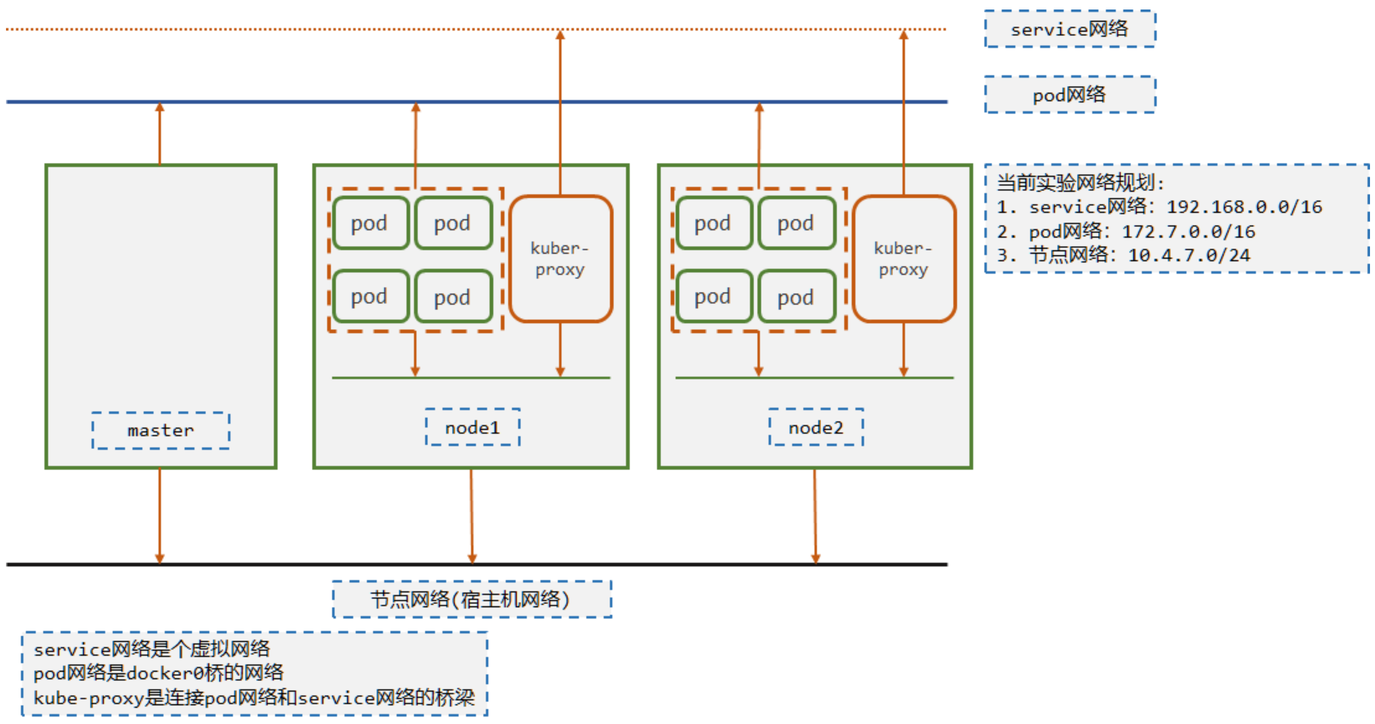
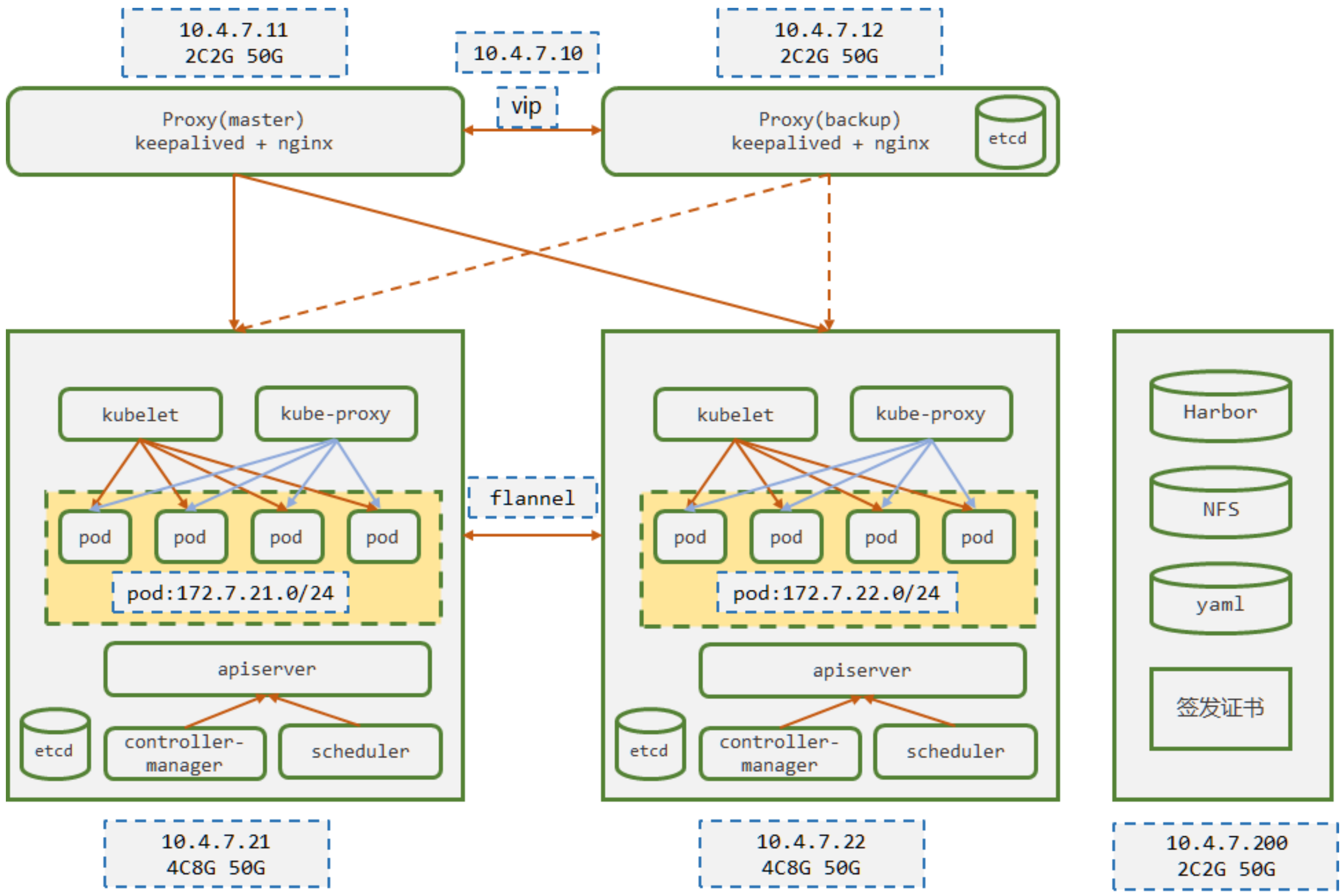
K8S二进制方式安装

1. 实验环境


service网络
pod网络
当前实验网络规划：
1. service网络：192.168.0.0/16
2. pod网络：172.7.0.0/16
3. 节点网络：10.4.7.0/24
pod
kuber-proxy
master
node1
node2
节点网络(宿主机网络)
service网络是个虚拟网络
pod网络是docker0桥的网络
kube-proxy是连接pod网络和service网络的桥梁



Role	HOSTNAME	IP	CPU	MEM	OS	DISK
LB,DNS	hdss7-11.host.com	10.4.7.11	2C	2G	Centos7.5	/data/ 50G
LB,ETCD	hdss7-12.host.com	10.4.7.12	2C	2G	Centos7.5	/data/ 50G
K8S Master,K8S Node,ETCD	hdss7-21.host.com	10.4.7.21	4C	8G	Centos7.5	/data/ 50G
K8S Master,K8S Node,ETCD	hdss7-22.host.com	10.4.7.21	4C	8G	Centos7.5	/data/ 50G
Harbor,NFS	hdss7-200.host.com	10.4.7.200	2C	2G	Centos7.5	/data/ 50G

2. 安装前准备

2.1. 环境准备

所有机器都需要执行

```
[root@hdss7-11 ~]# systemctl stop firewalld
[root@hdss7-11 ~]# systemctl disable firewalld
[root@hdss7-11 ~]# setenforce 0
[root@hdss7-11 ~]# sed -ir '/^SELINUX=/s/=.+/=disabled/' /etc/selinu
```

```
x/config

[root@hdss7-11 ~]# yum install -y epel-release
[root@hdss7-11 ~]# yum install -y wget net-tools telnet tree nmap sys
stat lrzsz dos2unix bind-utils vim less
```

2.2. bind安装

2.2.1. hdss7-11 安装bind

```
[root@hdss7-11 ~]# yum install -y bind
```

2.2.2. hdss7-11 配置bind

- 主配置文件

```
[root@hdss7-11 ~]# vim /etc/named.conf  # 确保以下配置正确
  listen-on port 53 { 10.4.7.11; };
    directory   "/var/named";
    allow-query     { any; };
  forwarders      { 10.4.7.254; };
  recursion yes;
  dnssec-enable no;
  dnssec-validation no;
```

