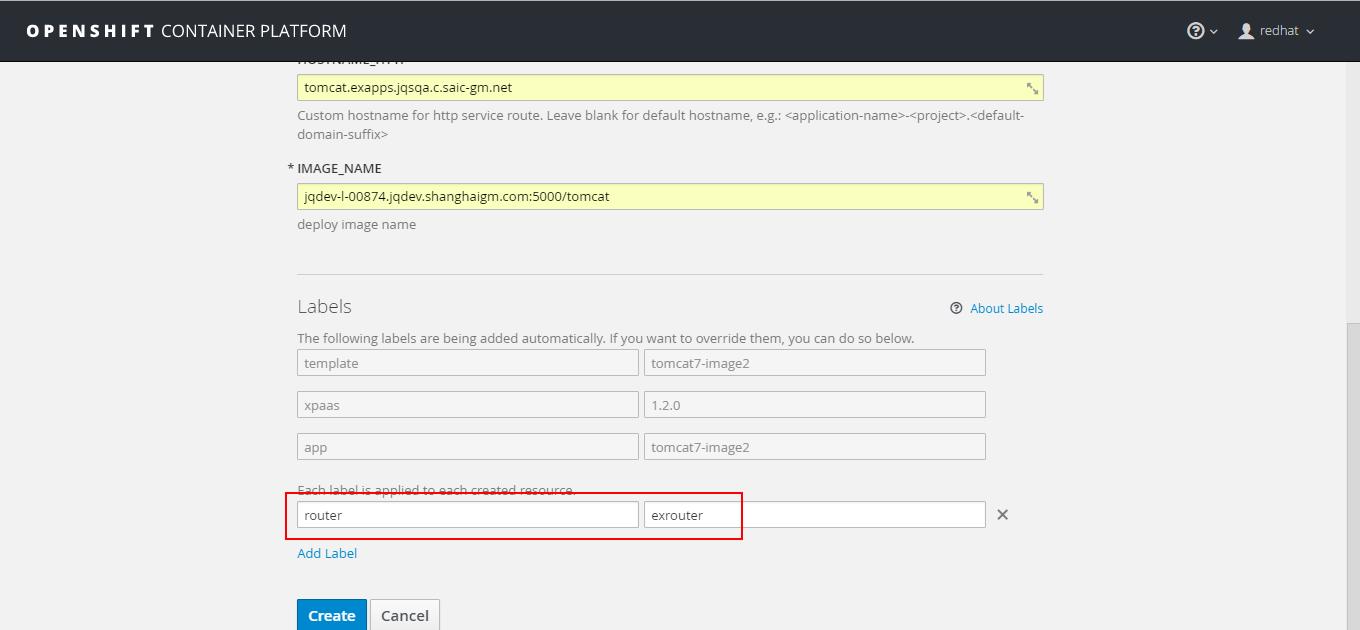
oc project default

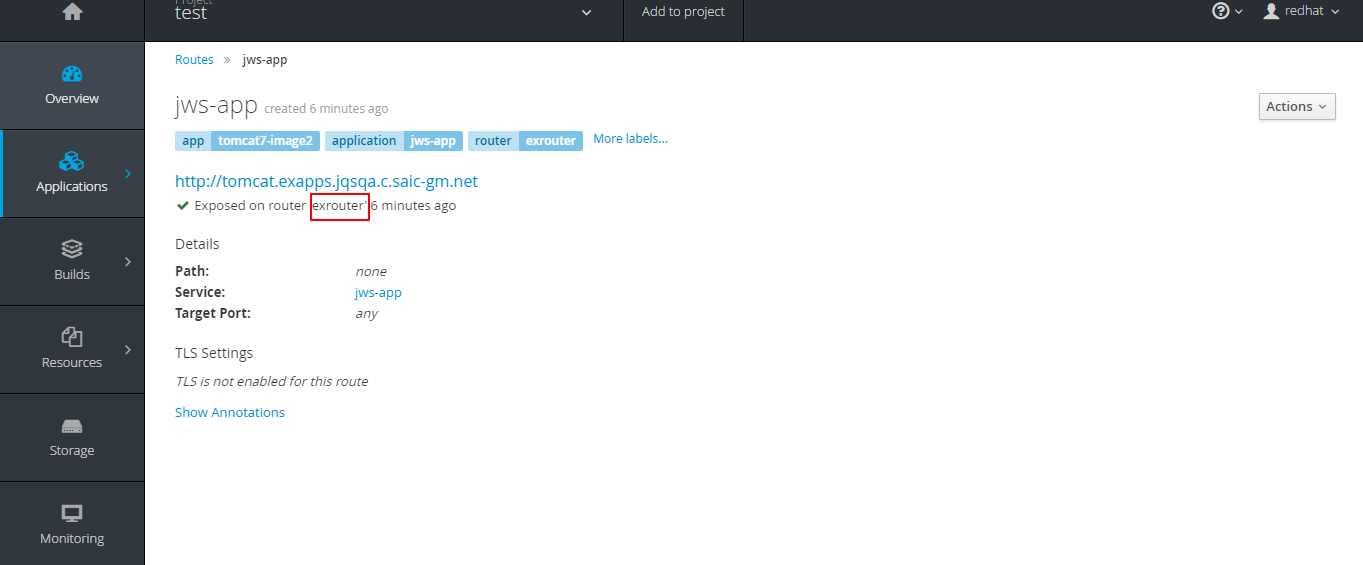
oc set env dc/exrouter ROUTE\_LABELS="router=exrouter"



oc set env dc/inrouter ROUTE\_LABELS="router=inrouter"

oc project test





# 八、内部 registry 持久化及外部 registry 挂载

## 1. 内部仓库持久化

**创建PV** (使用10.203.22.83上NFS存储路径为/ifs/docker1)

[root@sisvr02251 ~]# vim registry-pv.yaml

{

"apiVersion": "v1",

"kind": "PersistentVolume",

"metadata": {

"name": "registry-pv"

},

"spec": {

"capacity": {

"storage": "400Gi"

},

"accessModes": [ "ReadWriteOnce","ReadWriteMany" ],

"nfs": {

"path": "/ifs/docker1",

"server": "10.203.22.83"

}

}

}

[root@sisvr02251 ~]# oc create -f registry-pv.yaml

查看pv

[root@sisvr02251 ~]# oc get pv

NAME CAPACITY ACCESSMODES RECLAIMPOLICY STATUS CLAIM REASON AGE

harbormysql-pv 20Gi RWO,RWX Retain Bound default/harbormysql-pvc 15d

registry-pv 400Gi RWO,RWX Retain Bound default/registry-pvclaim 29d

**创建PVC**

[root@sisvr02251 ~]# oc project default

[root@sisvr02251 ~]# vim registry-pvc.yaml

{

"apiVersion": "v1",

"kind": "PersistentVolumeClaim",

"metadata": {

"name": "registry-pvclaim"

},

"spec": {

"accessModes": ["ReadWriteMany"],

"resources": {

"requests": {

"storage": "399Gi"

}

}

}

}

[root@sisvr02251 ~]# oc create registry-pvc.yaml

[root@sisvr02251 ~]# oc get pvc

查看仓库dc

[root@sisvr02251 ~]# oc get dc

NAME REVISION DESIRED CURRENT TRIGGERED BY

logging-curator 2 1 1 config

logging-es-fed24cop 9 1 1 config

logging-es-hofl7xzz 7 1 1 config

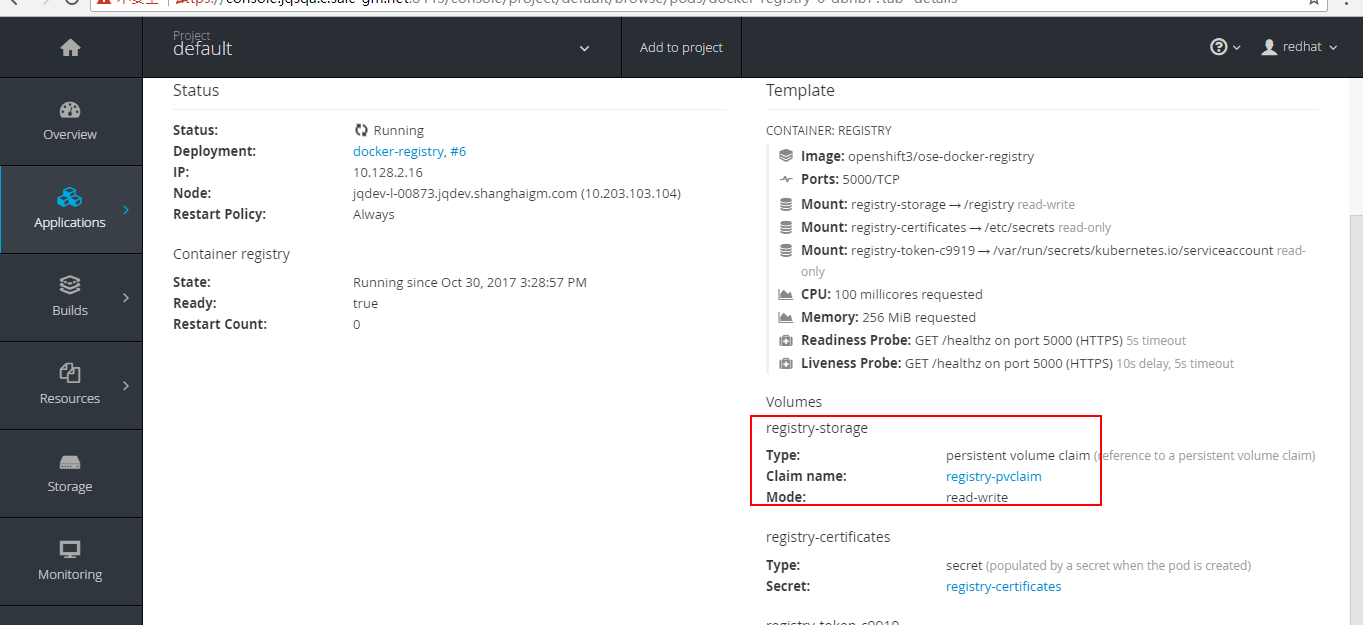
logging-es-obeg4f1b 8 1 1 config

logging-kibana 3 1 1 config

新建pvc

[root@sisvr02251 ~]# oc volumes dc docker-registry --add --overwrite -t pvc --claim-name=registry-pvclaim --name=registry-storage

[root@sisvr02251 ~]# oc get pvc



## 2. 挂载外部仓库

mkdir /root/backup

mv /var/lib/registry/\* /root/backup

2台registry执行下面命令

mount -t nfs 10.7.0.245:/stage2/dockerqa01 /var/lib/registry/

# 九、SSO配置

更改配置

所有的master上更改对应配置(10.203.65.53/54/55)，根据XXX的Oauth文件对应修改为如下，并将XXX的非公信证书加入到当前路径 (即ca:xxxCA.crt参数)

将/etc/origin/master/master-config.yaml中原有的oauthConfig:字段注释，并添加如下oauthConfig:字段

[root@sisvr02251 ~]# vim /etc/origin/master/master-config.yaml

….

assetConfig:

loggingPublicURL: https://kibana.inapps.jqp.c.saic-gm.net

logoutURL: https://idp.saic-gm.com/pkmslogout?filename=default.html&name=jqppaas&redirect=https://console.jqp.c.saic-gm.net:8443

masterPublicURL: https://console.jqp.c.saic-gm.net:8443

metricsPublicURL: https://hawkular-metrics.inapps.jqp.c.saic-gm.net/hawkular/metrics

publicURL: <https://console.jqp.c.saic-gm.net:8443/console/>

……

#oauthConfig:

# assetPublicURL: https://console.jqp.c.saic-gm.net:8443/console/

# grantConfig:

# method: auto

# identityProviders:

# - challenge: true

# login: true

# mappingMethod: claim

# name: htpasswd\_auth

# provider:

# apiVersion: v1

# file: /etc/origin/master/htpasswd/user

# kind: HTPasswdPasswordIdentityProvider

oauthConfig:

assetPublicURL: https://console.jqp.c.saic-gm.net:8443/console/

grantConfig:

method: auto

identityProviders:

- name: ocpoauth

challenge: false

login: true

mappingMethod: add

provider:

apiVersion: v1

kind: OpenIDIdentityProvider

clientID: 42954cdc-8640-45f7-b9c3-2fff6b41b768

clientSecret: 4e331548-f8c3-46c8-bad8-9205c6949eda

ca: SGMCA.crt

claims:

id:

- sub

urls:

authorize: https://idp.saic-gm.com/oauthweb/oauth/authorize

token: <https://saic-gm.com:7443/token.jsp>

…….

重启Openshift服务

在所有master上重启

[root@sisvr02251 ~]# systemctl restart atomic-openshift-{master-api,node}

验证跳转认证中心以获取权限

未登录XXX认证中心认证的用户访问平台 <https://console.jqp.c.saic-gm.net:8443> ：

# 十、部署日志持久化存储

基于日志的保存考虑，对es 节点所收集数据进行持久化处理，已提供SAN存储挂载目录 /data/ocp/esdata并配置好读写权限

## 对service account及es赋权

赋予logging项目挂载和编辑本地卷的权限，及编辑dc给es privileged的权限：

[root@sisvr02251 ~]# oadm policy add-scc-to-user privileged system:serviceaccount:logging:aggregated-logging-elasticsearch

[root@sisvr02251 ~]# oc project logging

[root@sisvr02251 ~]# oc get deploymentconfig --selector logging-infra=elasticsearch -o name

[root@sisvr02251 ~]# for dc in $(oc get deploymentconfig --selector logging-infra=elasticsearch -o name); do

oc scale $dc --replicas=0;

oc patch $dc\

-p '{"spec":{"template":{"spec":{"containers":[{"name":"elasticsearch","securityContext":{"privileged": true}}]}}}}'; done

## 配置es节点

[root@sisvr02251 ~]# oc label node sishb00501.sgm.shanghaigm.com logging-es-node=1

[root@sisvr02251 ~]# oc label node sishb00502.sgm.shanghaigm.com logging-es-node=2

[root@sisvr02251 ~]# oc label node sishb00503.sgm.shanghaigm.com logging-es-node=3

[root@sisvr02251 ~]# oc get nodes --shwo-labels | grep logging-es-node

[root@sisvr02251 ~]# oc get nodes --show-labels | grep logging-es-node

[root@sisvr02251 ~]# oc get dc

## 更新es的dc并重新部署

更新对应es节点的固定标签后，重新带上存储目录参数进行部署：

[root@sisvr02251 ~]# oc patch dc/logging-es-fed24cop -p '{"spec":{"template":{"spec":{"nodeSelector":{"logging-es-node":"1"}}}}}'

[root@sisvr02251 ~]# oc patch dc/logging-es-hofl7xzz -p '{"spec":{"template":{"spec":{"nodeSelector":{"logging-es-node":"2"}}}}}'

[root@sisvr02251 ~]# oc patch dc/logging-es-obeg4f1b -p '{"spec":{"template":{"spec":{"nodeSelector":{"logging-es-node":"3"}}}}}'

[root@sisvr02251 ~]# for dc in $(oc get deploymentconfig --selector logging-infra=elasticsearch -o name); do

oc set volume $dc

--add --overwrite --name=elasticsearch-storage

--type=hostPath --path=/data/ocp/esdata;