- 在 hdss7-11.host.com 配置区域文件

```
# 增加两个zone配置，od.com为业务域，host.com.zone为主机域
[root@hdss7-11 ~]# vim /etc/named.rfc1912.zones
zone "host.com" IN {
        type  master;
        file  "host.com.zone";
        allow-update { 10.4.7.11; };
};

zone "od.com" IN {
        type  master;
        file  "od.com.zone";
        allow-update { 10.4.7.11; };
};
```

- 在 hdss7-11.host.com 配置主机域文件

```
# line6中时间需要修改
[root@hdss7-11 ~]# vim /var/named/host.com.zone
$ORIGIN host.com.
$TTL 600    ; 10 minutes
@       IN SOA  dns.host.com. dnsadmin.host.com. (
                2020010501 ; serial
                10800      ; refresh (3 hours)
                900        ; retry (15 minutes)
                604800     ; expire (1 week)
                86400      ; minimum (1 day)
                )
            NS   dns.host.com.
$TTL 60 ; 1 minute
dns                A    10.4.7.11
HDSS7-11           A    10.4.7.11
HDSS7-12           A    10.4.7.12
HDSS7-21           A    10.4.7.21
HDSS7-22           A    10.4.7.22
HDSS7-200          A    10.4.7.200
```

- 在 hdss7-11.host.com 配置业务域文件

```
[root@hdss7-11 ~]# vim /var/named/od.com.zone
$ORIGIN od.com.
```

```
$TTL 600	; 10 minutes
@           IN SOA  dns.od.com. dnsadmin.od.com. (
                    2020010501 ; serial
                    10800      ; refresh (3 hours)
                    900        ; retry (15 minutes)
                    604800     ; expire (1 week)
                    86400      ; minimum (1 day)
                    )
                    NS   dns.od.com.
$TTL 60 ; 1 minute
dns                 A    10.4.7.11
```

- 在 hdss7-11.host.com 启动bind服务，并测试

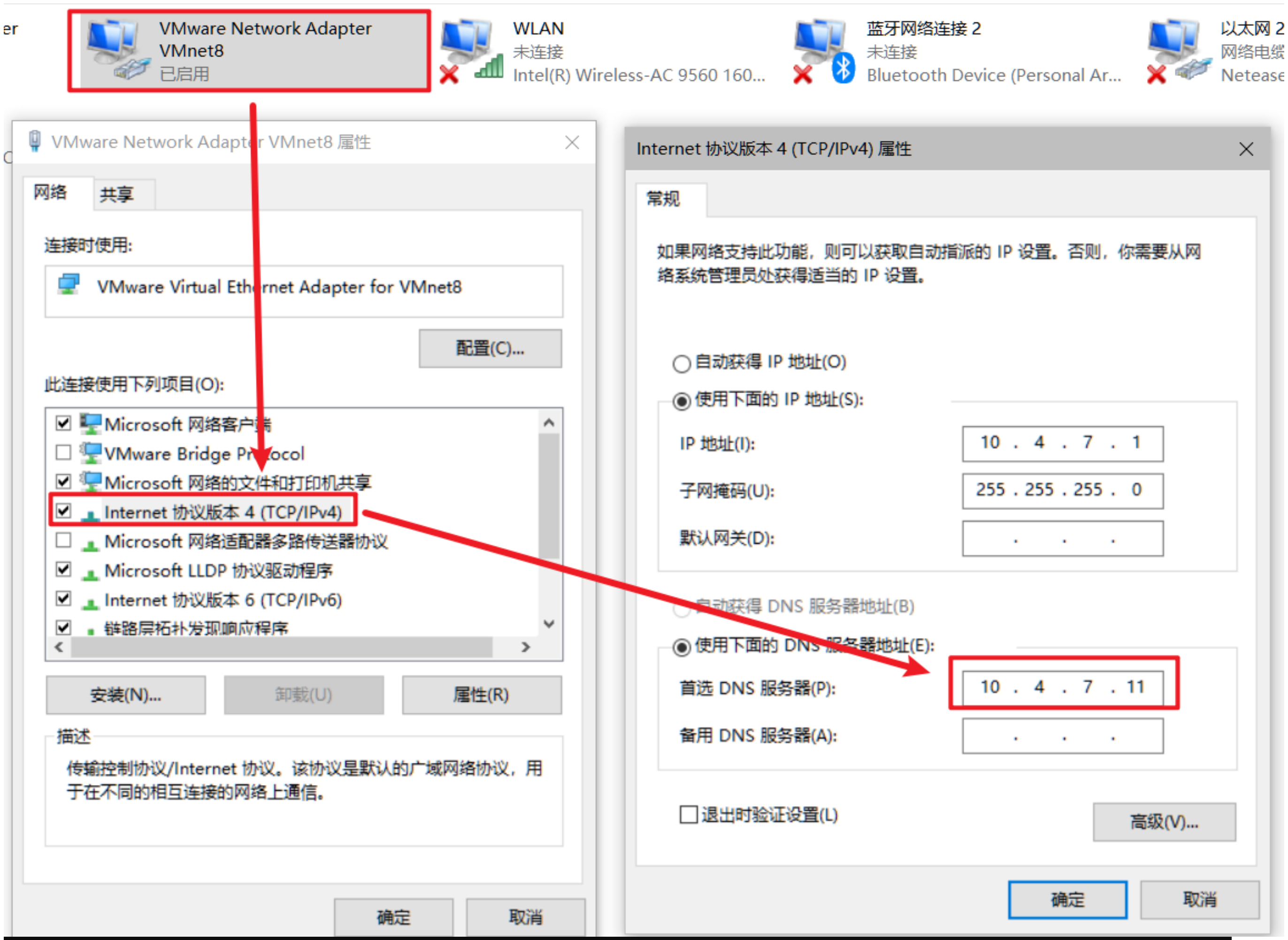
```
[root@hdss7-11 ~]# named-checkconf  # 检查配置文件
[root@hdss7-11 ~]# systemctl start named ; systemctl enable named
[root@hdss7-11 ~]# host HDSS7-200 10.4.7.11
Using domain server:
Name: 10.4.7.11
Address: 10.4.7.11#53
Aliases:

HDSS7-200.host.com has address 10.4.7.200
```

2.2.3. 修改主机DNS

- 修改**所有主机**的dns服务器地址

```
[root@hdss7-11 ~]# sed -i '/DNS1/s/10.4.7.254/10.4.7.11/' /etc/sysconfig/network-scripts/ifcfg-ens32
[root@hdss7-11 ~]# systemctl restart network
[root@hdss7-11 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search host.com
nameserver 10.4.7.11
```

- 本次实验环境使用的是虚拟机，因此也要对windows宿主机NAT网卡DNS进行修改


VMware Network Adapter VMnet8
已启用
WLAN
未连接
Intel(R) Wireless-AC 9560 160...
蓝牙网络连接 2
未连接
Bluetooth Device (Personal Ar...
以太网 2
网络电缆
Netease
VMware Network Adapter VMnet8 属性
网络
共享
连接时使用:
VMware Virtual Ethernet Adapter for VMnet8
配置(C)...
此连接使用下列项目(O):
安装(N)...
卸载(U)
属性(R)
描述
确定
取消
Internet 协议版本 4 (TCP/IPv4) 属性
常规
高级(V)...
确定
取消

此连接使用下列项目(O):

- [✓] Microsoft 网络客户端
- [ ] VMware Bridge Protocol
- [✓] Microsoft 网络的文件和打印机共享
- [✓] Internet 协议版本 4 (TCP/IPv4)
- [ ] Microsoft 网络适配器多路传送器协议
- [✓] Microsoft LLDP 协议驱动程序
- [✓] Internet 协议版本 6 (TCP/IPv6)
- [✓] 链路层拓扑发现响应程序

描述：传输控制协议/Internet 协议。该协议是默认的广域网络协议，用于在不同的相互连接的网络上通信。

Internet 协议版本 4 (TCP/IPv4) 属性

如果网络支持此功能，则可以获取自动指派的 IP 设置。否则，你需要从网络系统管理员处获得适当的 IP 设置。

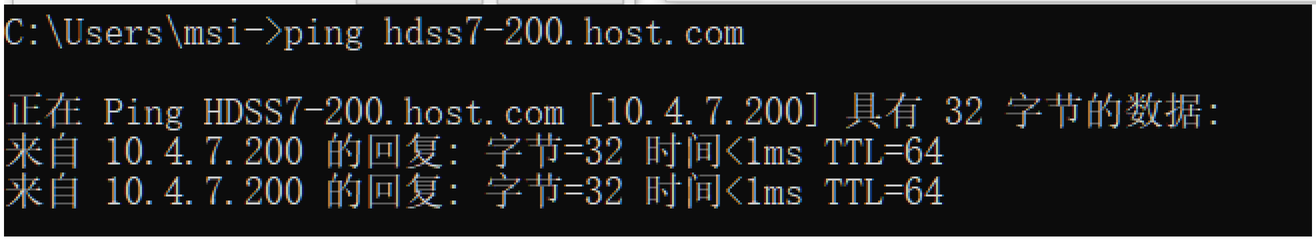
自动获得 IP 地址(O) [✓ 使用下面的 IP 地址(S):]

IP 地址(I): 10 . 4 . 7 . 1
子网掩码(U): 255 . 255 . 255 . 0
默认网关(D): . . .

自动获得 DNS 服务器地址(B) [✓ 使用下面的 DNS 服务器地址(E):]

首选 DNS 服务器(P): 10 . 4 . 7 . 11
备用 DNS 服务器(A): . . .

[ ] 退出时验证设置(L)



```
C:\Users\msi->ping hdss7-200.host.com

正在 Ping HDSS7-200.host.com [10.4.7.200] 具有 32 字节的数据:
来自 10.4.7.200 的回复: 字节=32 时间<1ms TTL=64
来自 10.4.7.200 的回复: 字节=32 时间<1ms TTL=64
```

2.3. 根证书准备

- 在 hdss7-200 下载工具

```
[root@hdss7-200 ~]# wget https://pkg.cfssl.org/R1.2/cfssl_linux-amd64 -O /usr/local/bin/cfssl
[root@hdss7-200 ~]# wget https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64 -O /usr/local/bin/cfssl-json
```

```
[root@hdss7-200 ~]# wget https://pkg.cfssl.org/R1.2/cfssl-certinfo_linux-amd64 -O /usr/local/bin/cfssl-certinfo
[root@hdss7-200 ~]# chmod u+x /usr/local/bin/cfssl*
```

- 在 hdss7-200 签发根证书

```
[root@hdss7-200 ~]# mkdir /opt/certs/ ; cd /opt/certs/
# 根证书配置:
# CN 一般写域名，浏览器会校验
# names 为地区和公司信息
# expiry 为过期时间
[root@hdss7-200 certs]# vim /opt/certs/ca-csr.json
{
    "CN": "AI",
    "hosts": [
    ],
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "jiangsu",
            "L": "wuxi",
            "O": "JNU",
            "OU": "AI"
        }
    ],
    "ca": {
        "expiry": "175200h"
    }
}
[root@hdss7-200 certs]# cfssl gencert -initca ca-csr.json | cfssl-json -bare ca
2020/01/05 10:42:07 [INFO] generating a new CA key and certificate from CSR
2020/01/05 10:42:07 [INFO] generate received request
```

```
2020/01/05 10:42:07 [INFO] received CSR
2020/01/05 10:42:07 [INFO] generating key: rsa-2048
2020/01/05 10:42:08 [INFO] encoded CSR
2020/01/05 10:42:08 [INFO] signed certificate with serial number 451005524427475354617025362003367427117323539780
[root@hdss7-200 certs]# ls -l ca*
-rw-r--r-- 1 root root  993 Jan  5 10:42 ca.csr
-rw-r--r-- 1 root root  328 Jan  5 10:39 ca-csr.json
-rw------- 1 root root 1675 Jan  5 10:42 ca-key.pem
-rw-r--r-- 1 root root 1346 Jan  5 10:42 ca.pem
```

2.4. docker环境准备

需要安装docker的机器：hdss7-21 hdss7-22 hdss7-200，以hdss7-21为例

```
[root@hdss7-21 ~]# wget -O /etc/yum.repos.d/docker-ce.repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
[root@hdss7-21 ~]# yum install -y docker-ce
[root@hdss7-21 ~]# mkdir /etc/docker/
# 不安全的registry中增加了harbor地址
# 各个机器上bip网段不一致，bip中间两段与宿主机最后两段相同，目的是方便定位问题
[root@hdss7-21 ~]# vim /etc/docker/daemon.json
{
  "graph": "/data/docker",
  "storage-driver": "overlay2",
  "insecure-registries": ["registry.access.redhat.com","quay.io","harbor.od.com"],
  "registry-mirrors": ["https://registry.docker-cn.com"],
  "bip": "172.7.21.1/24",
  "exec-opts": ["native.cgroupdriver=systemd"],
  "live-restore": true
}
[root@hdss7-21 ~]# mkdir /data/docker
[root@hdss7-21 ~]# systemctl start docker ; systemctl enable docker
```

2.5. harbor安装

参考地址：https://www.yuque.com/duduniao/trp3ic/ohrxds#9Zpxx
官方地址：https://goharbor.io/
下载地址：https://github.com/goharbor/harbor/releases

2.5.1. hdss7-200 安装harbor

```
# 目录说明：
# /opt/src : 源码、文件下载目录
# /opt/release : 各个版本软件存放位置
# /opt/apps : 各个软件当前版本的软链接
[root@hdss7-200 ~]# cd /opt/src
[root@hdss7-200 src]# wget https://github.com/goharbor/harbor/releases/download/v1.9.4/harbor-offline-installer-v1.9.4.tgz
[root@hdss7-200 src]# mv harbor /opt/release/harbor-v1.9.4
[root@hdss7-200 src]# ln -s /opt/release/harbor-v1.9.4 /opt/apps/harbor
[root@hdss7-200 src]# ll /opt/apps/
total 0
lrwxrwxrwx 1 root root 26 Jan  5 11:13 harbor -> /opt/release/harbor-v1.9.4
# 实验环境仅修改以下配置项，生产环境还得修改密码
[root@hdss7-200 src]# vim /opt/apps/harbor/harbor.yml
hostname: harbor.od.com
http:
  port: 180
data_volume: /data/harbor
location: /data/harbor/logs
[root@hdss7-200 src]# yum install -y docker-compose
[root@hdss7-200 src]# cd /opt/apps/harbor/
[root@hdss7-200 harbor]# ./install.sh
......
✔ ----Harbor has been installed and started successfully.----
[root@hdss7-200 harbor]# docker-compose ps
      Name                     Command                  State
Ports
```

```
--------------------------------------------------------------------
------------------
harbor-core         /harbor/harbor_core              Up
harbor-db           /docker-entrypoint.sh            Up      5432/tc
p
harbor-jobservice   /harbor/harbor_jobservice  ...   Up
harbor-log          /bin/sh -c /usr/local/bin/ ...   Up      127.0.0
.1:1514->10514/tcp
harbor-portal       nginx -g daemon off;             Up      8080/tc
p
nginx               nginx -g daemon off;             Up      0.0.0.0
:180->8080/tcp
redis               redis-server /etc/redis.conf     Up      6379/tc
p
registry            /entrypoint.sh /etc/regist ...   Up      5000/tc
p
registryctl         /harbor/start.sh                 Up
```

- 设置harbor开机启动

```
[root@hdss7-200 harbor]# vim /etc/rc.d/rc.local  # 增加以下内容
# start harbor
cd /opt/apps/harbor
/usr/bin/docker-compose stop
/usr/bin/docker-compose start
```

2.5.2. hdss7-200 安装nginx

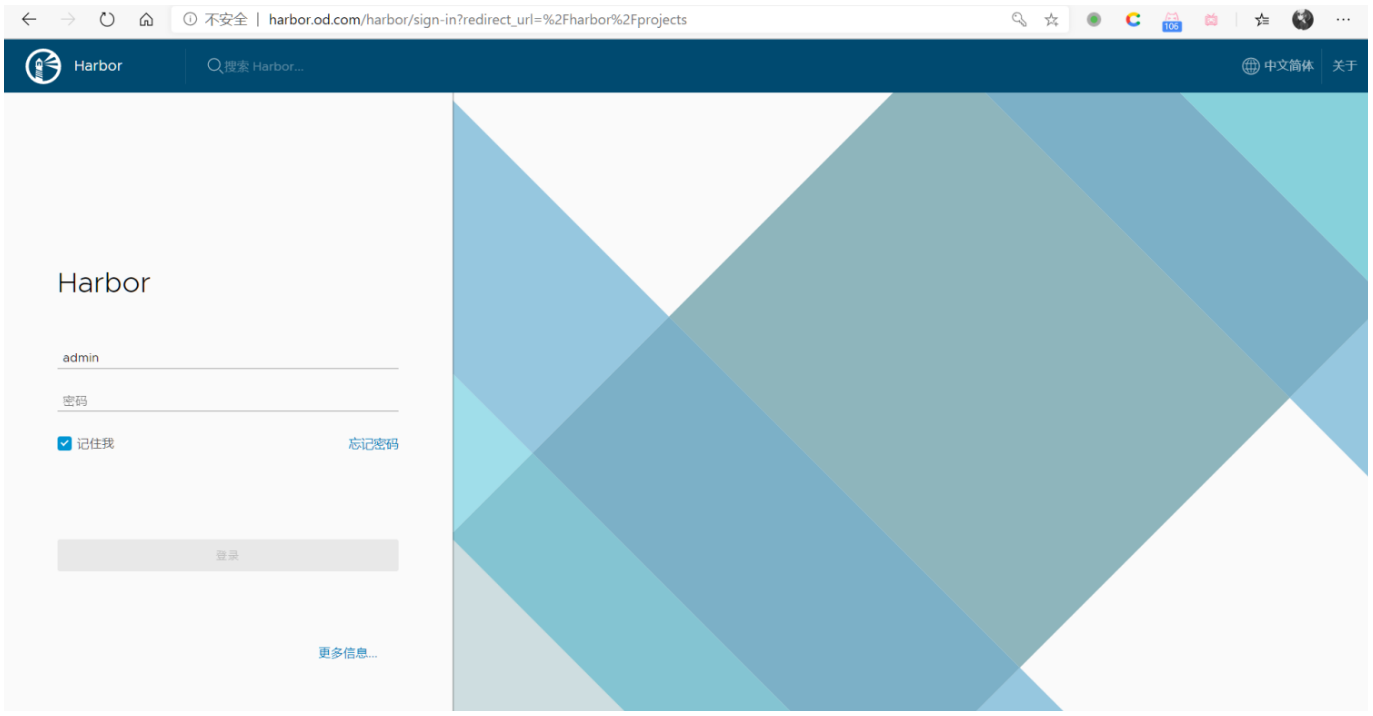
- 安装Nginx反向代理harbor

```
# 当前机器中Nginx功能较少，使用yum安装即可。如有多个harbor考虑源码编译且配置健康
检查
# nginx配置此处忽略，仅仅使用最简单的配置。
[root@hdss7-200 harbor]# vim /etc/nginx/conf.d/harbor.conf
[root@hdss7-200 harbor]# cat /etc/nginx/conf.d/harbor.conf
```