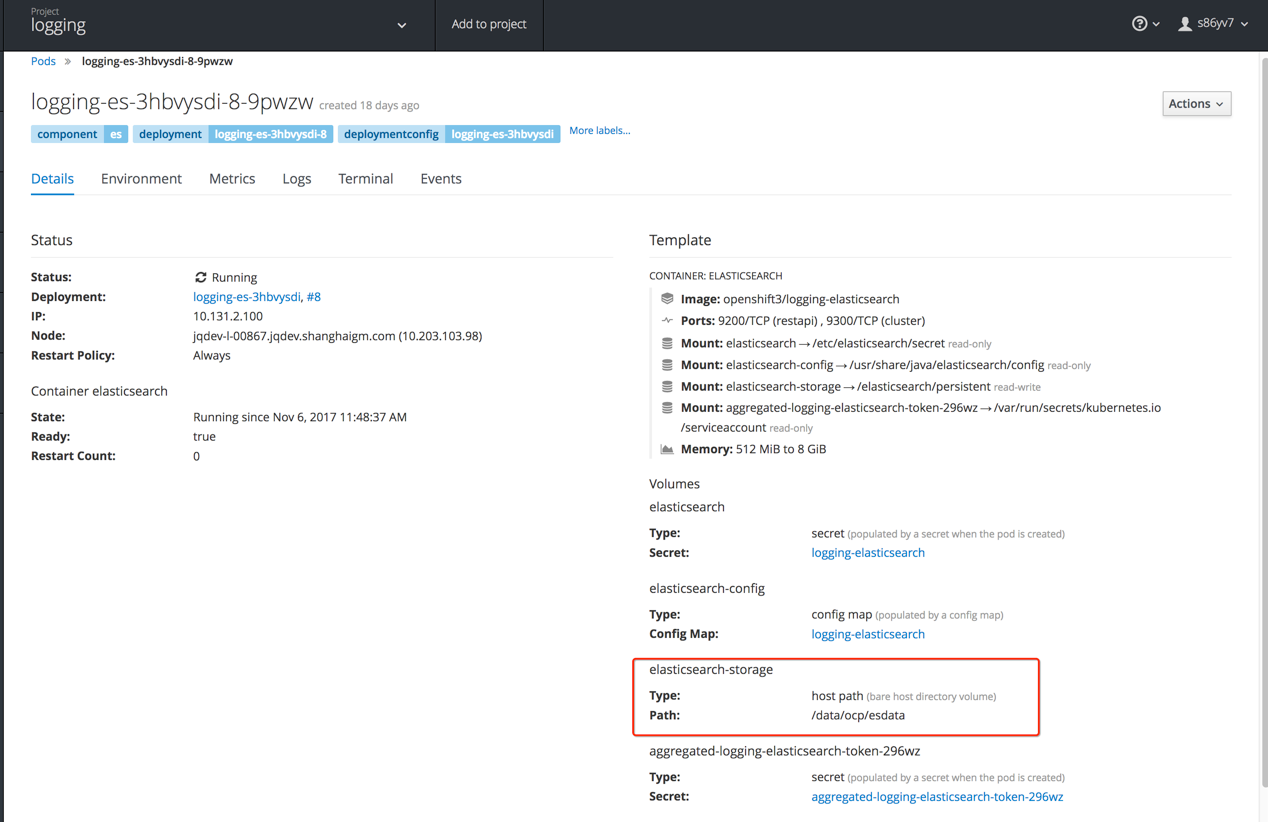
oc deploy --latest $dc;

oc scale $dc --replicas=1;

done

需求验证

部署好后在三个 es pod 上可以看到现在数据已经存储在挂载路径持久化



# 十一、部署定制化Webconsole

可能需要增加功能banner或者更换平台logo，现对QA环境进行配置为例。

准备如下三个文件，其中logo.png为要修改的logo图，

[root@sisvr02251 ~]# cd /etc/origin/master/

[root@sisvr02251 master]# ll | grep logo

-rwxr-xr-x. 1 root root 44460 Mar 14 14:02 logo.png

[root@sisvr02251 master]# ll | grep script

-rw-r--r--. 1 root root 422 Mar 14 14:03 script.js

[root@sisvr02251 master]# ll | grep style

-rw-r--r--. 1 root root 755 Mar 14 14:02 stylesheet.css

[root@sisvr02251 ~]# cd /etc/origin/master/

[root@sisvr02251 master]# cat script.js

(function() {

// Put your extension code here...

$('body').prepend('<div class="topnav" id="myTopnav" role="navigation"> <a href="https://console.jqpqa.c.saic-gm.net:8443">娉.?娴..</a> <a href="https://console.ygn.c.saic-gm.net:8443">?ㄩ.璺.ROD</a> <a class="active" href="https://console.jqp.c.saic-gm.net:8443">?.ˉPROD</a> <a href="https://console.jqppub.c.saic-gm.net:8443">?.ˉPUB</a> </div>');

}());

[root@sisvr02251 master]# cat stylesheet.css

/\* Add a black background color to the top navigation \*/

#header-logo {

background-image: url("../extensions/images/logo.png");

width: 560px;

height: 60px;

}

.topnav {

background-color: #333;

overflow: hidden;

}

/\* Style the links inside the navigation bar \*/

.topnav a {

float: left;

color: #f2f2f2;

text-align: center;

padding: 16px;

text-decoration: none;

font-size: 12px;

margin-top: 60px;

}

/\* Change the color of links on hover \*/

.topnav a:hover {

background-color: #ddd;

color: black;

}

/\* Add a color to the active/current link \*/

.topnav a.active {

background-color: #4CAF50;

color: white;

}

.middle {

margin-top: -60px;

}

.sidebar-left{

margin-top: -60px;

}

更改master配置文件

[root@sisvr02251 master]# pwd

/etc/origin/master

[root@sisvr02251 master]# vim master-config.yaml

assetConfig:

loggingPublicURL: https://kibana.inapps.jqp.c.saic-gm.net

masterPublicURL: https://console.jqp.c.saic-gm.net:8443

publicURL: https://console.jqp.c.saic-gm.net:8443/console/

servingInfo:

bindAddress: 0.0.0.0:8443

bindNetwork: tcp4

certFile: master.server.crt

loggingPublicURL: https://kibana.inapps.jqp.c.saic-gm.net

logoutURL: https://idp.saic-gm.com/pkmslogout?filename=default.html&name=jqppaas&redirect=https://console.jqp.c.saic-gm.net:8443

masterPublicURL: https://console.jqp.c.saic-gm.net:8443

metricsPublicURL: https://hawkular-metrics.inapps.jqp.c.saic-gm.net/hawkular/metrics

publicURL: https://console.jqp.c.saic-gm.net:8443/console/

servingInfo:

bindAddress: 0.0.0.0:8443

bindNetwork: tcp4

certFile: master.server.crt

clientCA: ""

keyFile: master.server.key

maxRequestsInFlight: 0

requestTimeoutSeconds: 0

extensions:

- name: images

sourceDirectory: /etc/origin/master/

extensionDevelopment: true

extensionScripts:

- /etc/origin/master/script.js

extensionStylesheets:

- /etc/origin/master/stylesheet.css

controllerConfig:

serviceServingCert:

signer:

certFile: service-signer.crt

keyFile: service-signer.key

….

重启服务

[root@sisvr02251 ~]# systemctl restart atomic-openshift-master-api

# 十二、部署默认应用模板及镜像，权限控制

根据需求对登录到平台的角色进行简单区分并配置权限，并新建projectmanager，projectviewer角色权限

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Counterpart in OCP-3.5** | **Description** | **Remark** |
| 管理员 | cluster-admin | A super-user that can perform any action in any project. When granted to a user within a local policy, they have full control over quota and every action on every resource in the project. |  |
| cluster-reader | cluster-reader | View access for the cluster |  |
| 项目经理  (projectmanager) | admin | A project manager. If used in a local binding, an admin user will have rights to view any resource in the project and modify any resource in the project except for quota. | Revoke the “delete project” privilege |
| 开发人员 | edit | A user that can modify most objects in a project, but does not have the power to view or modify roles or bindings. |  |
| 普通用户 | system:authenticated system:authenticated:oauth | Automatically associated with all users authenticated with an OAuth access token. | Revoke the “self-provisioner” privilege(create project) |

## 更改默认登录平台普通用户权限

去除普通用户创建 project 权限：

[root@sisvr02251 ~]# oadm policy remove-cluster-role-from-group self-provisioner system:authenticated

[root@sisvr02251 ~]# oadm policy remove-cluster-role-from-group self-provisioner system:authenticated:oauth

若需要加回权限角色给指定用户/组

[root@sisvr02251 ~]# oadm policy add-cluster-role-to-user self-provisioner <user>

## 新建projectmanager

基于默认角色admin上根据需求创建定制角色projectmanager，即去掉project delete的权限

[root@sisvr02251 ~]# oc get clusterrole admin -o yaml > projectmanager.yaml

[root@sisvr02251 ~]# vim projectmanager.yaml

…

metadata:

annotations:

openshift.io/description: A user that has edit rights within the project and can

change the project's membership.

creationTimestamp: 2017-09-30T04:19:43Z

name: projectmanager

resourceVersion: "20"

selfLink: /oapi/v1/clusterroles/projectmanager

uid: 95e0a581-a596-11e7-a6b1-005056af1d0d

…

attributeRestrictions: null

resources:

- projects

verbs:

- get

- patch

- update

…

[root@sisvr02251 ~]# oc create -f projectmanager.yaml

## 对用户授权

projectmanager, edit可以在项目内的membership里进行用户赋权，cluster-reader因为是集群角色，需要用oadm赋权，如下例子：

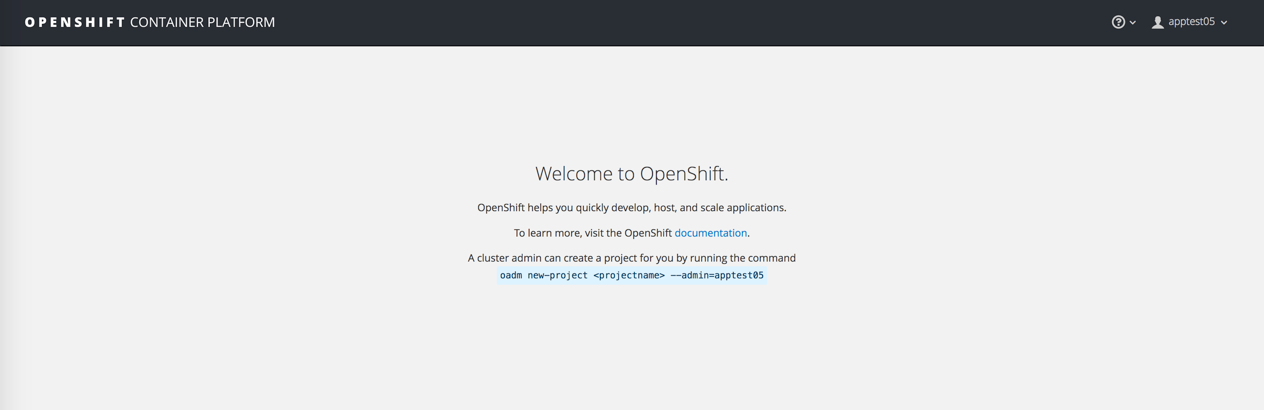
[root@sisvr02251 ~]# oadm policy add-cluster-role-to-user cluster-reader apptest03

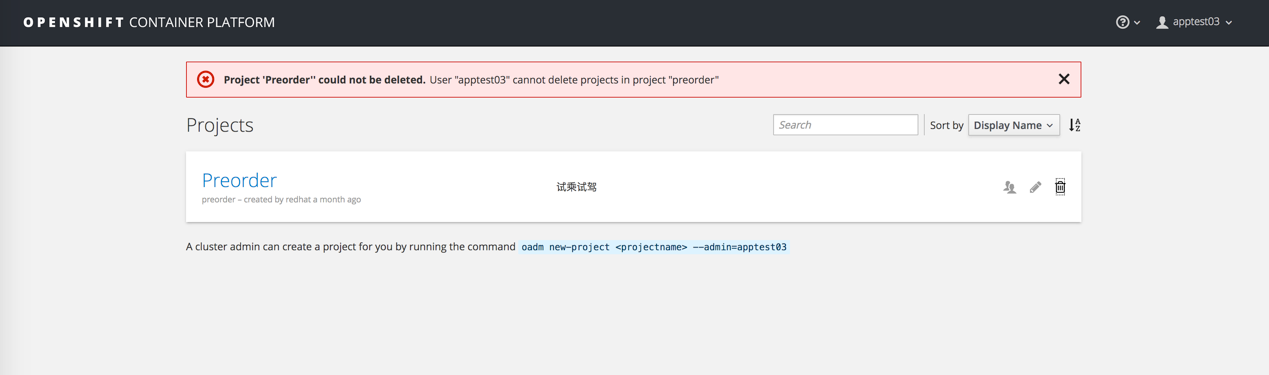
确认

[root@sisvr02251 ~]# oc get clusterrolebindings | grep apptest03

**需求验证**

普通用户登录后可以看到没有了新建project的权限，并且看不到所有project、user的信息。然后在具体project中选择用户赋予projectmanager角色权限后，做删除操作可以看到已经没有了删除权限。





# 十三、部署DMZ服务器

DMZ区域服务器为可连接外部网络的机器

根据需求在XXX的DMZ区域配置LB服务器进行请求处理转发至环境的外部router（exrouter）进行应用访问，下面以对QA环境的DMZ服务器进行配置为例。

安装基本组件

因网络原因将安装包传至DMZ服务器并安装haproxy

# yum install -y haproxy

# systemctl enable haproxy

# systemctl stop firewalld

# systemctl disable firewalld

准备haproxy配置文件

编辑haproxy.cfg，并上传对应证书

cat /etc/haproxy/haproxy.cfg

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

# Example configuration for a possible web application. See the

# full configuration options online.

#

# http://haproxy.1wt.eu/download/1.4/doc/configuration.txt

#

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

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

# Global settings

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

global

# to have these messages end up in /var/log/haproxy.log you will

# need to:

#

# 1) configure syslog to accept network log events. This is done

# by adding the '-r' option to the SYSLOGD\_OPTIONS in

# /etc/sysconfig/syslog

#

# 2) configure local2 events to go to the /var/log/haproxy.log

# file. A line like the following can be added to

# /etc/sysconfig/syslog

#

# local2.\* /var/log/haproxy.log

#

log 127.0.0.1 local2

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid

maxconn 4000

user haproxy

group haproxy

daemon

# turn on stats unix socket

stats socket /var/lib/haproxy/stats

tune.ssl.default-dh-param 2048

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

# 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 1m

timeout server 1m

timeout http-keep-alive 10s

timeout check 10s

maxconn 3000

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

# main frontend which proxys to the backends

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

frontend http\_free

bind \*:80

mode http

redirect scheme https if !{ ssl\_fc }

bind \*:443 ssl crt /etc/haproxy/exapps.jqpqa.c.saic-gm.net.cer

# acl helloworld hdr\_beg(host) -i helloworld.exapps.jqpqa.c.saic-gm.net

acl helloworld hdr\_beg(host) -f /etc/haproxy/whitelist.lst

use\_backend http\_exapps if helloworld

default\_backend http\_black

backend http\_exapps

mode http

server router 10.203.103.239:80 check

backend http\_black

mode http

errorfile 503 /etc/haproxy/401.http

# vim /etc/haproxy/401.http

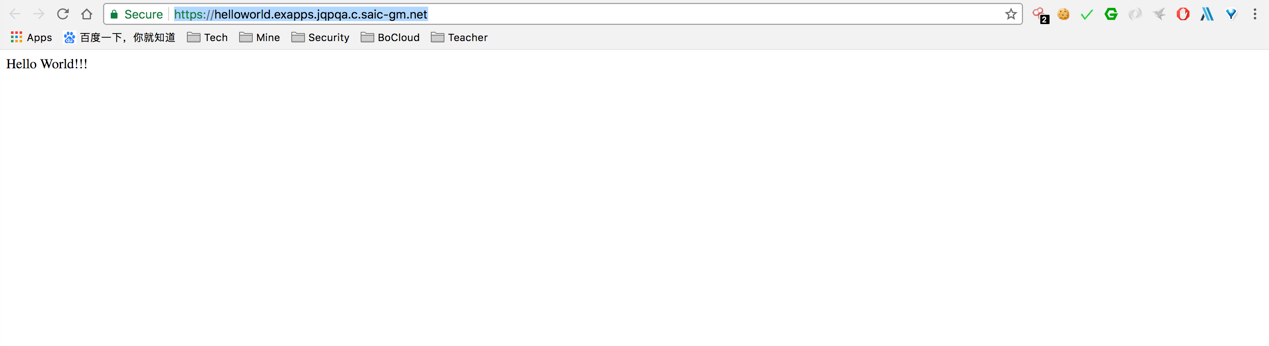
This site is not allowed.