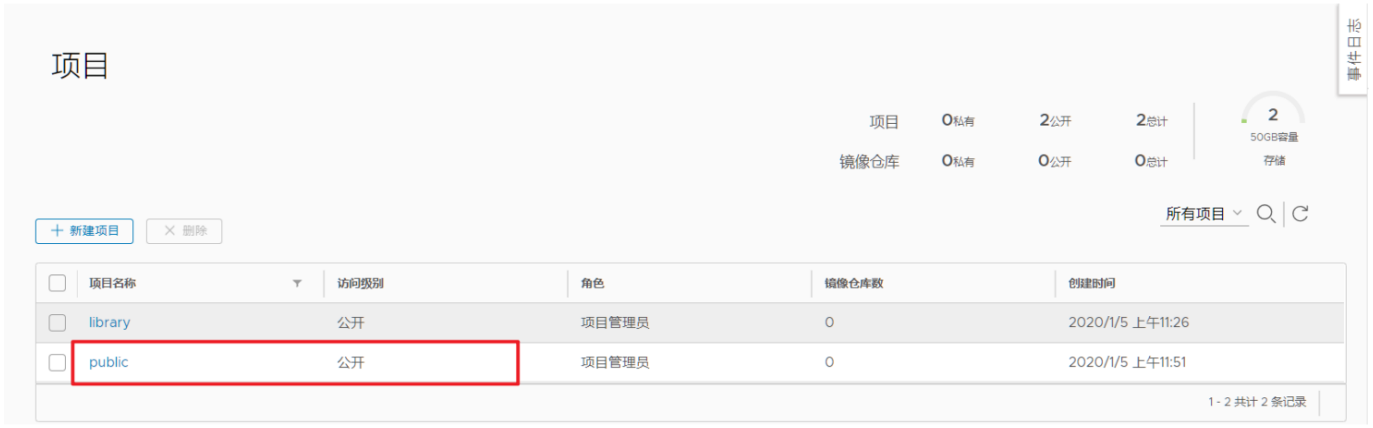
```
server {
    listen       80;
    server_name  harbor.od.com;
    # 避免出现上传失败的情况
    client_max_body_size 1000m;

    location / {
        proxy_pass http://127.0.0.1:180;
    }
}
[root@hdss7-200 harbor]# systemctl start nginx ; systemctl enable nginx
```

- hdss7-11 配置DNS解析

```
[root@hdss7-11 ~]# vim /var/named/od.com.zone  # 序列号需要滚动一个
$ORIGIN od.com.
$TTL 600    ; 10 minutes
@           IN SOA  dns.od.com. dnsadmin.od.com. (
                2020010502 ; serial
                10800      ; refresh (3 hours)
                900        ; retry (15 minutes)
                604800     ; expire (1 week)
                86400      ; minimum (1 day)
                )
                NS   dns.od.com.
$TTL 60 ; 1 minute
dns                A    10.4.7.11
harbor             A    10.4.7.200
[root@hdss7-11 ~]# systemctl restart named.service  # reload 无法使得配置生效
[root@hdss7-11 ~]# host harbor.od.com
harbor.od.com has address 10.4.7.200
```



- 新建项目：public

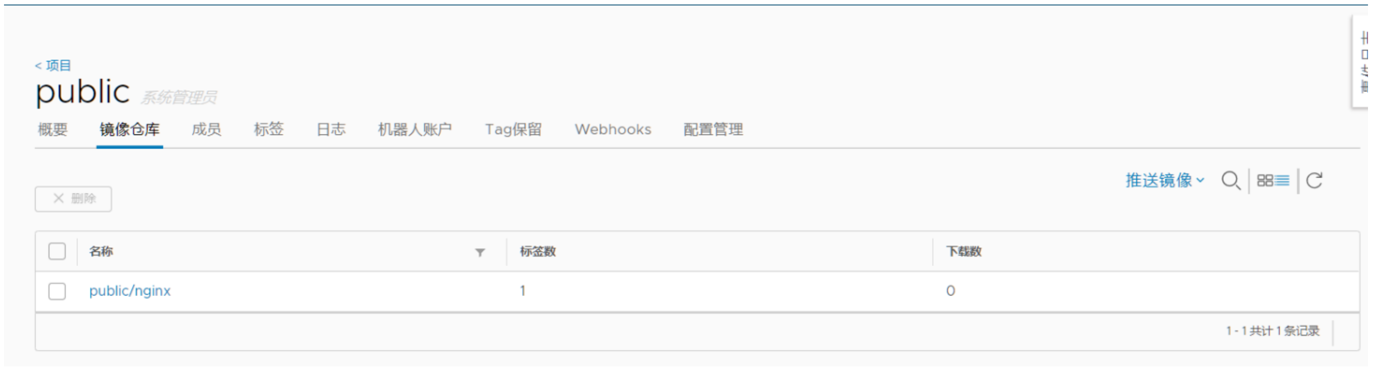


项目名称	访问级别	角色	镜像仓库数	创建时间
library	公开	项目管理员	0	2020/1/5 上午11:26
public	公开	项目管理员	0	2020/1/5 上午11:51

1 - 2 共计 2 条记录

- 测试harbor

```
[root@hdss7-21 ~]# docker image tag nginx:latest harbor.od.com/public/nginx:latest
[root@hdss7-21 ~]# docker login -u admin harbor.od.com
[root@hdss7-21 ~]# docker image push harbor.od.com/public/nginx:latest
[root@hdss7-21 ~]# docker logout
```



名称	标签数	下载数
public/nginx	1	0

1 - 1 共计 1 条记录

3. 主控节点安装

3.1. etcd安装

etcd 的leader选举机制，要求至少为3台或以上的奇数台。本次安装涉及：hdss7-12，hdss7-21，hdss7-22

3.1.1. 签发etcd证书

证书签发服务器 hdss7-200:

- 创建ca的json配置: /opt/certs/ca-config.json
  - server 表示服务端连接客户端时携带的证书，用于客户端验证服务端身份
  - client 表示客户端连接服务端时携带的证书，用于服务端验证客户端身份
  - peer 表示相互之间连接时使用的证书，如etcd节点之间验证

```
{
    "signing": {
        "default": {
            "expiry": "175200h"
        },
        "profiles": {
            "server": {
                "expiry": "175200h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "server auth"
                ]
            },
            "client": {
                "expiry": "175200h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "client auth"
                ]
            },
            "peer": {
                "expiry": "175200h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "server auth",
                    "client auth"
                ]
            }
        }
    }
}
```

- 创建etcd证书配置：/opt/certs/etcd-peer-csr.json

重点在hosts上，将所有可能的etcd服务器添加到host列表，不能使用网段，新增etcd服务器需要重新签发证书

```
{
    "CN": "k8s-etcd",
    "hosts": [
        "10.4.7.11",
        "10.4.7.12",
        "10.4.7.21",
        "10.4.7.22"
    ],
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "jiangsu",
            "L": "wuxi",
            "O": "JNU",
            "OU": "AI"
        }
    ]
}
```

- 签发证书

```
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=peer etcd-peer-csr.json |cfssl-json -bare etcd-peer
[root@hdss7-200 certs]# ll etcd-peer*
-rw-r--r-- 1 root root 1062 Jan  5 17:01 etcd-peer.csr
-rw-r--r-- 1 root root  363 Jan  5 16:59 etcd-peer-csr.json
-rw------- 1 root root 1675 Jan  5 17:01 etcd-peer-key.pem
-rw-r--r-- 1 root root 1428 Jan  5 17:01 etcd-peer.pem
```

3.1.2. 安装etcd

etcd地址：https://github.com/etcd-io/etcd/
实验使用版本: etcd-v3.1.20-linux-amd64.tar.gz
本次安装涉及：hdss7-12，hdss7-21，hdss7-22

- 下载etcd

```
[root@hdss7-12 ~]# useradd -s /sbin/nologin -M etcd
[root@hdss7-12 ~]# cd /opt/src/
[root@hdss7-12 src]# wget https://github.com/etcd-io/etcd/releases/download/v3.1.20/etcd-v3.1.20-linux-amd64.tar.gz
[root@hdss7-12 src]# tar -xf etcd-v3.1.20-linux-amd64.tar.gz
[root@hdss7-12 src]# mv etcd-v3.1.20-linux-amd64 /opt/release/etcd-v3.1.20
[root@hdss7-12 src]# ln -s /opt/release/etcd-v3.1.20 /opt/apps/etcd
[root@hdss7-12 src]# ll /opt/apps/etcd
lrwxrwxrwx 1 root root 25 Jan  5 17:56 /opt/apps/etcd -> /opt/release/etcd-v3.1.20
[root@hdss7-12 src]# mkdir -p /opt/apps/etcd/certs /data/etcd /data/logs/etcd-server
```

- 下发证书到各个etcd上

```
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# for i in 12 21 22;do scp ca.pem etcd-peer.pem etcd-peer-key.pem hdss7-${i}:/opt/apps/etcd/certs/ ;done
```

```
[root@hdss7-12 src]# md5sum /opt/apps/etcd/certs/*
8778d0c3411891af61a287e49a70c89a  /opt/apps/etcd/certs/ca.pem
7918783c2f6bf69e96edf03e67d04983  /opt/apps/etcd/certs/etcd-peer-key.pem
d4d849751a834c7727d42324fdedf92d  /opt/apps/etcd/certs/etcd-peer.pem
```

- 创建启动脚本(部分参数每台机器不同)

```
[root@hdss7-12 ~]# vim /opt/apps/etcd/etcd-server-startup.sh
#!/bin/sh
# listen-peer-urls etcd节点之间通信端口
# listen-client-urls 客户端与etcd通信端口
# quota-backend-bytes 配额大小
# 需要修改的参数：name,listen-peer-urls,listen-client-urls,initial-advertise-peer-urls
```

```

WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/etcd/etcd --name etcd-server-7-12 \
    --data-dir /data/etcd/etcd-server \
    --listen-peer-urls https://10.4.7.12:2380 \
    --listen-client-urls https://10.4.7.12:2379,http://127.0.0.1:2379 \
    --quota-backend-bytes 8000000000 \
    --initial-advertise-peer-urls https://10.4.7.12:2380 \
    --advertise-client-urls https://10.4.7.12:2379,http://127.0.0.1:2379 \
    --initial-cluster  etcd-server-7-12=https://10.4.7.12:2380,etcd-server-7-21=https://10.4.7.21:2380,etcd-server-7-22=https://10.4.7.22:2380 \
    --ca-file ./certs/ca.pem \
    --cert-file ./certs/etcd-peer.pem \
    --key-file ./certs/etcd-peer-key.pem \
    --client-cert-auth  \
    --trusted-ca-file ./certs/ca.pem \
    --peer-ca-file ./certs/ca.pem \
    --peer-cert-file ./certs/etcd-peer.pem \
    --peer-key-file ./certs/etcd-peer-key.pem \
    --peer-client-cert-auth \
    --peer-trusted-ca-file ./certs/ca.pem \
    --log-output stdout
```

```
[root@hdss7-12 ~]# chmod u+x /opt/apps/etcd/etcd-server-startup.sh
[root@hdss7-12 ~]# chown -R etcd.etcd /opt/apps/etcd/ /data/etcd /data/logs/etcd-server
```

3.1.3. 启动etcd

因为这些进程都是要启动为后台进程，要么手动启动，要么采用后台进程管理工具，实验中使用后台管理工具

```
[root@hdss7-12 ~]# yum install -y supervisor
[root@hdss7-12 ~]# systemctl start supervisord ; systemctl enable supervisord
[root@hdss7-12 ~]# vim /etc/supervisord.d/etcd-server.ini
[program:etcd-server-7-12]
command=/opt/apps/etcd/etcd-server-startup.sh         ; the program (relative uses PATH, can take args)
numprocs=1                                            ; number of processes copies to start (def 1)
directory=/opt/apps/etcd                              ; directory to cwd to before exec (def no cwd)
autostart=true                                        ; start at supervisord start (default: true)
autorestart=true                                      ; retstart at unexpected quit (default: true)
startsecs=30                                          ; number of secs prog must stay running (def. 1)
startretries=3                                        ; max # of serial start failures (default 3)
exitcodes=0,2                                         ; 'expected' exit codes for process (default 0,2)
stopsignal=QUIT                                       ; signal used to kill process (default TERM)
stopwaitsecs=10                                       ; max num secs to wait b4 SIGKILL (default 10)
user=etcd                                             ; setuid to this UNIX account to run the program
redirect_stderr=true                                  ; redirect proc stderr to stdout (default false)
stdout_logfile=/data/logs/etcd-server/etcd.stdout.log ; stdout log path, NONE for none; default AUTO
stdout_logfile_maxbytes=64MB                          ; max # logfile bytes b4 rotation (default 50MB)
stdout_logfile_backups=5                              ; # of stdout logfile backups (default 10)
stdout_capture_maxbytes=1MB                           ; number of bytes in 'capturemode' (default 0)
stdout_events_enabled=false                           ; emit events
```

```
 on stdout writes (default false)
[root@hdss7-12 ~]# supervisorctl update
etcd-server-7-12: added process group
```

- etcd 进程状态查看

```
[root@hdss7-12 ~]# supervisorctl status  # supervisorctl 状态
etcd-server-7-12                 RUNNING   pid 22375, uptime 0:00:39

[root@hdss7-12 ~]# netstat -lntp|grep etcd
tcp        0      0 10.4.7.12:2379          0.0.0.0:*
 LISTEN      22379/etcd
tcp        0      0 127.0.0.1:2379          0.0.0.0:*
 LISTEN      22379/etcd
tcp        0      0 10.4.7.12:2380          0.0.0.0:*
 LISTEN      22379/etcd

[root@hdss7-12 ~]# /opt/apps/etcd/etcdctl member list # 随着etcd重启，leader会变化
988139385f78284: name=etcd-server-7-22 peerURLs=https://10.4.7.22:2380 clientURLs=http://127.0.0.1:2379,https://10.4.7.22:2379 isLeader=false
5a0ef2a004fc4349: name=etcd-server-7-21 peerURLs=https://10.4.7.21:2380 clientURLs=http://127.0.0.1:2379,https://10.4.7.21:2379 isLeader=true
f4a0cb0a765574a8: name=etcd-server-7-12 peerURLs=https://10.4.7.12:2380 clientURLs=http://127.0.0.1:2379,https://10.4.7.12:2379 isLeader=false

[root@hdss7-12 ~]# /opt/apps/etcd/etcdctl cluster-health
member 988139385f78284 is healthy: got healthy result from http://127.0.0.1:2379
member 5a0ef2a004fc4349 is healthy: got healthy result from http://127.0.0.1:2379
member f4a0cb0a765574a8 is healthy: got healthy result from http://127.0.0.1:2379
cluster is healthy
```

- etcd 启停方式

```
[root@hdss7-12 ~]# supervisorctl start etcd-server-7-12
[root@hdss7-12 ~]# supervisorctl stop etcd-server-7-12
[root@hdss7-12 ~]# supervisorctl restart etcd-server-7-12
[root@hdss7-12 ~]# supervisorctl status etcd-server-7-12
```

3.2. apiserver 安装

3.2.1. 下载kubernetes服务端

aipserver 涉及的服务器：hdss7-21，hdss7-22
下载 kubernetes 二进制版本包需要科学上网工具

- 进入kubernetes的github页面: https://github.com/kubernetes/kubernetes
- 进入tags页签: https://github.com/kubernetes/kubernetes/tags
- 选择要下载的版本: https://github.com/kubernetes/kubernetes/releases/tag/v1.15.2
- 点击 CHANGELOG-${version}.md 进入说明页面: https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.15.md#downloads-for-v1152
- 下载Server Binaries: https://dl.k8s.io/v1.15.2/kubernetes-server-linux-amd64.tar.gz

```
[root@hdss7-21 ~]# cd /opt/src
[root@hdss7-21 src]# wget https://dl.k8s.io/v1.15.2/kubernetes-server-linux-amd64.tar.gz

[root@hdss7-21 src]# tar -xf kubernetes-server-linux-amd64.tar.gz
[root@hdss7-21 src]# mv kubernetes /opt/release/kubernetes-v1.15.2
[root@hdss7-21 src]# ln -s /opt/release/kubernetes-v1.15.2 /opt/apps/kubernetes
[root@hdss7-21 src]# ll /opt/apps/kubernetes
lrwxrwxrwx 1 root root 31 Jan  6 12:59 /opt/apps/kubernetes -> /opt/release/kubernetes-v1.15.2
```

```

[root@hdss7-21 src]# cd /opt/apps/kubernetes
[root@hdss7-21 kubernetes]# rm -f kubernetes-src.tar.gz
[root@hdss7-21 kubernetes]# cd server/bin/
[root@hdss7-21 bin]# rm -f *.tar *_tag  # *.tar *_tag 镜像文件
[root@hdss7-21 bin]# ll
total 884636
-rwxr-xr-x 1 root root  43534816 Aug  5 18:01 apiextensions-apiserver
-rwxr-xr-x 1 root root 100548640 Aug  5 18:01 cloud-controller-manager
-rwxr-xr-x 1 root root 200648416 Aug  5 18:01 hyperkube
-rwxr-xr-x 1 root root  40182208 Aug  5 18:01 kubeadm
-rwxr-xr-x 1 root root 164501920 Aug  5 18:01 kube-apiserver
-rwxr-xr-x 1 root root 116397088 Aug  5 18:01 kube-controller-manager
-rwxr-xr-x 1 root root  42985504 Aug  5 18:01 kubectl
-rwxr-xr-x 1 root root 119616640 Aug  5 18:01 kubelet
-rwxr-xr-x 1 root root  36987488 Aug  5 18:01 kube-proxy
-rwxr-xr-x 1 root root  38786144 Aug  5 18:01 kube-scheduler
-rwxr-xr-x 1 root root   1648224 Aug  5 18:01 mounter
```