# chmod 755 401.http

#systemctl restart haproxy

需求验证

外网对发布的应用进行访问确认反向代理生效：



## 更新白名单

配置

2台服务器免密互信

[root@shctygvl0240 ~]# ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

5d:f2:e5:ed:5c:dd:18:ad:98:80:f8:62:1d:b7:1b:00 root@shctygvl0240

The key's randomart image is:

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

|      E          |

|       o .     . |

|      . + + . o .|

|       o = \* = =o|

|      o S + + + =|

|     . .   o   o.|

|          .     o|

|                 |

|                 |

+-----------------+

在/root路径下创建updateWhitelist.sh文件，更新其中SERVERLIST部分的地址，替换为实际DMZ服务器的IP地址。

vim updateWhitelist.sh

#!/bin/sh

SERVERLIST='

10.200.5.158

10.200.5.159

'

TIMESTAMP=`date +%Y%m%d%H%M%S`

cd /tmp

scp `echo $SERVERLIST | awk '{print $1}'`:/etc/haproxy/whitelist.lst ${PWD}/whitelist.lst

cp whitelist.lst whitelist.lst.orig

vim whitelist.lst

clear

echo "##### Changed #####"

diff whitelist.lst whitelist.lst.orig

echo "###################"

printf "Sync whitelist and restart haproxy?[y/N]:"

read -n1 REPLY

if [ "${REPLY}" != "y" ]; then

echo "Answer: "${REPLY}". Existing..."

else

for node in ${SERVERLIST}

do

ssh $node "cp /etc/haproxy/whitelist.lst /etc/haproxy/whitelist.lst.${TIMESTAMP}";

   scp ${PWD}/whitelist.lst ${node}:/etc/haproxy/whitelist.lst;

   ssh $node 'systemctl restart haproxy';

done

fi

rm whitelist.lst whitelist.lst.orig

执行sh updateWhitelist.sh，在打开的vi编辑器中每行填入一条白名单记录，:wq保存后根据提示决定是否更新白名单。

## 多证书配置方法

准备对应的证书及配置文件

# cat exapps.jqpreh.c.saic-gm.net.key >> exapps.jqpreh.c.saic-gm.net.cer

# vim /etc/haproxy/haproxy.cfg

// 1. 加上对应域名证书绑定；2. 加上对应 ssl 跳转规则；3. 加上对应 跳转的 backend。

frontend http\_free

bind \*:80

mode http

redirect scheme https if !{ ssl\_fc }

bind \*:443 ssl crt /etc/haproxy/exapps.jqpreh.c.saic-gm.net.cer

# acl helloworld hdr\_beg(host) -i helloworld.exapps.jqppub.c.saic-gm.net

acl helloworld hdr\_beg(host) -f /etc/haproxy/whitelist.lst

use\_backend http\_exapps\_reh if helloworld

use\_backend http\_exapps\_reh if { ssl\_fc\_sni reh.exapps.jqpreh.c.saic-gm.net }

default\_backend http\_black

backend http\_exapps\_reh

mode http

server router 10.203.101.132:80 check

backend http\_black

mode http

errorfile 503 /etc/haproxy/401.http

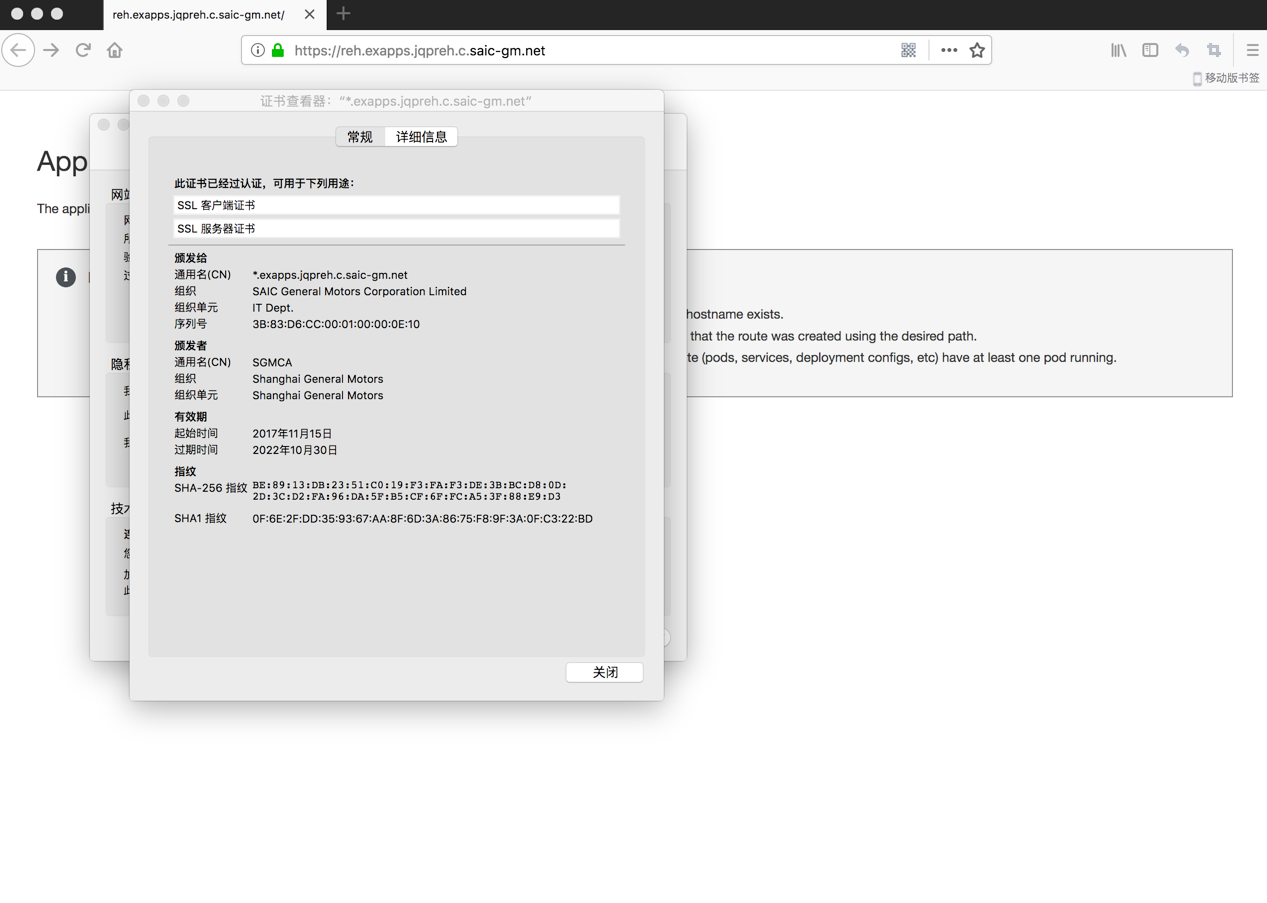
# chmod 755 401.http

# systemctl restart haproxy

本地 hosts 文件加上对应解析

|  |
| --- |
| 10.203.101.140 reh.exapps.jqpreh.c.saic-gm.net |

需求验证



# 十四、增加Node节点

所增加的节点（node）的资源与系统结构应与要加入的环境对应角色的配置完全一致，包括但不限于CPU，内存，存储，网络等

配置SELinux

**系统中NetworkManager和selinux两个服务必须打开**

# systemctl enable NetworkManager

# systemctl restart NetworkManager

# /bin/sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config

# setenforce 0

配置Yum源，并安装依赖程序

cat << EOF >/etc/yum.repos.d/ose35.repo

[ose35]

name=OpenShift3.5

baseurl=ftp://10.203.101.133:81/repo

enabled=1

gpgcheck=0

EOF

安装依赖程序

# yum -y install libcgroup sysstat bridge-utils gcc bind-utils libaio expect ntp libstdc++-devel curl dnsmasq autoconf automake gcc-c++ git libtool xinetd check openssl-devel zlib-devel krb5-devel net-tools libxml2 dos2unix wget telnet tar gzip unzip yum-utils ftp bash-completion lrzsz gpm-libs vim

配置Iptables

Docker必须使用iptables，不能用系统自带的firewalld，如下为配置

# systemctl stop firewalld

# systemctl disable firewalld;

# yum install iptables-services -y;

# systemctl enable iptables

# iptables -A INPUT -s 192.0.0.0/8 -j ACCEPT

# iptables -A INPUT -s 172.0.0.0/8 -j ACCEPT

# iptables -A INPUT -p tcp --sport 22 -j ACCEPT

# iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# iptables -A INPUT -p tcp --sport 21 -j ACCEPT

# iptables -A INPUT -p tcp --dport 21 -j ACCEPT

# iptables -A INPUT -p tcp --sport 81 -j ACCEPT

# iptables -A INPUT -p tcp --dport 81 -j ACCEPT

# iptables -A INPUT -p udp --sport 20 -j ACCEPT

# iptables -A INPUT -p udp --dport 20 -j ACCEPT

# iptables -A INPUT -p udp --sport 123 -j ACCEPT

# iptables -A INPUT -p udp --dport 123 -j ACCEPT

# iptables -A INPUT -p udp --sport 53 -j ACCEPT

# iptables -A INPUT -p udp --dport 53 -j ACCEPT

# iptables -P INPUT ACCEPT; iptables -P OUTPUT ACCEPT; iptables -P FORWARD ACCEPT

# iptables-save |grep -v DROP |grep -v REJECT >/etc/sysconfig/iptables

# systemctl restart iptables

配置DNS域名解析

重启服务后随机找个服务器做如下的验证

# systemctl restart network

# ping sisivr02253.jqp.shanghaigm.com -c 3;

SSH信任

在master上把它的key复制到节点

# ssh-copy-id -i ~/.ssh/id\_rsa.pub sisvr02260.jqp.shanghaigm.com

然后测试通过ssh server-name登陆不需要输入密码

# ssh sisivr02260.jqp.shanghaigm.com uname –a

配置NTP

在新添加的节点上配置NTP，使其与原有Openshift环境时间同步

# yum install -y ntp

# sed -i '/^server.\*/d' /etc/ntp.conf

# echo "server 10.203.238.254 iburst" >>/etc/ntp.conf

# systemctl enable ntpd

# systemctl restart ntpd

验证，在其他所有节点执行： timedatectl，当列表出现表示同步成功

# timedatectl

NTP enabled: yes

NTP synchronized: ye

安装Docker

# yum -y install docker

# systemctl enable docker

配置docker选项，registry.jqp.c.saic-gm.net为registry私有仓库的域名

# cp /etc/sysconfig/docker /etc/sysconfig/docker.bak.$(date "+%Y%m%d%H%M%S")

# sed -i s/".\*OPTIONS=.\*"/"OPTIONS=' --selinux-enabled --log-driver=journal --insecure-registry 172.30.0.0\/16'"/g /etc/sysconfig/docker

# sed -i 's/registry.access.redhat.com/registry.jqp.c.saic-gm.net/g' /etc/sysconfig/docker

# echo "BLOCK\_REGISTRY='--block-registry public --block-registry registry.

access.redhat.com'">> /etc/sysconfig/docker

[root@sisvr02260 ~]# vim /etc/sysconfig/docker

ADD\_REGISTRY='--add-registry registry.jqp.c.saic-gm.net --add-registry registry.access.redhat.com'

BLOCK\_REGISTRY='--block-registry public --block-registry registry.acce

ss.redhat.com'

INSECURE\_REGISTRY='--insecure-registry registry.jqp.c.saic-gm.net'

配置Docker存储

# systemctl stop docker

# rm -rf /var/lib/docker/\*

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

虚拟机：

cat <<EOF >/etc/sysconfig/docker-storage-setup

DEVS=/dev/sdb

VG=docker-vg

SETUP\_LVM\_THIN\_POOL=yes

EOF

物理机：

//有外挂存储的机子，如3个 es 节点，需要更改 filter

# vim /etc/lvm/lvm.conf

…

filter = [ "a|/dev/sda[0-9]\*|", "a|/dev/emcpower.\*|", "r|.\*|" ]

cat <<EOF >/etc/sysconfig/docker-storage-setup

VG=vg01

SETUP\_LVM\_THIN\_POOL=yes

EOF

# docker-storage-setup

加入对应 registry 域名证书，重新启动docker

# mkdir -p /etc/docker/certs.d/registry.jqp.c.saic-gm.net/registry.crt

# cp SGMCA2048.crt/etc/docker/certs.d/registry.jqp.c.saic-gm.net/registry.crt

# systemctl restart docker

**配置Ansible文件**

在master上的/etc/ansible/hosts加入如下高亮内容

[root@sisvr02251 ~]# cat /etc/ansible/hosts

# Logging

openshift\_hosted\_logging\_deploy=true

openshift\_logging\_image\_prefix=registry.jqp.c.saic-gm.net/openshift3/

openshift\_logging\_es\_cluster\_size=3

openshift\_logging\_image\_version=3.5.0

openshift\_logging\_es\_nodeselector={"region":"infra"}

# enable ntp on masters to ensure proper failover

openshift\_clock\_enabled=true

# Router

openshift\_hosted\_router\_selector="env=inrouter"

openshift\_hosted\_router\_replicas=2

# Registry

openshift\_hosted\_registry\_selector="env=inrouter"

openshift\_hosted\_registry\_replicas=2

[OSEv3:children]

masters

nodes

etcd

new\_nodes

[masters]

sisvr02251.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02252.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02253.sgm.shanghaigm.com host\_zone=cn-east-sh

# host group for etcd

[etcd]

sisvr02251.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02252.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02253.sgm.shanghaigm.com host\_zone=cn-east-sh

# host group for nodes, includes region info

[nodes]

## master

sisvr02251.sgm.shanghaigm.com openshift\_hostname=sisvr02251.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

sisvr02252.sgm.shanghaigm.com openshift\_hostname=sisvr02252.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

sisvr02253.sgm.shanghaigm.com openshift\_hostname=sisvr02253.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

## These are regular nodes

sishb00301.sgm.shanghaigm.com openshift\_hostname=sishb00301.sgm.shanghaigm.com openshift\_node\_labels="{'env':'inrouter', 'zone': 'cn-east-sh'}"

sishb00302.sgm.shanghaigm.com openshift\_hostname=sishb00302.sgm.shanghaigm.com openshift\_node\_labels="{'env':'inrouter', 'zone': 'cn-east-sh'}"

sishb00303.sgm.shanghaigm.com openshift\_hostname=sishb00303.sgm.shanghaigm.com openshift\_node\_labels="{'env':'exrouter', 'zone': 'cn-east-sh'}"

sishb00304.sgm.shanghaigm.com openshift\_hostname=sishb00304.sgm.shanghaigm.com openshift\_node\_labels="{'env':'exrouter', 'zone': 'cn-east-sh'}"

## infra nodes

sishb00501.sgm.shanghaigm.com openshift\_hostname=sishb00501.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

sishb00502.sgm.shanghaigm.com openshift\_hostname=sishb00502.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

sishb00503.sgm.shanghaigm.com openshift\_hostname=sishb00503.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

## compute nodes

sishb00305.sgm.shanghaigm.com openshift\_hostname=sishb00305.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00401.sgm.shanghaigm.com openshift\_hostname=sishb00401.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00402.sgm.shanghaigm.com openshift\_hostname=sishb00402.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00403.sgm.shanghaigm.com openshift\_hostname=sishb00403.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

new\_nodes

sisvr02260.sgm.shanghaigm.com openshift\_hostname=sisvr02260.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

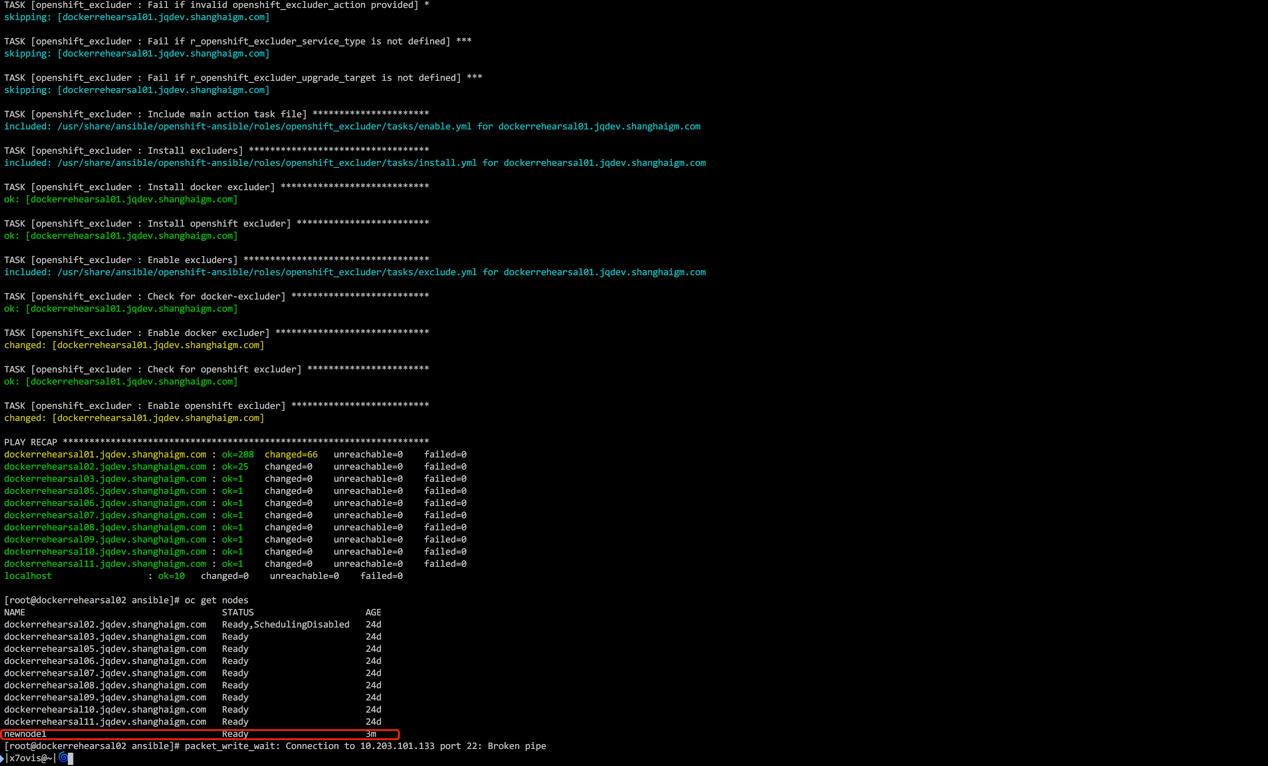
执行ansible脚本

在master1（10.203.65.53）上执行扩展脚本

# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/

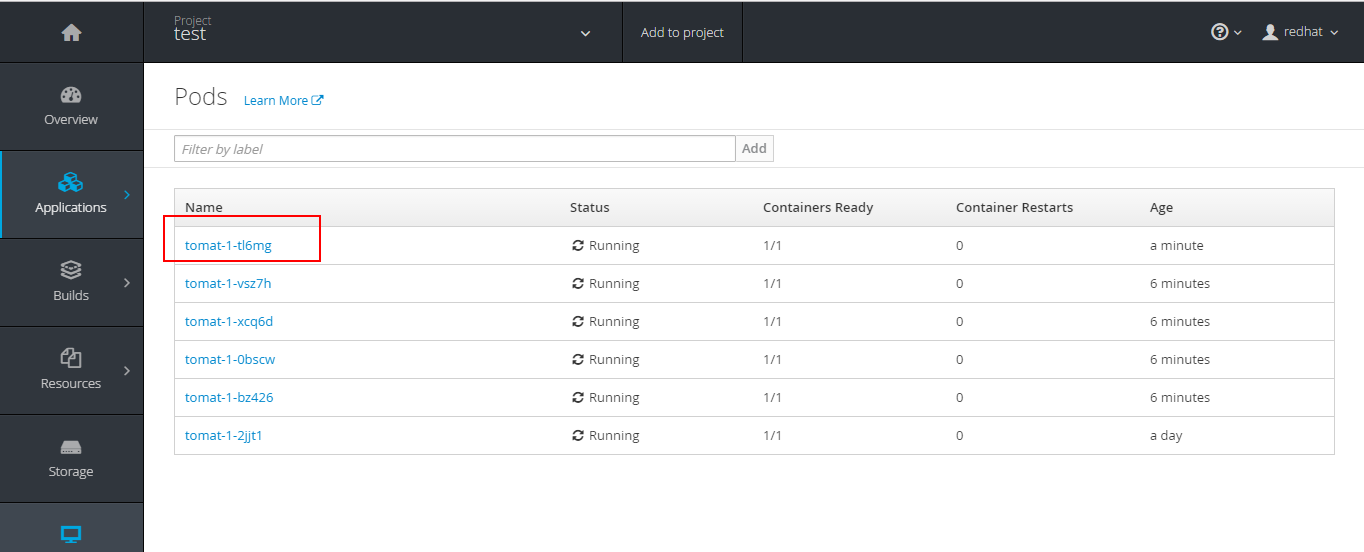
openshift-node/scaleup.yml

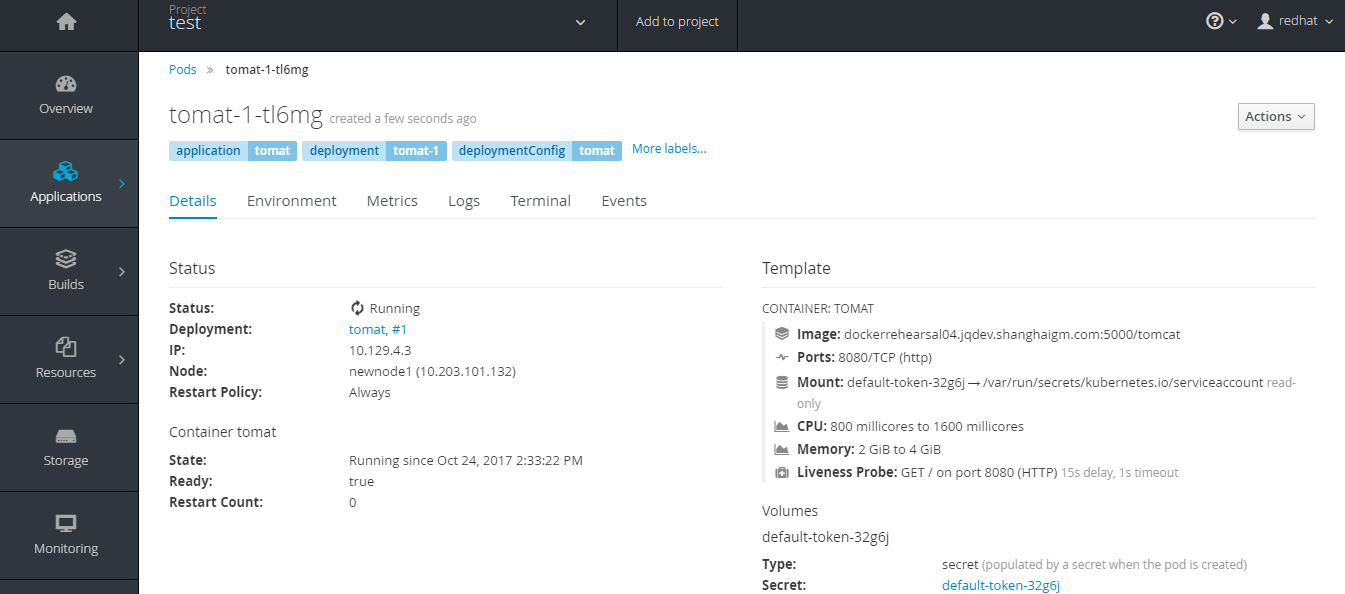
安装过程5~10分钟，最后结果应该全部节点failed=0。



验证新节点功能正常

对现存应用进行 scaleup，确认可以正常运行在新加入的 node上





再次更新ansible hosts文件

将新加入的节点加入 [nodes] 块中，但扔需留着 [new\_nodes] 标签，

[root@sisvr02251 ~]# cat /etc/ansible/hosts

# Logging

openshift\_hosted\_logging\_deploy=true

openshift\_logging\_image\_prefix=registry.jqp.c.saic-gm.net/openshift3/

openshift\_logging\_es\_cluster\_size=3

openshift\_logging\_image\_version=3.5.0

openshift\_logging\_es\_nodeselector={"region":"infra"}

# enable ntp on masters to ensure proper failover

openshift\_clock\_enabled=true

# Router

openshift\_hosted\_router\_selector="env=inrouter"

openshift\_hosted\_router\_replicas=2

# Registry

openshift\_hosted\_registry\_selector="env=inrouter"

openshift\_hosted\_registry\_replicas=2

[OSEv3:children]

masters

nodes

etcd

new\_nodes

[masters]

sisvr02251.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02252.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02253.sgm.shanghaigm.com host\_zone=cn-east-sh

# host group for etcd

[etcd]

sisvr02251.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02252.sgm.shanghaigm.com host\_zone=cn-east-sh

sisvr02253.sgm.shanghaigm.com host\_zone=cn-east-sh

# host group for nodes, includes region info

[nodes]

## master

sisvr02251.sgm.shanghaigm.com openshift\_hostname=sisvr02251.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

sisvr02252.sgm.shanghaigm.com openshift\_hostname=sisvr02252.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

sisvr02253.sgm.shanghaigm.com openshift\_hostname=sisvr02253.sgm.shanghaigm.com openshift\_node\_labels="{'openshift\_schedulable':'False', 'zone': 'cn-east-sh'}"

## These are regular nodes

sishb00301.sgm.shanghaigm.com openshift\_hostname=sishb00301.sgm.shanghaigm.com openshift\_node\_labels="{'env':'inrouter', 'zone': 'cn-east-sh'}"

sishb00302.sgm.shanghaigm.com openshift\_hostname=sishb00302.sgm.shanghaigm.com openshift\_node\_labels="{'env':'inrouter', 'zone': 'cn-east-sh'}"

sishb00303.sgm.shanghaigm.com openshift\_hostname=sishb00303.sgm.shanghaigm.com openshift\_node\_labels="{'env':'exrouter', 'zone': 'cn-east-sh'}"

sishb00304.sgm.shanghaigm.com openshift\_hostname=sishb00304.sgm.shanghaigm.com openshift\_node\_labels="{'env':'exrouter', 'zone': 'cn-east-sh'}"

## infra nodes

sishb00501.sgm.shanghaigm.com openshift\_hostname=sishb00501.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

sishb00502.sgm.shanghaigm.com openshift\_hostname=sishb00502.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

sishb00503.sgm.shanghaigm.com openshift\_hostname=sishb00503.sgm.shanghaigm.com openshift\_node\_labels="{'region':'infra', 'zone': 'cn-east-sh'}"

## compute nodes

sishb00305.sgm.shanghaigm.com openshift\_hostname=sishb00305.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00401.sgm.shanghaigm.com openshift\_hostname=sishb00401.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00402.sgm.shanghaigm.com openshift\_hostname=sishb00402.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sishb00403.sgm.shanghaigm.com openshift\_hostname=sishb00403.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

sisvr02260.sgm.shanghaigm.com openshift\_hostname=sisvr02260.sgm.shanghaigm.com openshift\_node\_labels="{'region': 'primary','zone': 'cn-east-sh'}"

new\_nodes

# 十五、节点高可用部署

根据需求对Openshift的关键角色节点做服务器之间的高可用，通过配置 VIP + Keepalived 来保证节点所在服务器的高可用，下面对 QA环境的外部registry（10.203.103.105, 106, VIP为10.203.103.239）进行配置为例

10.203.103.105,/106为registry节点

安装必要组件及基本环境

两台registry节点10.203.103.105/106安装

# yum install -y ipvsadm keepalived

# systemctl enable keepalived

配置基本环境

# systemctl enable NetworkManager

# systemctl restart NetworkManager

# /bin/sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config

# setenforce 0

# systemctl stop firewalld

# systemctl disable firewalld;

# yum install iptables-services -y;

# systemctl enable iptables

# iptables -A INPUT -s 192.0.0.0/8 -j ACCEPT

# iptables -A INPUT -s 172.0.0.0/8 -j ACCEPT

# iptables -A INPUT -p tcp --sport 22 -j ACCEPT

# iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# iptables -A INPUT -p tcp --sport 21 -j ACCEPT

# iptables -A INPUT -p tcp --dport 21 -j ACCEPT

# iptables -A INPUT -p tcp --sport 81 -j ACCEPT

# iptables -A INPUT -p tcp --dport 81 -j ACCEPT

# iptables -A INPUT -p tcp --sport 49942 -j ACCEPT

# iptables -A INPUT -p tcp --dport 49942 -j ACCEPT

# iptables -A INPUT -p udp --sport 20 -j ACCEPT

# iptables -A INPUT -p udp --dport 20 -j ACCEPT

# iptables -A INPUT -p udp --sport 123 -j ACCEPT

# iptables -A INPUT -p udp --dport 123 -j ACCEPT

# iptables -A INPUT -p udp --sport 53 -j ACCEPT

# iptables -A INPUT -p udp --dport 53 -j ACCEPT

# iptables -P INPUT ACCEPT; iptables -P OUTPUT ACCEPT; iptables -P FORWARD ACCEPT

# iptables -I INPUT -i ens192 -d 224.0.0.0/8 -p vrrp -j ACCEPT

# iptables -I OUTPUT -o ens192 -d 224.0.0.0/8 -p vrrp -j ACCEPT

# iptables-save |grep -v DROP |grep -v REJECT >/etc/sysconfig/iptables

# systemctl restart iptables

准备配置文件及脚本

编辑keepalived.conf，在master（10.203.103.105）上

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

! Configuration File for keepalived

global\_defs {

notification\_email {

}

router\_id registry\_01

}

vrrp\_script chk\_http\_port {

script "/opt/registry\_check.sh"

interval 2

weight 2

}

vrrp\_instance VI\_1 {

state MASTER

interface ens192

virtual\_router\_id 51

priority 120

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 1111

}

track\_script {

chk\_http\_port

}

virtual\_ipaddress {

10.203.103.239

}

}

EOF

BACKUP（10.203.103.106）上的配置

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

! Configuration File for keepalived

global\_defs {

notification\_email {

}

router\_id registry\_02

}

vrrp\_script chk\_http\_port {

script "/opt/registry\_check.sh"