3.2.2. 签发证书

签发证书 涉及的服务器：hdss7-200

- 签发client证书（apiserver和etcd通信证书）

```
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# vim /opt/certs/client-csr.json
{
    "CN": "k8s-node",
    "hosts": [
    ],
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "jiangsu",
            "L": "wuxi",
            "O": "JNU",
            "OU": "AI"
        }
    ]
}
[root@hdss7-200 certs]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=client client-csr.json |cfssl-json -bare client
2020/01/06 13:42:47 [INFO] generate received request
2020/01/06 13:42:47 [INFO] received CSR
2020/01/06 13:42:47 [INFO] generating key: rsa-2048
2020/01/06 13:42:47 [INFO] encoded CSR
2020/01/06 13:42:47 [INFO] signed certificate with serial number 268276380983442021656020268926931973684313260543
2020/01/06 13:42:47 [WARNING] This certificate lacks a "hosts" field. This makes it unsuitable for
websites. For more information see the Baseline Requirements for the Issuance and Management
of Publicly-Trusted Certificates, v.1.1.6, from the CA/Browser Forum (https://cabforum.org);
specifically, section 10.2.3 ("Information Requirements").
[root@hdss7-200 certs]# ls client* -l
-rw-r--r-- 1 root root  993 Jan  6 13:42 client.csr
-rw-r--r-- 1 root root  280 Jan  6 13:42 client-csr.json
-rw------- 1 root root 1679 Jan  6 13:42 client-key.pem
-rw-r--r-- 1 root root 1363 Jan  6 13:42 client.pem
```

- 签发server证书（apiserver和其它k8s组件通信使用）

```
# hosts中将所有可能作为apiserver的ip添加进去，VIP 10.4.7.10 也要加入
[root@hdss7-200 certs]# vim /opt/certs/apiserver-csr.json
{
    "CN": "k8s-apiserver",
    "hosts": [
        "127.0.0.1",
        "192.168.0.1",
        "kubernetes.default",
        "kubernetes.default.svc",
        "kubernetes.default.svc.cluster",
        "kubernetes.default.svc.cluster.local",
        "10.4.7.10",
        "10.4.7.21",
        "10.4.7.22",
        "10.4.7.23"
    ],
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "jiangsu",
            "L": "wuxi",
            "O": "JNU",
            "OU": "AI"
        }
    ]
}
[root@hdss7-200 certs]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem
 -config=ca-config.json -profile=server apiserver-csr.json |cfssl-js
on -bare apiserver
2020/01/06 13:46:56 [INFO] generate received request
2020/01/06 13:46:56 [INFO] received CSR
2020/01/06 13:46:56 [INFO] generating key: rsa-2048
2020/01/06 13:46:56 [INFO] encoded CSR
2020/01/06 13:46:56 [INFO] signed certificate with serial number 573
076691386375893093727554861295529219004473872
2020/01/06 13:46:56 [WARNING] This certificate lacks a "hosts" field
. This makes it unsuitable for
websites. For more information see the Baseline Requirements for the
Issuance and Management
```

```
of Publicly-Trusted Certificates, v.1.1.6, from the CA/Browser Forum (https://cabforum.org);
specifically, section 10.2.3 ("Information Requirements").
[root@hdss7-200 certs]# ls apiserver* -l
-rw-r--r-- 1 root root 1249 Jan  6 13:46 apiserver.csr
-rw-r--r-- 1 root root  566 Jan  6 13:45 apiserver-csr.json
-rw------- 1 root root 1675 Jan  6 13:46 apiserver-key.pem
-rw-r--r-- 1 root root 1598 Jan  6 13:46 apiserver.pem
```

- 证书下发

```
[root@hdss7-200 certs]# for i in 21 22;do echo hdss7-$i;ssh hdss7-$i "mkdir /opt/apps/kubernetes/server/bin/certs";scp apiserver-key.pem apiserver.pem ca-key.pem ca.pem client-key.pem client.pem hdss7-$i:/opt/apps/kubernetes/server/bin/certs/;done
```

3.2.3. 配置apiserver日志审计

aipserver 涉及的服务器：hdss7-21，hdss7-22

```
[root@hdss7-21 bin]# mkdir /opt/apps/kubernetes/conf
[root@hdss7-21 bin]# vim /opt/apps/kubernetes/conf/audit.yaml # 打开文件后，设置 :set paste，避免自动缩进
apiVersion: audit.k8s.io/v1beta1 # This is required.
kind: Policy
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      # Resource "pods" doesn't match requests to any subresource of
```

```
pods,
      # which is consistent with the RBAC policy.
      resources: ["pods"]
  # Log "pods/log", "pods/status" at Metadata level
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]

  # Don't log requests to a configmap called "controller-leader"
  - level: None
    resources:
    - group: ""
      resources: ["configmaps"]
      resourceNames: ["controller-leader"]

  # Don't log watch requests by the "system:kube-proxy" on endpoints or services
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
    - group: "" # core API group
      resources: ["endpoints", "services"]

  # Don't log authenticated requests to certain non-resource URL paths.
  - level: None
    userGroups: ["system:authenticated"]
    nonResourceURLs:
    - "/api*" # Wildcard matching.
    - "/version"

  # Log the request body of configmap changes in kube-system.
  - level: Request
    resources:
    - group: "" # core API group
      resources: ["configmaps"]
    # This rule only applies to resources in the "kube-system" namespace.
    # The empty string "" can be used to select non-namespaced resources.
    namespaces: ["kube-system"]

  # Log configmap and secret changes in all other namespaces at the Metadata level.
  - level: Metadata
    resources:
    - group: "" # core API group
      resources: ["secrets", "configmaps"]

  # Log all other resources in core and extensions at the Request level.
  - level: Request
    resources:
    - group: "" # core API group
    - group: "extensions" # Version of group should NOT be included.

  # A catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    # Long-running requests like watches that fall under this rule will not
    # generate an audit event in RequestReceived.
    omitStages:
      - "RequestReceived"
```

3.2.4. 配置启动脚本

aipserver 涉及的服务器：hdss7-21，hdss7-22

- 创建启动脚本

```
[root@hdss7-21 bin]# vim /opt/apps/kubernetes/server/bin/kube-apiserver-startup.sh
#!/bin/bash

```

```
WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/kubernetes/server/bin/kube-apiserver \
    --apiserver-count 2 \
    --audit-log-path /data/logs/kubernetes/kube-apiserver/audit-log \
    --audit-policy-file ../../conf/audit.yaml \
    --authorization-mode RBAC \
    --client-ca-file ./certs/ca.pem \
    --requestheader-client-ca-file ./certs/ca.pem \
    --enable-admission-plugins NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,DefaultTolerationSeconds,MutatingAdmissionWebhook,ValidatingAdmissionWebhook,ResourceQuota \
    --etcd-cafile ./certs/ca.pem \
    --etcd-certfile ./certs/client.pem \
    --etcd-keyfile ./certs/client-key.pem \
    --etcd-servers https://10.4.7.12:2379,https://10.4.7.21:2379,https://10.4.7.22:2379 \
    --service-account-key-file ./certs/ca-key.pem \
    --service-cluster-ip-range 192.168.0.0/16 \
    --service-node-port-range 3000-29999 \
    --target-ram-mb=1024 \
    --kubelet-client-certificate ./certs/client.pem \
    --kubelet-client-key ./certs/client-key.pem \
    --log-dir  /data/logs/kubernetes/kube-apiserver \
    --tls-cert-file ./certs/apiserver.pem \
    --tls-private-key-file ./certs/apiserver-key.pem \
    --v 2
```

- 配置supervisor启动配置

```
[root@hdss7-21 bin]# vim /etc/supervisord.d/kube-apiserver.ini
[program:kube-apiserver-7-21]
command=/opt/apps/kubernetes/server/bin/kube-apiserver-startup.sh
numprocs=1
directory=/opt/apps/kubernetes/server/bin
autostart=true
```

```
autorestart=true
startsecs=30
startretries=3
exitcodes=0,2
stopsignal=QUIT
stopwaitsecs=10
user=root
redirect_stderr=true
stdout_logfile=/data/logs/kubernetes/kube-apiserver/apiserver.stdout.log
stdout_logfile_maxbytes=64MB
stdout_logfile_backups=5
stdout_capture_maxbytes=1MB
stdout_events_enabled=false
[root@hdss7-21 bin]# supervisorctl update
[root@hdss7-21 bin]# supervisorctl status
etcd-server-7-21                 RUNNING   pid 23637, uptime 22:26:08
kube-apiserver-7-21              RUNNING   pid 32591, uptime 0:05:37
```