interval 2

weight 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens192

virtual\_router\_id 51

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 1111

}

track\_script {

chk\_http\_port

}

virtual\_ipaddress {

10.203.103.239

}

}

EOF

范例脚本

# cat /opt/registry\_check.sh

#!/bin/bash

x=`systemctl status docker-distribution | grep running | wc -l`

echo $x

if [ $x -eq 0 ]; then

sleep 3

if [ `systemctl status docker-distribution | grep running | wc -l` -eq 0 ]; then

systemctl stop keepalived

fi

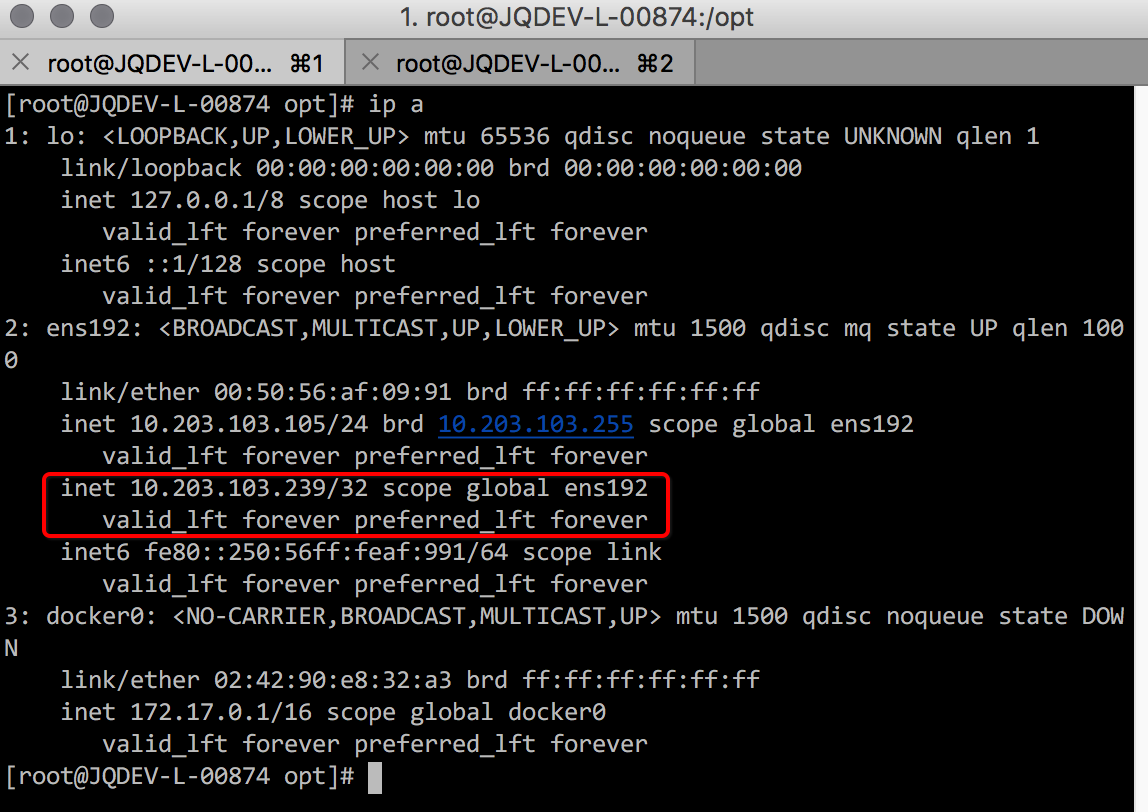
fi

#chmod +x registry\_check.sh

需求验证

验证VIP生效

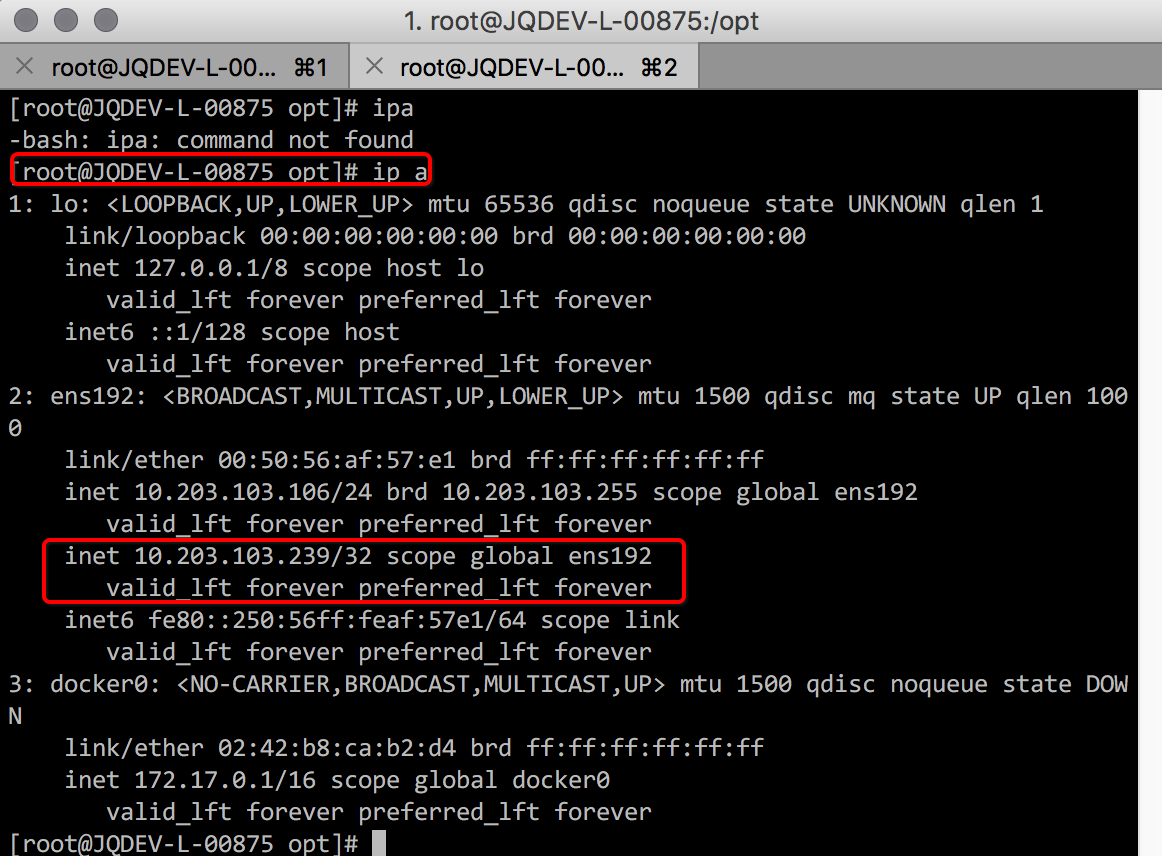
启动好registry（docker-distribution）及keepalived可以看到VIP在master上已经生效



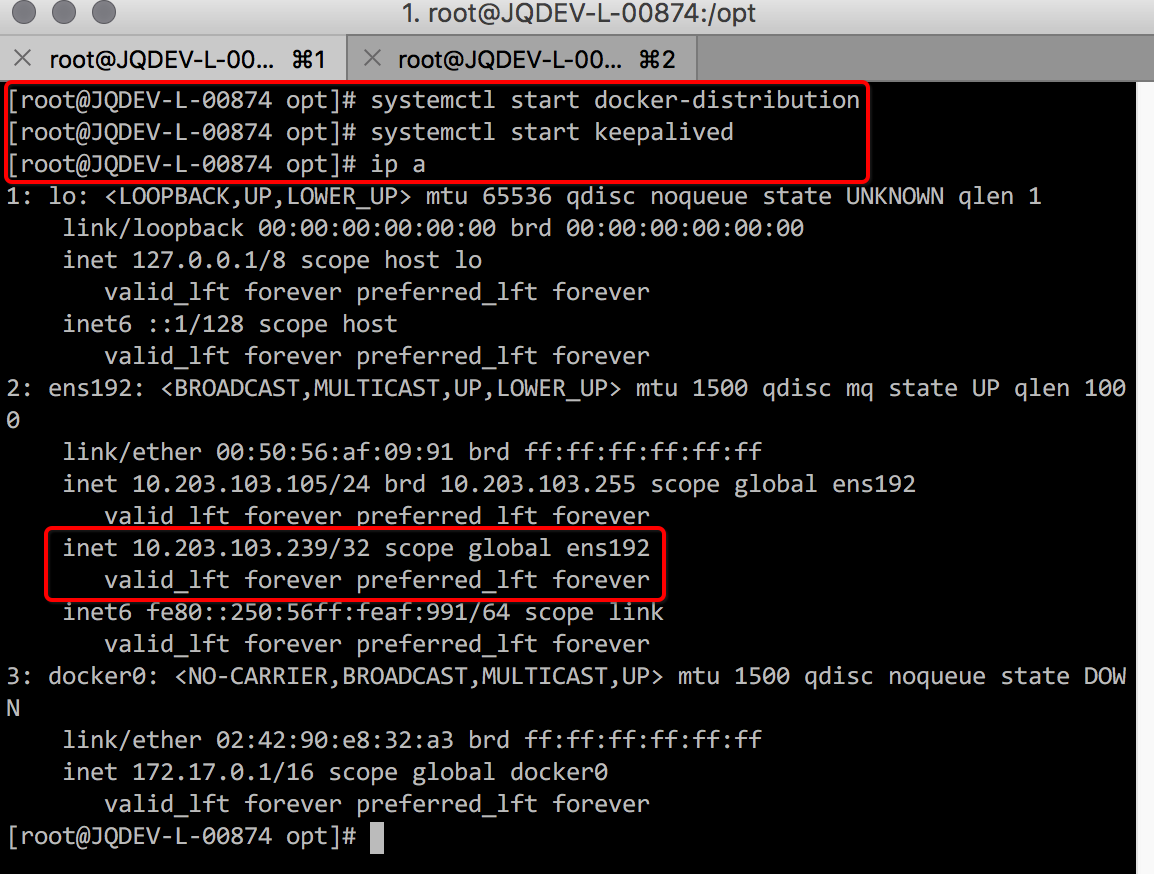
验证高可用

在master（10.203.103.105）上将registry停掉，3秒后可以看到Keepalived服务也已经被停掉（脚本检测定义），而后可以看到VIP已经漂移到BACKUP节点（10.203.103.106）

# systemctl stop docker-distribution



当master节点上的registry及keepalived服务重新起来后，可以看到VIP已经按照预期漂回，至此节点VIP高可用配置完成。



# 十六、证书更新部署

因证书需要定期更换，在此以QA环境为例操作验证平台内部各访问url/模块的证书更换。

更换master证书

将证书上传至目录 /etc/origin/master 后并对配置文件进行更改

# vim /etc/origin/master/master-config.yaml

…

assetConfig:

loggingPublicURL: https://kibana.inapps.jqpqa.c.saic-gm.net

logoutURL: https://idpdev.saic-gm.com/pkmslogout?filename=default.html&name=jqpreh&redirect=https://console.jqpqa.c.saic-gm.net:8443

masterPublicURL: https://console.jqpqa.c.saic-gm.net:8443

metricsPublicURL: https://hawkular-metrics.inapps.jqpqa.c.saic-gm.net/hawkular/metrics

publicURL: https://console.jqpqa.c.saic-gm.net:8443/console/

servingInfo:

bindAddress: 0.0.0.0:8443

bindNetwork: tcp4

certFile: master.server.crt

clientCA: ""

keyFile: master.server.key

namedCertificates:

- certFile: console.jqpqa.c.saic-gm.net.cer

keyFile: console.jqpqa.c.saic-gm.net.key

names:

- "console.jqpqa.c.saic-gm.net"

…

servingInfo:

bindAddress: 0.0.0.0:8443

bindNetwork: tcp4

certFile: master.server.crt

clientCA: ca-bundle.crt

keyFile: master.server.key

namedCertificates:

- certFile: console.jqpqa.c.saic-gm.net.cer

keyFile: console.jqpqa.c.saic-gm.net.key

names:

- "console.jqpqa.c.saic-gm.net"

…

在所有master上重启后登陆可以看到证书已经替换为所配置证书

# systemctl restart atomic-openshift-{master-api,node}

更换registry-console，Kibana，Metrics证书

因已经定义这几个模块的访问url为 \*.inapp.jqpqa.c.saic-gm.net，所以更换内部router的证书为inapps的即可。准备好\*.inapps的crt、key及对应ca

//生成 pem 文件

# cat \*.inapp.jqpqa.c.saic-gm.net.crt ca.crt > inrouter.pem

//备份原来的证书

# oc export secret inrouter-certs > old-inrouter-certs-secret.yaml

//替换

# oc secret new inrouter-certs tls.crt=inrouter.pem tls.key=\*.inapp.

jqpqa.c.saic-gm.net.key -o json -type=’kubernetes.io/tls’ -confirm | oc replace –f -

//重新部署

# oc deploy dc/inrouter --latest

需求验证

在浏览器导入（或已有）SGMCA.cer 后可以看到登录以下 url 都已经是信任：

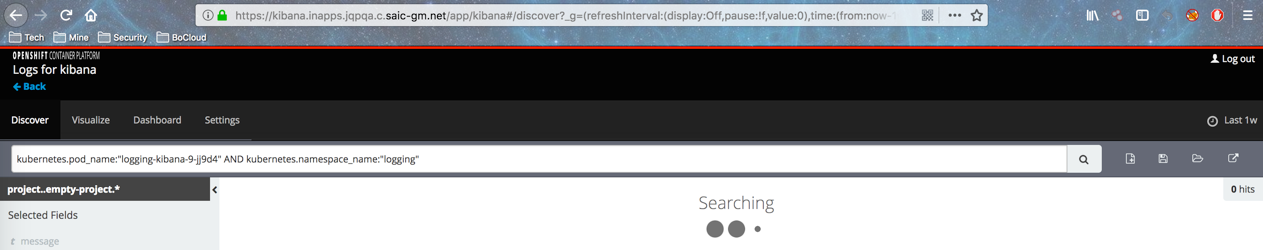
<https://console.jqpqa.c.saic-gm.net:8443>

<https://kibana.inapps.jqpqa.c.saic-gm.net>

<https://registry-console-default.inapps.jqpqa.c.saic-gm.net>

<https://hawkular-metrics.inapps.jqpqa.c.saic-gm.net>

例：



**备注**

如果部署中间出现错误或者失败，还原证书的方法：

# oc delete secret inrouter-certs

# oc create -f old-inrouter-certs-secret.yaml

# oc deploy dc/inrouter –latest

# 十七、Openshift平滑升级

目前Openshift Container Platform最新版为3.7版本。Openshift的版本升级需要一个一个小版本的逐步升级，也就是说在升级到Openshift 3.7之前，集群必须已经升级到最新的3.6版本。本次我们需要升级的版本是3.5，要升级到的版本是3.7，所以升级路径为3.5->3.6和3.6->3.7。

操作系统的包更新或版本升级均可能影响到平台，特别是iptables的规则或ovs相关。

官方升级文档请参考：

<https://docs.openshift.com/container-platform/3.6/install_config/upgrading/index.html>

和

<https://docs.openshift.com/container-platform/3.7/upgrading/index.html>

注意：在升级前请对环境进行备份，以便升级失败时的环境恢复

验证升级

在升级前或升级后均需要执行此步骤来验证集群的健康情况

1. 确认所有节点均为Ready状态

# oc get nodes

NAME STATUS AGE

master.example.com Ready,SchedulingDisabled 165d

node1.example.com Ready 165d

1. （升级后）确认运行的docker-registry和router镜像版本正确。

# oc get -n default dc/docker-registry -o json | grep \"image\"

"image": "openshift3/ose-docker-registry:<tag>",

# oc get -n default dc/router -o json | grep \"image\"

"image": "openshift3/ose-haproxy-router:<tag>",

3） 在master节点上使用诊断工具发现问题，排除必要的错误信息。

# oc adm diagnostics

...

[Note] Summary of diagnostics execution:

[Note] Completed with no errors or warnings seen.