- 启停apiserver

```
[root@hdss7-12 ~]# supervisorctl start kube-apiserver-7-21
[root@hdss7-12 ~]# supervisorctl stop kube-apiserver-7-21
[root@hdss7-12 ~]# supervisorctl restart kube-apiserver-7-21
[root@hdss7-12 ~]# supervisorctl status kube-apiserver-7-21
```

- 查看进程

```
[root@hdss7-21 bin]# netstat -lntp|grep api
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN      32595/kube-apiserve
tcp6       0      0 :::6443                 :::*                    LISTEN      32595/kube-apiserve
[root@hdss7-21 bin]# ps uax|grep kube-apiserver|grep -v grep
root      32591  0.0  0.0 115296  1476 ?        S    20:17   0:00 /bin/bash /opt/apps/kubernetes/server/bin/kube-apiserver-startup.sh
root      32595  3.0  2.3 402720 184892 ?       Sl   20:17   0:16 /opt/apps/kubernetes/server/bin/kube-apiserver --apiserver-count 2 --audit-log-path /data/logs/kubernetes/kube-apiserver/audit-log --audit-policy-file ../../conf/audit.yaml --authorization-mode RBAC --client-ca-file ./certs/ca.pem --requestheader-client-ca-file ./certs/ca.pem --enable-admission-plugins NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,DefaultTolerationSeconds,MutatingAdmissionWebhook,ValidatingAdmissionWebhook,ResourceQuota --etcd-cafile ./certs/ca.pem --etcd-certfile ./certs/client.pem --etcd-keyfile ./certs/client-key.pem --etcd-servers https://10.4.7.12:2379,https://10.4.7.21:2379,https://10.4.7.22:2379 --service-account-key-file ./certs/ca-key.pem --service-cluster-ip-range 192.168.0.0/16 --service-node-port-range 3000-29999 --target-ram-mb=1024 --kubelet-client-certificate ./certs/client.pem --kubelet-client-key ./certs/client-key.pem --log-dir /data/logs/kubernetes/kube-apiserver --tls-cert-file ./certs/apiserver.pem --tls-private-key-file ./certs/apiserver-key.pem --v 2
```

3.3. 配置apiserver L4代理

3.3.1. nginx配置

L4 代理涉及的服务器：hdss7-11，hdss7-12

```
[root@hdss7-11 ~]# yum install -y nginx
[root@hdss7-11 ~]# vim /etc/nginx/nginx.conf
# 末尾加上以下内容，stream 只能加在 main 中
# 此处只是简单配置下nginx，实际生产中，建议进行更合理的配置
stream {
    log_format proxy '$time_local|$remote_addr|$upstream_addr|$protocol|$status|'
                     '$session_time|$upstream_connect_time|$bytes_sent|$bytes_received|'
                     '$upstream_bytes_sent|$upstream_bytes_received' ;

    upstream kube-apiserver {
        server 10.4.7.21:6443     max_fails=3 fail_timeout=30s;
        server 10.4.7.22:6443     max_fails=3 fail_timeout=30s;
    }
    server {
        listen 7443;
        proxy_connect_timeout 2s;
        proxy_timeout 900s;
        proxy_pass kube-apiserver;
        access_log /var/log/nginx/proxy.log proxy;
    }
}
[root@hdss7-11 ~]# systemctl start nginx; systemctl enable nginx
[root@hdss7-11 ~]# curl 127.0.0.1:7443  # 测试几次
Client sent an HTTP request to an HTTPS server.
[root@hdss7-11 ~]# cat /var/log/nginx/proxy.log
06/Jan/2020:21:00:27 +0800|127.0.0.1|10.4.7.21:6443|TCP|200|0.001|0.000|76|78|78|76
06/Jan/2020:21:05:03 +0800|127.0.0.1|10.4.7.22:6443|TCP|200|0.020|0.019|76|78|78|76
06/Jan/2020:21:05:04 +0800|127.0.0.1|10.4.7.21:6443|TCP|200|0.001|0.001|76|78|78|76
```

3.3.2. keepalived配置

aipserver L4 代理涉及的服务器：hdss7-11，hdss7-12

- 安装keepalive

```
[root@hdss7-11 ~]# yum install -y keepalived
[root@hdss7-11 ~]# vim /etc/keepalived/check_port.sh # 配置检查脚本
#!/bin/bash
if [ $# -eq 1 ] && [[ $1 =~ ^[0-9]+ ]];then
    [ $(netstat -lntp|grep ":$1 " |wc -l) -eq 0 ] && echo "[ERROR] nginx may be not running!" && exit 1 || exit 0
else
    echo "[ERROR] need one port!"
    exit 1
fi
[root@hdss7-11 ~]# chmod +x /etc/keepalived/check_port.sh
```

- 配置主节点：/etc/keepalived/keepalived.conf

主节点中，必须加上 nopreempt
因为一旦因为网络抖动导致VIP漂移，不能让它自动飘回来，必须要分析原因后手动迁移VIP到主节点！如主节点确认正常后，重启备节点的keepalive，让VIP飘到主节点.
keepalived 的日志输出配置此处省略，生产中需要进行处理。

```
! Configuration File for keepalived
global_defs {
   router_id 10.4.7.11
}
vrrp_script chk_nginx {
    script "/etc/keepalived/check_port.sh 7443"
    interval 2
    weight -20
}
vrrp_instance VI_1 {
    state MASTER
    interface ens32
    virtual_router_id 251
    priority 100
    advert_int 1
    mcast_src_ip 10.4.7.11
    nopreempt

    authentication {
        auth_type PASS
        auth_pass 11111111
    }
    track_script {
         chk_nginx
    }
    virtual_ipaddress {
        10.4.7.10
    }
}
```

- 配置备节点：/etc/keepalived/keepalived.conf

```
! Configuration File for keepalived
global_defs {
    router_id 10.4.7.12
}
vrrp_script chk_nginx {
    script "/etc/keepalived/check_port.sh 7443"
    interval 2
    weight -20
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens32
    virtual_router_id 251
    mcast_src_ip 10.4.7.12
    priority 90
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 11111111
    }
    track_script {
        chk_nginx
    }
    virtual_ipaddress {
        10.4.7.10
    }
}
```

- 启动keepalived

```
[root@hdss7-11 ~]# systemctl start keepalived ; systemctl enable keepalived
```

```
[root@hdss7-11 ~]# ip addr show ens32
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:6d:b8:82 brd ff:ff:ff:ff:ff:ff
    inet 10.4.7.11/24 brd 10.4.7.255 scope global noprefixroute ens32
       valid_lft forever preferred_lft forever
    inet 10.4.7.10/32 scope global ens32
       valid_lft forever preferred_lft forever
......
```

3.4. controller-manager 安装

controller-manager 涉及的服务器：hdss7-21，hdss7-22
controller-manager 设置为只调用当前机器的 apiserver，走127.0.0.1网卡，因此不配制SSL证书

```
[root@hdss7-21 ~]# vim /opt/apps/kubernetes/server/bin/kube-controller-manager-startup.sh
#!/bin/sh
WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/kubernetes/server/bin/kube-controller-manager \
    --cluster-cidr 172.7.0.0/16 \
    --leader-elect true \
    --log-dir /data/logs/kubernetes/kube-controller-manager \
    --master http://127.0.0.1:8080 \
    --service-account-private-key-file ./certs/ca-key.pem \
    --service-cluster-ip-range 192.168.0.0/16 \
    --root-ca-file ./certs/ca.pem \
    --v 2
[root@hdss7-21 ~]# chmod u+x /opt/apps/kubernetes/server/bin/kube-controller-manager-startup.sh
```

```
[root@hdss7-21 ~]# vim /etc/supervisord.d/kube-controller-manager.ini
```

```
[program:kube-controller-manager-7-21]
command=/opt/apps/kubernetes/server/bin/kube-controller-manager-startup.sh                     ; the program (relative uses PATH, can take args)
numprocs=1                                                                       ; number of processes copies to start (def 1)
directory=/opt/apps/kubernetes/server/bin                                        ; directory to cwd to before exec (def no cwd)
autostart=true                                                                   ; start at supervisord start (default: true)
autorestart=true                                                                 ; retstart at unexpected quit (default: true)
startsecs=30                                                                     ; number of secs prog must stay running (def. 1)
startretries=3                                                                   ; max # of serial start failures (default 3)
exitcodes=0,2                                                                    ; 'expected' exit codes for process (default 0,2)
stopsignal=QUIT                                                                  ; signal used to kill process (default TERM)
stopwaitsecs=10                                                                  ; max num secs to wait b4 SIGKILL (default 10)
user=root                                                                        ; setuid to this UNIX account to run the program
redirect_stderr=true                                                             ; redirect proc stderr to stdout (default false)
stdout_logfile=/data/logs/kubernetes/kube-controller-manager/controller.stdout.log  ; stderr log path, NONE for none; default AUTO
stdout_logfile_maxbytes=64MB                                                     ; max # logfile bytes b4 rotation (default 50MB)
stdout_logfile_backups=4                                                         ; # of stdout logfile backups (default 10)
stdout_capture_maxbytes=1MB                                                      ; number of bytes in 'capturemode' (default 0)
stdout_events_enabled=false                                                      ; emit events on stdout writes (default false)
```

```
[root@hdss7-21 ~]# supervisorctl update
```

```
kube-controller-manager-7-21: stopped
kube-controller-manager-7-21: updated process group
[root@hdss7-21 ~]# supervisorctl status
etcd-server-7-21                 RUNNING   pid 23637, uptime 1 day, 0:16:54
kube-apiserver-7-21              RUNNING   pid 32591, uptime 1:56:23
kube-controller-manager-7-21     RUNNING   pid 33357, uptime 0:00:38
```