备注，不是所有的错误都会影响升级工作，例如遇到以下错误，则可忽略，根据不同环境，可能遇到的问题不同，可以提交红帽800确认：

1. AGL0515: "Did not find ServiceAccounts: logging-deployer"

可忽略，参考https://access.redhat.com/solutions/3227201

从3.5升级至3.6

1. 升级前了解本次升级的已知问题

*因本次升级未遇到此问题，所以文档中未介绍*

https://docs.openshift.com/container-platform/3.6/install\_config/upgrading/upgrading\_known\_issues.html

1. 升级前验证

执行章节2的验证升级步骤

1. 更新每个计算节点的yum源

禁用ose-3.5的源

"rhel-7-server-ose-3.5-rpms"

启用ose-3.6的源

"rhel-7-server-ose-3.6-rpms"

确保以下每个yum源里的包是最新的

rhel-7-server-ose-3.6-rpms

"rhel-7-server-extras-rpms"

"rhel-7-fast-datapath-rpms"

执行

# yum clean all

1. 在每一个master节点上安装或升级atomic-openshift-utils包

# yum install atomic-openshift-utils

# yum update atomic-openshift-utils

当更新此包时，/usr/share/openshift/examples/目录并不会同时更新，所以请在更新后执行以下命令

# mkdir –p /usr/share/openshift/examples

# cp -R /usr/share/ansible/openshift-ansible/roles/openshift\_examples/files/examples/v3.6/\* /usr/share/openshift/examples/

在每个master节点上执行以下操作

mkdir –p /usr/share/openshift/examples

cp -R /usr/share/ansible/openshift-ansible/roles/openshift\_examples/files/examples/v3.6/\* user@masterx:/usr/share/openshift/examples

5）登陆集群管理员用户

在执行升级的过程中，需要登陆为集群管理员。

$ oc login

1. 检查ansible hosts文件

确认以下项目已设置

openshift\_deployment\_type=openshift-enterprise

openshift\_rolling\_restart\_mode=services

1. 执行升级Playbook

# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/upgrades/v3\_6/upgrade.yml

备注：

1. 执行任务：Fixup shared-resource-viewer role时如遇到以下问题，请再次执行升级playbook



参考：https://bugzilla.redhat.com/show\_bug.cgi?id=1507119

8）升级后验证

执行章节2的验证升级步骤

1. 升级EFK

必须使用这个playbook来更新EFK：

/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-logging.yml

1. 确认并更正ansible的host文件中以下内容：

[OSEv3:vars]

openshift\_hosted\_logging\_deploy=true

openshift\_hosted\_logging\_deployer\_prefix=registry.example.com:8888/openshift3/

openshift\_hosted\_logging\_deployer\_version=v3.6

1. 运行playbook

$ ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-logging.yml

10）升级Metrics

必须使用这个playbook来更新Metrics：

/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-metrics.yml

1. 确认并更正ansible的host文件中以下内容

openshift\_hosted\_metrics\_deploy=true

openshift\_hosted\_metrics\_deployer\_prefix=registry.example.com:8888/openshift3/

openshift\_hosted\_metrics\_deployer\_version=v3.6

openshift\_metrics\_hawkular\_hostname=<fqdn>

openshift\_metrics\_cassandra\_storage\_type=(emptydir|pv|dynamic)

其中openshift\_metrics\_cassandra\_storage\_type请选择与现有环境一致的选项。

1. 运行playbook

$ ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-metrics.yml

1. 升级后验证

执行章节2的验证升级步骤

从3.6升级至3.7

1. 迁移 etcd 数据: v2 to v3

在升级到3.7之前必须将etcd从v2版本迁移至v3版本。迁移过程为离线方式，所有etcd节点和master节点服务都将停止。迁移文档仅支持在ansible的host文件中含有etcd节点的迁移，并不支持内嵌etcd服务的数据迁移。

1. 确认openshift-ansible版本

必须将openshift-ansible更新至版本3.6.173.0.21或以上，再开始迁移工作

# rpm -q openshift-ansible

# yum upgrade openshift-ansible\\*

b）执行playbook migrate.yml

# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-etcd/migrate.yml

1. 回滚

如果迁移后发现问题，可以参考以下链接进行回滚。

i.停止master服务

# systemctl stop atomic-openshift-master-api atomic-openshift-master-controllers

ii.在master配置文件的kubernetesMasterConfig.apiServerArguments段中去掉以下2个键值storage-backend 和storage-media-type

kubernetesMasterConfig:

apiServerArguments:

...

iii.从备份中恢复，在迁移前保存在目录/var/lib/etcd中，例如：

/var/lib/etcd/openshift-backup-pre-migration20170825135732

iv.重启master服务

# systemctl restart atomic-openshift-master-api atomic-openshift-master-controllers

1. 升级前验证

执行章节2的验证升级步骤

1. 更新每个计算节点的yum源

禁用ose-3.6的源

"rhel-7-server-ose-3.6-rpms"

启用ose-3.7的源

"rhel-7-server-ose-3.7-rpms"

确保以下每个yum源里的包是最新的

rhel-7-server-ose-3.7-rpms

"rhel-7-server-extras-rpms"

"rhel-7-fast-datapath-rpms"

执行

# yum clean all

1. 升级atomic-openshift-utils包

# yum update atomic-openshift-utils

1. 检查ansible hosts文件

确认以下项目已设置

openshift\_deployment\_type=openshift-enterprise

openshift\_rolling\_restart\_mode=services

openshift\_disable\_check=docker\_image\_availability

1. 执行升级Playbook

# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/upgrades/v3\_7/upgrade.yml

1. 升级后验证

执行章节2的验证升级步骤

1. 升级EFK

必须使用这个playbook来更新EFK：

/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-logging.yml

1. 确认并更正ansible的host文件中以下内容：

注释掉升级3.6时加入的行

[OSEv3:vars]

#openshift\_openshift\_hosted\_logging\_deployer\_version=v3.6

logging\_install\_logging=true 1

openshift\_logging\_image\_version=<tag> 2

其中<tag>为最新的版本号，例如v3.7.23

1. 运行playbook

$ ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-logging.yml

1. 升级Metrics

必须使用这个playbook来更新Metrics：

/usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-metrics.yml

1. 确认并更正ansible的host文件中以下内容

注释掉升级3.6时加入的行

#openshift\_hosted\_metrics\_deployer\_version=v3.6

openshift\_metrics\_install\_metrics=true 1

openshift\_metrics\_image\_version=<tag> 2

openshift\_metrics\_hawkular\_hostname=<fqdn> 3

openshift\_metrics\_cassandra\_storage\_type=(emptydir|pv|dynamic)

其中<tag>为最新的版本号，例如v3.7.23，openshift\_metrics\_cassandra\_storage\_type请选择与现有环境一致的选项。

1. 运行playbook

$ ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/openshift-metrics.yml

1. 升级后验证

执行章节2的验证升级步骤

操作系统更新或升级

按以下步骤安全的更新操作系统：

1. 确认节点不能被调度

$ oc adm manage-node <node\_name> --schedulable=false

1. 将pod迁往其他计算节点

$ oc adm drain <node\_name> --force --delete-local-data --ignore-daemonsets

1. 更新软件包并重启操作系统
2. 将节点重新设置为可被调度

$ oc adm manage-node <node\_name> --schedulable=true

附录：样例，升级后的ansible hosts文件

[OSEv3:children]

masters

nodes

etcd

lb

[OSEv3:vars]

ansible\_ssh\_user=root

openshift\_deployment\_type=openshift-enterprise

#deployment\_type=openshift-enterprise

openshift\_rolling\_restart\_mode=services

openshift\_disable\_check=memory\_availability,disk\_availability,docker\_image\_availability

openshift\_master\_default\_subdomain=cloud.ocp35.homelab.com

openshift\_master\_identity\_providers=[{'name': 'htpasswd\_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]

oreg\_url=registry.homelab.com:5000/openshift3/ose-${component}:${version}

openshift\_examples\_modify\_imagestreams=true

openshift\_master\_cluster\_method=native

openshift\_master\_cluster\_hostname=ocp35-cluster.homelab.com

openshift\_master\_cluster\_public\_hostname=ocp35-cluster.homelab.com

openshift\_logging\_install\_logging=true

openshift\_metrics\_install\_metrics=true

openshift\_hosted\_logging\_deploy=true

openshift\_hosted\_logging\_deployer\_prefix=registry.homelab.com:5000/openshift3/

#openshift\_hosted\_logging\_deployer\_version=v3.5

#openshift\_hosted\_logging\_deployer\_version=v3.6

openshift\_logging\_image\_version=v3.7.23

openshift\_hosted\_metrics\_deploy=true

openshift\_hosted\_metrics\_deployer\_prefix=registry.homelab.com:5000/openshift3/

#openshift\_hosted\_metrics\_deployer\_version=v3.5

#openshift\_hosted\_metrics\_deployer\_version=v3.6

openshift\_metrics\_hawkular\_hostname=hawkular-metrics.cloud.ocp35.homelab.com

openshift\_metrics\_cassandra\_storage\_type=emptydir

openshift\_metrics\_image\_version=v3.7.23

openshift\_router\_selector='region=infra'

openshift\_registry\_selector='region=infra'

# apply updated node defaults

openshift\_node\_kubelet\_args={'pods-per-core': ['10'], 'max-pods': ['250'], 'image-gc-high-threshold': ['90'], 'image-gc-low-threshold': ['80']}

# enable ntp on masters to ensure proper failover

openshift\_clock\_enabled=true

# host group for masters

[masters]

ocp35\_master1.homelab.com

ocp35\_master2.homelab.com

ocp35\_master3.homelab.com

# host group for etcd

[etcd]

ocp35\_master1.homelab.com

ocp35\_master2.homelab.com

ocp35\_master3.homelab.com

# Specify load balancer host

[lb]

ocp35\_lb.homelab.com

# host group for nodes, includes region info

[nodes]

ocp35\_master[1:3].homelab.com

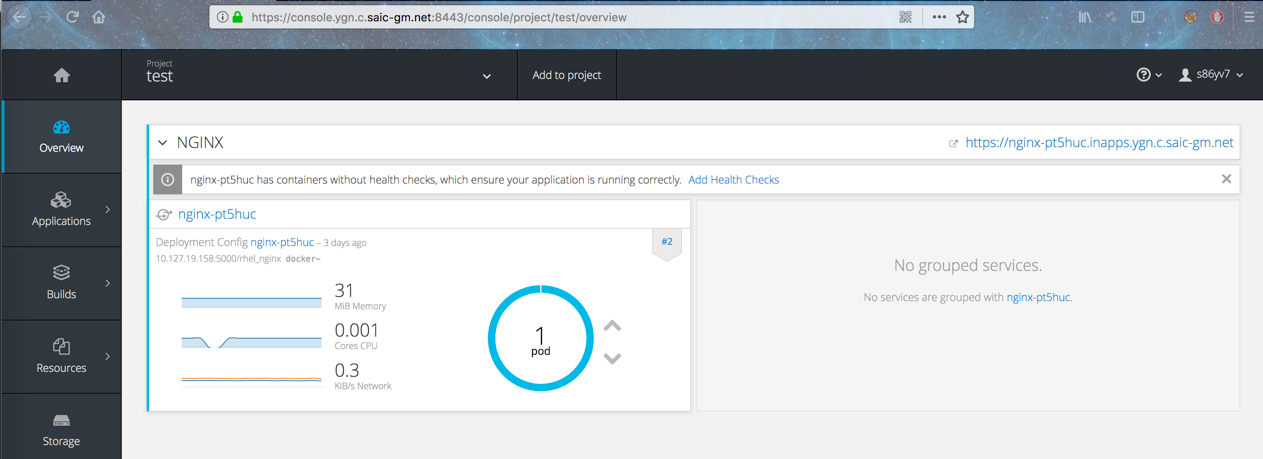
ocp35\_node1.homelab.com openshift\_node\_labels="{'region': 'primary', 'zone': 'east'}"

ocp35\_node2.homelab.com openshift\_node\_labels="{'region': 'primary', 'zone': 'west'}"

ocp35\_infra.homelab.com openshift\_node\_labels="{'region': 'infra', 'zone': 'default'}"

# 十八、服务Nodeport配置

根据实际应用场景，需要对平台的一些服务进行暴露以便外部程序访问使用，Nodeport是其中一种有效的方式，现对YGN的一个 nginx应用进行服务暴露，以命令行操作为例。暴露后，访问路径为IP:Nodeport，使用XXX方提供的域名 services.ygn.c.saic-xx.net代替IP。



正常发布一个nginx应用

对service patch nodeport配置并替换

# oc project test

# oc patch svc nginx-pt5huc -p '{"spec":{"type": "NodePort"}}'

# oc get svc nginx-pt5huc -o json | sed 's/"nodePort": 3..../" nodePort": 30012/g' | oc replace -f -

需求验证

对应用进行IP:Nodeport和domain:Nodeport访问测试





# 十九、域名更新部署

根据需求有时有可能要对平台的访问url进行更新，如console，kibana console，registry-console等，现进行简单配置说明

更改配置

所有的master上更改对应配置(10.203.65.53/54/55)

# vim /etc/origin/master/master-config.yaml

admissionConfig:

pluginConfig:

BuildDefaults:

configuration:

apiVersion: v1

env: []

kind: BuildDefaultsConfig

resources:

limits: {}

requests: {}

BuildOverrides:

configuration:

apiVersion: v1

kind: BuildOverridesConfig

openshift.io/ImagePolicy:

configuration:

apiVersion: v1

executionRules:

- matchImageAnnotations:

- key: images.openshift.io/deny-execution

value: 'true'

name: execution-denied

onResources:

- resource: pods

- resource: builds

reject: true

skipOnResolutionFailure: true

kind: ImagePolicyConfig

apiLevels:

- v1

apiVersion: v1

assetConfig:

loggingPublicURL: https://kibana.inapps.jqp.c.saic-xx.net

logoutURL: https://idpdev.saic-gm.com/pkmslogout?filename=default.html&name=jqpreh&redirect=https://console.jqpqa.c.saic-gm.net:8443

masterPublicURL: https://console.jqp.c.saic-xx.net:8443

metricsPublicURL: https://hawkular-metrics.inapps.jqp.c.saic-xx.net/hawkular/metrics

publicURL: https://console.jqp.c.saic-xx.net:8443/console/

servingInfo:

bindAddress: 0.0.0.0:8443

bindNetwork: tcp4

certFile: master.server.crt

clientCA: ""

keyFile: master.server.key

namedCertificates:

- certFile: console.jqpqa.c.saic-gm.net.cer

keyFile: console.jqpqa.c.saic-gm.net.key

names:

- "console.jqpqa.c.saic-gm.net"

maxRequestsInFlight: 0

requestTimeoutSeconds: 0

controllerConfig:

serviceServingCert:

signer:

certFile: service-signer.crt

keyFile: service-signer.key

controllerLeaseTTL: 30

controllers: '\*'

corsAllowedOrigins:

- 127.0.0.1

- localhost

- 10.203.103.91

- kubernetes.default

- kubernetes.default.svc.cluster.local

- kubernetes

- openshift.default

- console.jqpqa.c.saic-gm.net

- JQDEV-L-00860.jqdev.shanghaigm.com

- openshift.default.svc

- 172.30.0.1

- jqdev-l-00860.jqdev.shanghaigm.com

- openshift.default.svc.cluster.local

- kubernetes.default.svc

- openshift

dnsConfig:

bindAddress: 0.0.0.0:8053

bindNetwork: tcp4

etcdClientInfo:

ca: master.etcd-ca.crt

certFile: master.etcd-client.crt

keyFile: master.etcd-client.key

urls:

- https://jqdev-l-00860.jqdev.shanghaigm.com:2379

- https://jqdev-l-00861.jqdev.shanghaigm.com:2379

- https://jqdev-l-00862.jqdev.shanghaigm.com:2379

etcdStorageConfig:

kubernetesStoragePrefix: kubernetes.io

kubernetesStorageVersion: v1

openShiftStoragePrefix: openshift.io

openShiftStorageVersion: v1

imageConfig:

format: jqdev-l-00874.jqdev.shanghaigm.com:5000/openshift3/ose-${component}:${version}

latest: false

kind: MasterConfig

kubeletClientInfo:

ca: ca-bundle.crt

certFile: master.kubelet-client.crt

keyFile: master.kubelet-client.key

port: 10250

kubernetesMasterConfig:

apiServerArguments:

controllerArguments:

masterCount: 3

masterIP: 10.203.103.91

podEvictionTimeout:

proxyClientInfo:

certFile: master.proxy-client.crt

keyFile: master.proxy-client.key

schedulerArguments:

schedulerConfigFile: /etc/origin/master/scheduler.json

servicesNodePortRange: ""

servicesSubnet: 172.30.0.0/16

staticNodeNames: []

masterClients:

externalKubernetesClientConnectionOverrides:

acceptContentTypes: application/vnd.kubernetes.protobuf,application/json

burst: 400

contentType: application/vnd.kubernetes.protobuf

qps: 200

externalKubernetesKubeConfig: ""

openshiftLoopbackClientConnectionOverrides:

acceptContentTypes: application/vnd.kubernetes.protobuf,application/json

burst: 600

contentType: application/vnd.kubernetes.protobuf

qps: 300

openshiftLoopbackKubeConfig: openshift-master.kubeconfig

masterPublicURL: https://console.jqpqa.c.saic-xx.net:8443

networkConfig:

clusterNetworkCIDR: 10.128.0.0/14

externalIPNetworkCIDRs:

- 0.0.0.0/0

hostSubnetLength: 9

networkPluginName: redhat/openshift-ovs-subnet

serviceNetworkCIDR: 172.30.0.0/16

oauthConfig:

assetPublicURL: https://console.jqp.c.saic-xx.net:8443/console/

grantConfig:

method: auto

identityProviders:

- name: ocpoauth

challenge: false

login: true

mappingMethod: add

provider:

apiVersion: v1

kind: OpenIDIdentityProvider

clientID: 56e557db-fca2-4187-8f94-d2e62d3f7101

clientSecret: 70f6c030-43c1-4c99-8d35-5e3d89142f7b

ca: SGMCA.crt

claims:

id:

- sub

urls:

authorize: https://idpdev.saic-gm.com/oauthweb/oauth/authorize

token: https:// idpdev.saic-gm.com/oauthweb/oauth/token.jsp

masterCA: ca-bundle.crt

masterPublicURL: https://console.jqp.c.saic-xx.net:8443

masterURL: https://jqdev-l-00860.jqdev.shanghaigm.com:8443

sessionConfig:

sessionMaxAgeSeconds: 3600

sessionName: ssn

sessionSecretsFile: /etc/origin/master/session-secrets.yaml

tokenConfig:

accessTokenMaxAgeSeconds: 86400

authorizeTokenMaxAgeSeconds: 500

pauseControllers: false

policyConfig:

bootstrapPolicyFile: /etc/origin/master/policy.json

openshiftInfrastructureNamespace: openshift-infra

openshiftSharedResourcesNamespace: openshift

projectConfig:

defaultNodeSelector: ""

projectRequestMessage: ""

projectRequestTemplate: ""

securityAllocator:

mcsAllocatorRange: s0:/2

mcsLabelsPerProject: 5

uidAllocatorRange: 1000000000-1999999999/10000

routingConfig:

subdomain: inapps.jqpqa.c.saic-gm.net

serviceAccountConfig:

limitSecretReferences: false

managedNames:

- default

- builder

- deployer

masterCA: ca-bundle.crt

privateKeyFile: serviceaccounts.private.key

publicKeyFiles:

- serviceaccounts.public.key

servingInfo:

bindAddress: 0.0.0.0:8443

bindNetwork: tcp4

certFile: master.server.crt

clientCA: ca-bundle.crt

keyFile: master.server.key

namedCertificates:

- certFile: console.jqpqa.c.saic-xx.net.cer

keyFile: console.jqpqa.c.saic-xx.net.key

names:

- "console.jqpqa.c.saic-gm.net"

maxRequestsInFlight: 500

requestTimeoutSeconds: 3600

volumeConfig:

dynamicProvisioningEnabled: true

**如果有更改console（master\_public\_url）的还需要重新部署master的证书**

# ansible-playbook -i /etc/ansible/hosts /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-cluster/redeploy-master-certificates.yml

重启Openshift服务

在所有master上重启

# systemctl restart atomic-openshift-{master-api,node}

需求验证

可以看到对应url都已经变成所修改的并且证书对应

# 二十、Redis集群化集群部署

1. 部署架构说明

部署Redis集群至少需要三个分组，每组为主从备份方式部署，因此至少需要6个节点Redis，集群的结构如下图所示



Redis集群实现方式与Docker网络机制的限制另外外部应用也需要访问Redis集群需要在OCP中采用HostNetwork方式使用宿主机IP和端口直接访问Redis集群



1. 部署环境说明

分别在每个物理节点使用DeploymentConfig模板进行部署共需创建和部署6

个节点具体参数见下面说明。

控制台URL https://master.ocp.com:8443/console/

Redis镜像registry.ocp.com:5000/rhscl/redis-32-rhel7:latest

* 1. Node节点

将6个Redis节点分为Master和Slave两组分别在三台测试机上部署

Master和Slave节点Master使用6379端口Slave使用6380

环境中需要部署的Redis节点列表如下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | Hostname / IP | 端口 | 标签(Key=Value) | 类型 |
| 1 | rediscluster11 | 192.168.80.121 | 6379，16379 | redis6379=true | Master |
| 2 | rediscluster12 | 192.168.80.122 | 6380，16380 | redis6380=true | Slave |
| 3 | rediscluster21 | 192.168.80.122 | 6379，16379 | redis6379=true | Master |
| 4 | rediscluster22 | 192.168.80.123 | 6380，16380 | redis6380=true | Slave |
| 5 | rediscluster31 | 192.168.80.123 | 6379，16379 | redis6379=true | Master |
| 6 | rediscluster32 | 192.168.80.121 | 6380，16380 | redis6380=true | Slave |

需要给每个Node设置标签见标签列在部署时通过该标签Selector选择Redis部署节点。

* 1. Volume定义

通过配置持久化为Redis保存数据为保证Redis的性能使用本地磁盘进行持久化需要在PaaS平台中通过hostPath的方式配置持久化持久化卷列表及参数如下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 容量 | 路径Path | Access Mode |
| 1 | rediscluster11 | 20Gi | 192.168.80.121:/opt/redis-32-rhel7/6379-data | ReadWriteOnce |
| 2 | rediscluster12 | 20Gi | 192.168.80.122:/opt/redis-32-rhel7/6380-data | ReadWriteOnce |
| 3 | rediscluster21 | 20Gi | 192.168.80.122:/opt/redis-32-rhel7/6379-data | ReadWriteOnce |
| 4 | rediscluster22 | 20Gi | 192.168.80.123:/opt/redis-32-rhel7/6380-data | ReadWriteOnce |
| 5 | rediscluster31 | 20Gi | 192.168.80.123:/opt/redis-32-rhel7/6379-data | ReadWriteOnce |
| 6 | rediscluster32 | 20Gi | 192.168.80.121:/opt/redis-32-rhel7/6380-data | ReadWriteOnce |

分别在每个节点上创建目录并赋予权限

1. 实施步骤

本次基础镜像使用registry.access.redhat.com/rhscl/redis-32-rhel7不需要单独制作镜像文件。

部署Redis Cluster需要在redis.conf中需要修改如下配置

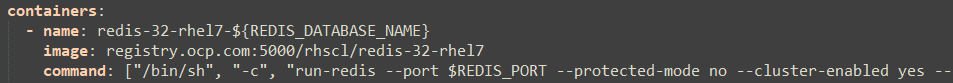
cluster-enabled yes

cluster-config-file nodes-6379.conf

cluster-node-timeout 15000

appendonly yes

或者在启动Redis时通过参数指定如--cluster-enabled yes本次将相应的参数写入了需要使用DeploymentConfig模板在执行参数会覆盖在redis.conf的定义。



* 1. Redis镜像的集群部署验证

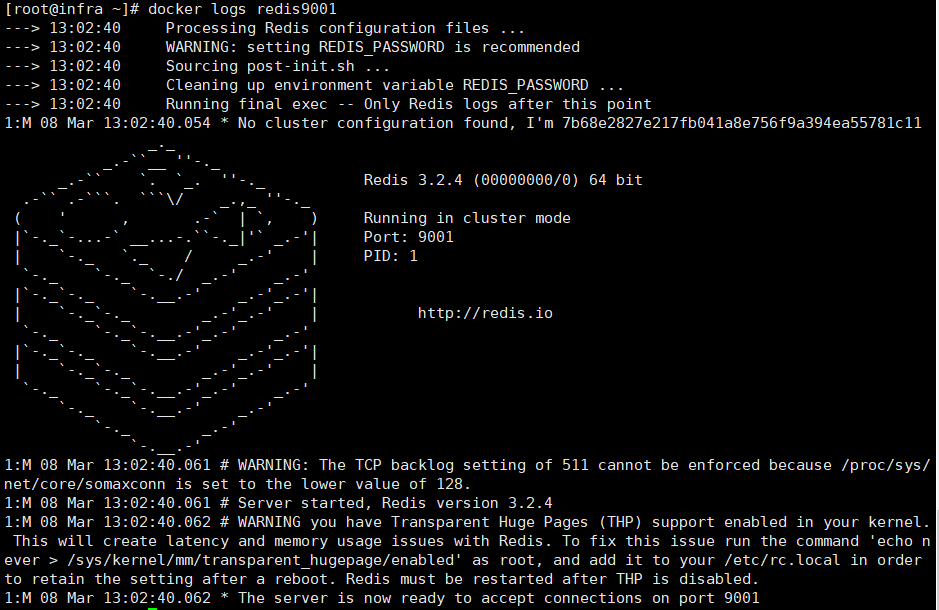
1．创建Redis Cluster

使用以下命名启动6个Docker Container端口分别为9001、9002、9003、9004、9005、9006在Docker中启动时必须使用“--net=host”方式

# docker run -d --net=host --name redis9001 registry.access.redhat.

com/rhscl/redis-32-rhel7 /bin/sh -c "run-redis --daemonize no --port 9001 --protected-mode no --cluster-enabled yes --cluster-config-file nodes.conf --cluster-node-timeout 15000 --appendonly yes"

执行之后使用docker logs redis9001可以查看到输出的日志信息



在Container中使用cluster meet、cluster replicate、cluster addslots命令执行创建集群

# redis-cli -c -h 127.0.0.1 -p 9001 cluster meet 127.0.0.1 9002

使用cluster replicate命令为当前节点指定Master节点当前节点将变为Slave节点

# redis-cli -c -h 127.0.0.1 -p 9002 cluster replicate 32cc09405282

2da3b5afeac…...

其中32cc...为对应节点的ID。

为Redis Master分配SLOTS

# for i in {0..5400}; do redis-cli -c -h 127.0.0.1 -a $REDIS\_PASSWORD -p 9001 cluster addslots $i; done;

# for i in {5401..10800}; do redis-cli -c -h 127.0.0.1 -a $REDIS\_PASSWORD -p 9003 cluster addslots $i; done;

# for i in {10801..16383}; do redis-cli -c -h 127.0.0.1 -a $REDIS\_PASSWORD -p 9005 cluster addslots $i; done;

2 使用redis-cli命令验证

使用redis-cli命令连接到任意节点如redis9001，查看集群节点状态在连接时需要使用“-c”参数以指明为集群模式Ping命令如下

# redis-cli -c -h 192.168.80.121 -p 9001 ping

在查看Redis集群节点状态使用命令

# redis-cli -c -h 192.168.80.122 -p 9001 cluster nodes

3 使用Jave客户端验证

jedis支持Redis的Cluster模式使用jedis可以在pom.xml中添加

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.9.0</version>

<type>jar</type>

<scope>compile</scope>

</dependency>

详情可参见“redis-sample”工程执行java程序

java -jar redis-sample-0.0.1-SNAPSHOT.jar -l <服务器列表使用“;”分割>

该java程序主要工作为连接到Redis Cluster并执行set操作在java程序执行过程中使用“cluster nodes”监控集群节点状态。

使用docker stop <name> 停止一个master节点在java控制台和监控集群节点中分别观察各自的执行状态。

以下是几种常见的错误说明

1. java.net.ConnectException: Connection refused (Connection refused)

当停止的Redis节点是当前jedis客户端使用的节点时会出现拒绝连接的错误。

1. redis.clients.jedis.exceptions.JedisClusterException: CLUSTERDOWN The cluster is down

当Redis的Master节点停止后Redis在迁移集群状态时jedis会出现此错误。

为处理上述错误通常在执行redis操作时需要使用try{} catch{} 来处理异常或在必要的时候进行重试。

* 1. 推送Redis镜像到本地Registry

在本地使用docker build命令构建Redis镜像并将镜像导入到PaaS平台的本地 Docker Registry。

**如果镜像已经存在此步骤可忽略。**

* 1. 创建数据文件

分别在每个节点上创建数据文件目录并设置相应权限例如

# mkdir -p /opt/redis-32-rhel7/6379-data

# chmod -R 777 /opt/redis-32-rhel7/6379-data

# chcon -Rt svirt\_sandbox\_file\_t /opt/redis-32-rhel7/6379-data

# mkdir -p /opt/redis-32-rhel7/6380-data

# chmod -R 777 /opt/redis-32-rhel7/6380-data

# chcon -Rt svirt\_sandbox\_file\_t /opt/redis-32-rhel7/6380-data

其中6379，6380分别为Redis的启动端口，需要分别为6个节点创建文件目录（见1.2.2 Volume定义）

* 1. 创建项目和部署Redis镜像

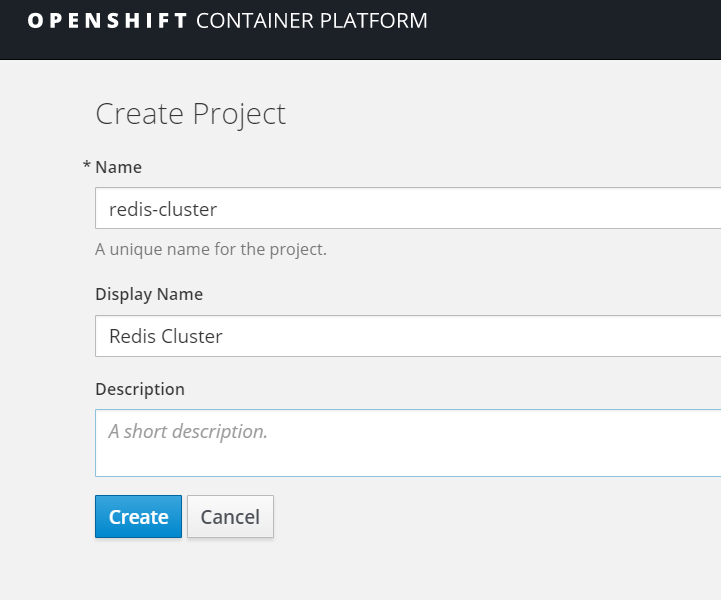
上传模板文件到Master节点下的/root/redis目录

# scp dc-redis-cluster-template.yaml root@192.168.80.101:/root/redis

在OCP中创建名为“Redis Cluster”的项目使用命令行方式

# oc new-project redis-cluster --display-name="Redis Cluster"

或者使用Web Console方式



使用以下命令设置Project权限

# oadm policy add-scc-to-group privileged system:serviceaccounts:redis-cluster

或者分别赋予下面的权限

# oadm policy add-scc-to-group anyuid system:serviceaccounts:redis-cluster

# oadm policy add-scc-to-group hostnetwork system:serviceaccounts:redis-cluster

# oadm policy add-scc-to-group hostmount-anyuid system:serviceaccounts:

redis-cluster

分别为OpenShift节点设置Label例如

# oc label node <nodename> redis6379=true

其中nodename为openshift节点名称可以使用oc get nodes查看有哪些节点 rediscluster11表示为在该节点上部署哪个Redis Server。

oc label node node1.ocp.com redis6379=true

oc label node node2.ocp.com redis6379=true

oc label node node3.ocp.com redis6379=true

oc label node node1.ocp.com redis6380=true

oc label node node2.ocp.com redis6380=true

oc label node node3.ocp.com redis6380=true

* 1. 使用模板分别创建6个Redis节点

dc-redis-cluster-template.yaml是Redis节点部署的模板文件通过执行命令可以通过模板创建、部署集群模式的Redis Server。命令如下

# oc process -p REDIS\_PORT=6379 -p REDIS\_DATABASE\_NAME=redis6379 -p REDIS\_NODE\_LABEL=redis6379 -f dc-redis-cluster-template.yaml |oc create -f - -n redis-cluster

# oc process -p REDIS\_PORT=6380 -p REDIS\_DATABASE\_NAME=redis6380 -p REDIS\_NODE\_LABEL=redis6380 -f dc-redis-cluster-template.yaml |oc create -f - -n redis-cluster

其中

●REDIS\_PORTRedis的启动端口“10000+启动端口”为集群使用端口，请确保两个端口未被占用

●REDIS\_DATABASE\_NAME指明要部署的Redis名称

●REDIS\_NODE\_LABEL要求和该节点上设置的标签内容一致

●REDIS\_PASSWORD Redis的数据库密码默认设置为“welcome1”

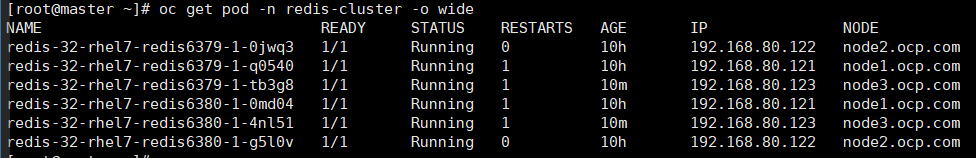
●MEMORY\_LIMIT Redis的内存限制默认为“64Mi”

执行以下命令是的每个DC运行3个Pods

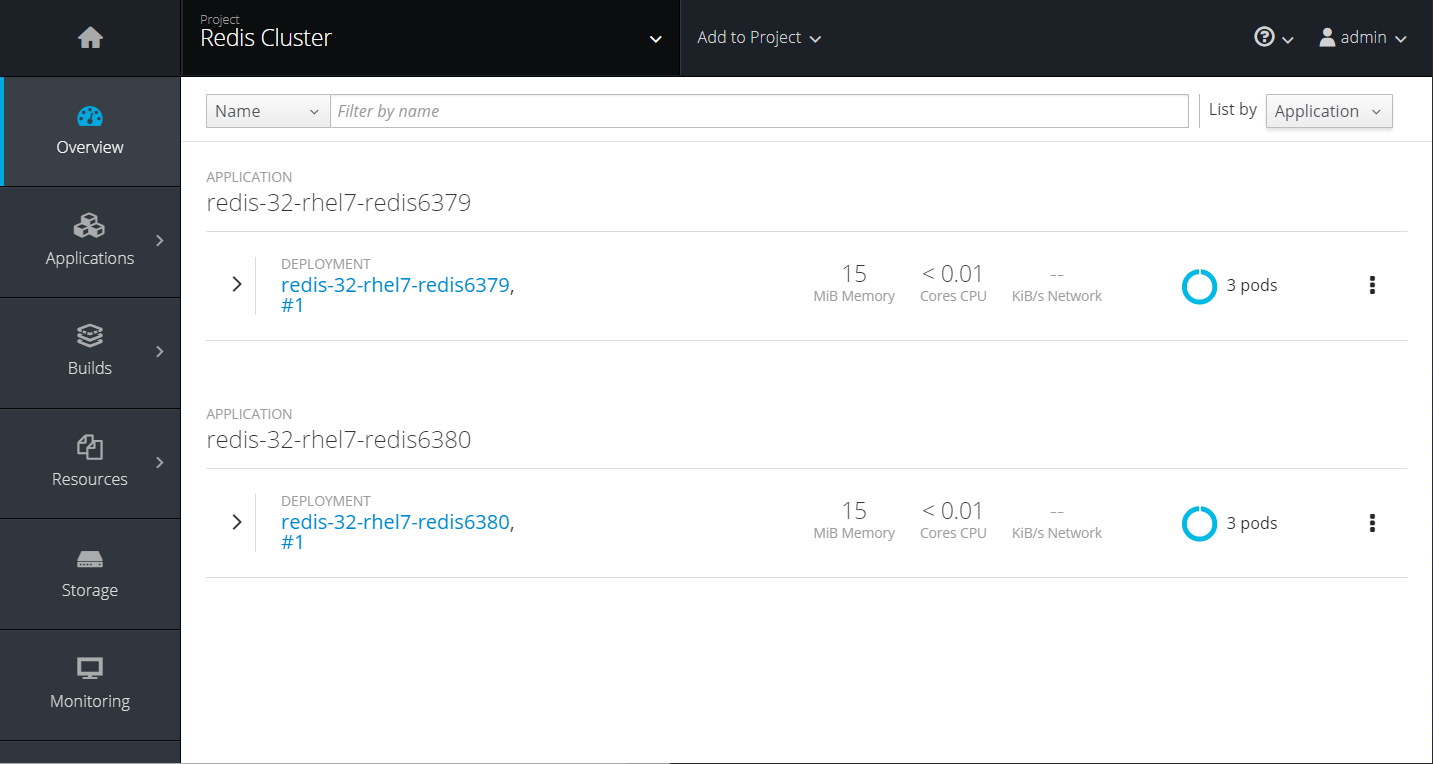
# oc scale dc redis-32-rhel7-redis6379 --replicas=3 -n redis-cluster

# oc scale dc redis-32-rhel7-redis6380 --replicas=3 -n redis-cluster

检查Pods的执行状态



在控制台可以查看6个Redis Server的状态



可以选择任意Pod节点进入Terminal使用redis-cli命令测试Redis服务器是否启动成功。如

# redis-cli -h 127.0.0.1 -a $REDIS\_PASSWORD -p $REDIS\_PORT ping

返回结果应为“PONG”。

需要进行防火墙设置将对应端口放开分别在三个节点上修改IPTABLES配置

# cp /etc/sysconfig/iptables /etc/sysconfig/iptables.redis.bak.$(date "+%Y%m%d%H%M%S");

# sed -i '/.\*--dport 22 -j ACCEPT.\*/a\-A INPUT -p tcp -m state --state NEW -m tcp --dport 6379 -j ACCEPT' /etc/sysconfig/iptables;

# sed -i '/.\*--dport 22 -j ACCEPT.\*/a\-A INPUT -p tcp -m state --state NEW -m tcp --dport 6380 -j ACCEPT' /etc/sysconfig/iptables;

# sed -i '/.\*--dport 22 -j ACCEPT.\*/a\-A INPUT -p tcp -m state --state NEW -m tcp --dport 16379 -j ACCEPT' /etc/sysconfig/iptables;

# sed -i '/.\*--dport 22 -j ACCEPT.\*/a\-A INPUT -p tcp -m state --state NEW -m tcp --dport 16380 -j ACCEPT' /etc/sysconfig/iptables;

重启IPTABLES使配置生效

# systemctl restart iptables;

* 1. 执行创建Redis Cluster命令

获得6个节点的IP地址选择任意Pod节点进入Terminal使用cluster meet命令在集群中添加节点。如下所示

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster meet 192.168.80.121 6380

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster meet 192.168.80.122 6379

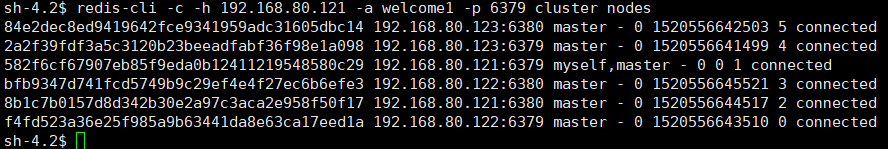
# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster meet 192.168.80.122 6380

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster meet 192.168.80.123 6379

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster meet 192.168.80.123 6380

此时添加的节点均为Master节点

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster nodes



使用cluster replicate命令为当前节点指定Master节点当前节点将变为Slave节点

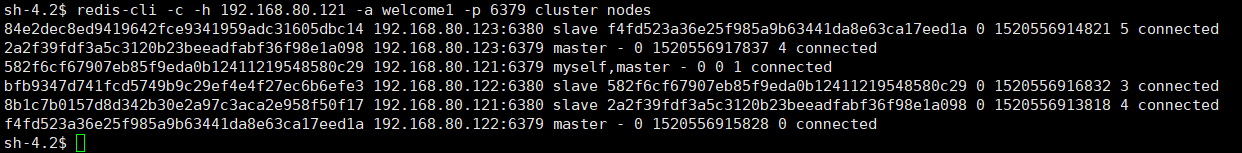
# redis-cli -c -h 192.168.80.122 -a welcome1 -p 6380 cluster replicate 582f6cf67907eb85f9eda0b12411219548580c29

# redis-cli -c -h 192.168.80.123 -a welcome1 -p 6380 cluster replicate f4fd523a36e25f985a9b63441da8e63ca17eed1a

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6380 cluster replicate 2a2f39fdf3a5c3120b23beeadfabf36f98e1a098

查看集群节点信息

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster nodes



为Redis Master分配SLOTS

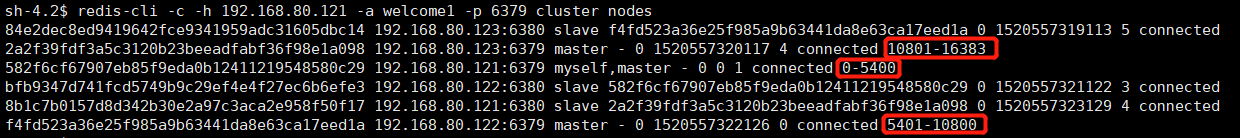
# for i in {0..5400}; do redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster addslots $i; done;

# for i in {5401..10800}; do redis-cli -c -h 192.168.80.122 -a welcome1 -p 6379 cluster addslots $i; done;

# for i in {10801..16383}; do redis-cli -c -h 192.168.80.123 -a welcome1 -p 6379 cluster addslots $i; done;

查看集群节点信息

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster nodes

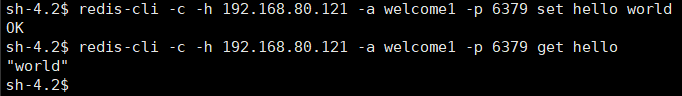


* 1. 验证Redsi Cluster

选择任意Pod节点进入Terminal使用redis-cli命令验证Redis Cluster是否正常运行。

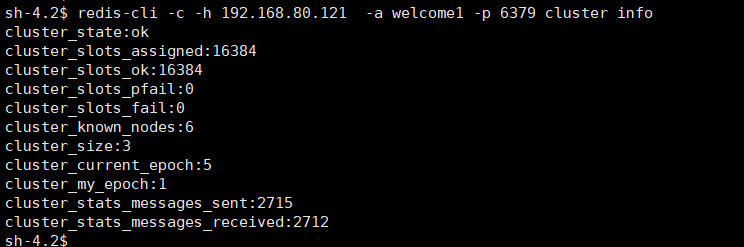
例如

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 set hello world

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 get hello 

执行如下命令

# redis-cli -c -h 192.168.80.121 -a welcome1 -p 6379 cluster info



4 Redis与应用集成

Redis的Java客户端jedis支持Redis Cluster模式使用jedis可以在pom.xml中添加

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

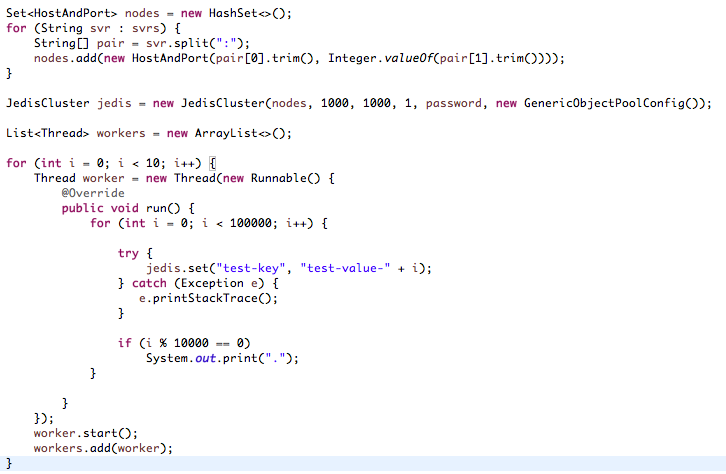
<version>2.9.0</version>

<type>jar</type>

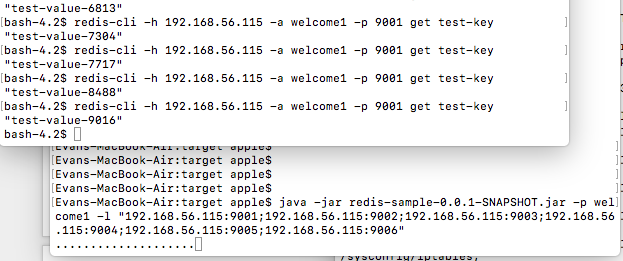
<scope>compile</scope>

</dependency>

例如



执行情况如下图



五、参考

Redis镜像参考<https://github.com/sclorg/redis-container/tree/master/3.2>

ImageStream基础镜像参考<https://github.com/openshift/origin/blob/master/examples/db-templates/redis-persistent-template.json>

Redis on OpenShift参考<https://github.com/mangirdaz/redis-openshift>

Redis Cluster<https://redis.io/topics/cluster-tutorial>

六、附录

**dc-redis-cluster-template.yaml**

#

# oadm policy add-scc-to-group privileged system:serviceaccounts:redis-cluster

#

# create redis server by command line:

# oc process -p REDIS\_PORT=9012 -p REDIS\_DATABASE\_NAME=rediscluster11 -p REDIS\_NODE\_LABEL=rediscluster11 -f dc-redis-cluster-template.yaml |oc create -f -

#

# delete all by labels:

# oc delete all -lapp=redis-32-rhel7-rediscluster11

#

kind: Template

apiVersion: v1

metadata:

name: redis-cluster-template

objects:

- apiVersion: v1

kind: DeploymentConfig

metadata:

name: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

namespace: redis-cluster

selfLink: /oapi/v1/namespaces/redis-cluster/deploymentconfigs/redis-32-rhel7-${REDIS\_DATABASE\_NAME}

creationTimestamp: null

labels:

app: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

annotations:

openshift.io/generated-by: OpenShiftWebConsole

spec:

strategy:

type: Recreate

recreateParams:

timeoutSeconds: 600

resources: {}

activeDeadlineSeconds: 21600

triggers:

- type: ConfigChange

replicas: 1

test: false

selector:

app: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

deploymentconfig: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

template:

metadata:

creationTimestamp: null

labels:

app: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

deploymentconfig: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

annotations:

openshift.io/generated-by: OpenShiftWebConsole

spec:

volumes:

- name: redis-32-rhel7-data

hostPath:

path: /opt/redis-32-rhel7/${REDIS\_PORT}-data

containers:

- name: redis-32-rhel7-${REDIS\_DATABASE\_NAME}

image: registry.ocp.com:5000/rhscl/redis-32-rhel7

command: ["/bin/sh", "-c", "run-redis --port $REDIS\_PORT --protected-mode no --cluster-enabled yes --cluster-config-file nodes.conf --cluster-node-timeout 15000 --appendonly yes --masterauth ${REDIS\_PASSWORD}"]

ports:

- hostPort: ${REDIS\_PORT}

containerPort: ${REDIS\_PORT}

protocol: TCP

- hostPort: 1${REDIS\_PORT}

containerPort: 1${REDIS\_PORT}

protocol: TCP

env:

- name: REDIS\_PORT

value: ${REDIS\_PORT}

- name: REDIS\_PASSWORD

value: ${REDIS\_PASSWORD}

resources:

limits:

memory: ${MEMORY\_LIMIT}

volumeMounts:

- name: redis-32-rhel7-data

mountPath: /var/lib/redis/data

livenessProbe:

exec:

command:

- /bin/sh

- -i

- -c

- test "$(redis-cli -h 127.0.0.1 -a $REDIS\_PASSWORD -p $REDIS\_PORT ping)" == "PONG"

initialDelaySeconds: 15

timeoutSeconds: 3

readinessProbe:

exec:

command:

- /bin/sh

- -i

- -c

- test "$(redis-cli -h 127.0.0.1 -a $REDIS\_PASSWORD -p $REDIS\_PORT ping)" == "PONG"

initialDelaySeconds: 10

timeoutSeconds: 5

periodSeconds: 10

terminationMessagePath: /dev/termination-log

imagePullPolicy: Always

restartPolicy: Always

terminationGracePeriodSeconds: 30

dnsPolicy: ClusterFirst

nodeSelector:

${REDIS\_NODE\_LABEL}: 'true'

hostNetwork: true

securityContext: {}

parameters:

- description: Port for Redis Server, default is 6379.

displayName: Redis Port

name: REDIS\_PORT

value: "6379"

required: true

- description: Password for the Redis connection user.

displayName: Redis Connection Password

name: REDIS\_PASSWORD

value: "welcome1"

required: true

- description: Maximum amount of memory the container can use.

displayName: Memory Limit

name: MEMORY\_LIMIT

required: true

value: 64Mi

- description: The name of redis server.

displayName: Redis Database Name

name: REDIS\_DATABASE\_NAME

required: true

value: ""

- description: The label on openshift node the container will be deployed.

displayName: Redis Node Label

name: REDIS\_NODE\_LABEL

required: true

value: ""