3.5. kube-scheduler安装

kube-scheduler 涉及的服务器：hdss7-21，hdss7-22
kube-scheduler 设置为只调用当前机器的 apiserver，走127.0.0.1网卡，因此不配制SSL证书

```
[root@hdss7-21 ~]# vim /opt/apps/kubernetes/server/bin/kube-scheduler-startup.sh
#!/bin/sh
WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/kubernetes/server/bin/kube-scheduler \
    --leader-elect  \
    --log-dir /data/logs/kubernetes/kube-scheduler \
    --master http://127.0.0.1:8080 \
    --v 2
[root@hdss7-21 ~]# chmod u+x /opt/apps/kubernetes/server/bin/kube-scheduler-startup.sh
[root@hdss7-21 ~]# mkdir -p /data/logs/kubernetes/kube-scheduler
```

```
[root@hdss7-21 ~]# vim /etc/supervisord.d/kube-scheduler.ini
[program:kube-scheduler-7-21]
command=/opt/apps/kubernetes/server/bin/kube-scheduler-startup.sh
numprocs=1
directory=/opt/apps/kubernetes/server/bin
autostart=true
autorestart=true
```

```
startsecs=30
startretries=3
exitcodes=0,2
stopsignal=QUIT
stopwaitsecs=10
user=root
redirect_stderr=true
stdout_logfile=/data/logs/kubernetes/kube-scheduler/scheduler.stdout.log
stdout_logfile_maxbytes=64MB
stdout_logfile_backups=4
stdout_capture_maxbytes=1MB
stdout_events_enabled=false
```

```
[root@hdss7-21 ~]# supervisorctl update
kube-scheduler-7-21: stopped
kube-scheduler-7-21: updated process group
[root@hdss7-21 ~]# supervisorctl status
etcd-server-7-21                 RUNNING   pid 23637, uptime 1 day, 0:26:53
kube-apiserver-7-21              RUNNING   pid 32591, uptime 2:06:22
kube-controller-manager-7-21     RUNNING   pid 33357, uptime 0:10:37
kube-scheduler-7-21              RUNNING   pid 33450, uptime 0:01:18
```

3.6. 检查主控节点状态

```
[root@hdss7-21 ~]# ln -s /opt/apps/kubernetes/server/bin/kubectl /usr/local/bin/
[root@hdss7-21 ~]# kubectl get cs
NAME                 STATUS    MESSAGE              ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-1               Healthy   {"health": "true"}
etcd-0               Healthy   {"health": "true"}
etcd-2               Healthy   {"health": "true"}
```

```
[root@hdss7-22 ~]# ln -s /opt/apps/kubernetes/server/bin/kubectl /usr/local/bin/
[root@hdss7-22 ~]# kubectl get cs
NAME                 STATUS    MESSAGE              ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-2               Healthy   {"health": "true"}
etcd-1               Healthy   {"health": "true"}
etcd-0               Healthy   {"health": "true"}
```

---

4. 运算节点部署

4.1. kubelet 部署

4.1.1. 签发证书

证书签发在 hdss7-200 操作

```
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# vim kubelet-csr.json # 将所有可能的kubelet机器IP添加到hosts中
{
    "CN": "k8s-kubelet",
    "hosts": [
    "127.0.0.1",
    "10.4.7.10",
    "10.4.7.21",
    "10.4.7.22",
    "10.4.7.23",
    "10.4.7.24",
    "10.4.7.25",
    "10.4.7.26",
    "10.4.7.27",
    "10.4.7.28"
    ],
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "jiangsu",
            "L": "wuxi",
            "O": "JNU",
            "OU": "AI"
        }
    ]
}
[root@hdss7-200 certs]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem
 -config=ca-config.json -profile=server kubelet-csr.json | cfssl-jso
n -bare kubelet
2020/01/06 23:10:56 [INFO] generate received request
2020/01/06 23:10:56 [INFO] received CSR
2020/01/06 23:10:56 [INFO] generating key: rsa-2048
2020/01/06 23:10:56 [INFO] encoded CSR
2020/01/06 23:10:56 [INFO] signed certificate with serial number 612
21942784856969738771370531559555767101820379
2020/01/06 23:10:56 [WARNING] This certificate lacks a "hosts" field
. This makes it unsuitable for
websites. For more information see the Baseline Requirements for the
Issuance and Management
of Publicly-Trusted Certificates, v.1.1.6, from the CA/Browser Forum
(https://cabforum.org);
specifically, section 10.2.3 ("Information Requirements").
[root@hdss7-200 certs]# ls kubelet* -l
-rw-r--r-- 1 root root 1115 Jan  6 23:10 kubelet.csr
-rw-r--r-- 1 root root  452 Jan  6 23:10 kubelet-csr.json
-rw------- 1 root root 1675 Jan  6 23:10 kubelet-key.pem
-rw-r--r-- 1 root root 1468 Jan  6 23:10 kubelet.pem
```

```
[root@hdss7-200 certs]# scp kubelet.pem kubelet-key.pem hdss7-21:/opt/apps/kubernetes/server/bin/certs/
[root@hdss7-200 certs]# scp kubelet.pem kubelet-key.pem hdss7-22:/opt/apps/kubernetes/server/bin/certs/
```

4.1.2. 创建kubelet配置

kubelet配置在 hdss7-21 hdss7-22 操作

- set-cluster # 创建需要连接的集群信息，可以创建多个k8s集群信息

```
[root@hdss7-21 ~]# kubectl config set-cluster myk8s \
--certificate-authority=/opt/apps/kubernetes/server/bin/certs/ca.pem \
--embed-certs=true \
--server=https://10.4.7.10:7443 \
--kubeconfig=/opt/apps/kubernetes/conf/kubelet.kubeconfig
```

- set-credentials # 创建用户账号，即用户登陆使用的客户端私有和证书，可以创建多个证书

```
[root@hdss7-21 ~]# kubectl config set-credentials k8s-node \
--client-certificate=/opt/apps/kubernetes/server/bin/certs/client.pem \
--client-key=/opt/apps/kubernetes/server/bin/certs/client-key.pem \
--embed-certs=true \
--kubeconfig=/opt/apps/kubernetes/conf/kubelet.kubeconfig
```

- set-context # 设置context，即确定账号和集群对应关系

```
[root@hdss7-21 ~]# kubectl config set-context myk8s-context \
--cluster=myk8s \
--user=k8s-node \
--kubeconfig=/opt/apps/kubernetes/conf/kubelet.kubeconfig
```

- use-context # 设置当前使用哪个context

```
[root@hdss7-21 ~]# kubectl config use-context myk8s-context --kubeconfig=/opt/apps/kubernetes/conf/kubelet.kubeconfig
```

4.1.3. 授权k8s-node用户

此步骤只需要在一台master节点执行
授权 k8s-node 用户绑定集群角色 system:node ，让 k8s-node 成为具备运算节点的权限。

```
[root@hdss7-21 ~]# vim k8s-node.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: k8s-node
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:node
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: k8s-node
[root@hdss7-21 ~]# kubectl create -f k8s-node.yaml
clusterrolebinding.rbac.authorization.k8s.io/k8s-node created
[root@hdss7-21 ~]# kubectl get clusterrolebinding k8s-node
NAME       AGE
k8s-node   36s
```

4.1.4. 装备pause镜像

将pause镜像放入到harbor私有仓库中，仅在 hdss7-200 操作：

```
[root@hdss7-200 ~]# docker image pull kubernetes/pause
[root@hdss7-200 ~]# docker image tag kubernetes/pause:latest harbor.od.com/public/pause:latest
[root@hdss7-200 ~]# docker login -u admin harbor.od.com
[root@hdss7-200 ~]# docker image push harbor.od.com/public/pause:latest
```

4.1.5. 创建启动脚本

在node节点创建脚本并启动kubelet，涉及服务器： hdss7-21 hdss7-22

```
[root@hdss7-21 ~]# vim /opt/apps/kubernetes/server/bin/kubelet-startup.sh
#!/bin/sh

WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/kubernetes/server/bin/kubelet \
    --anonymous-auth=false \
    --cgroup-driver systemd \
    --cluster-dns 192.168.0.2 \
    --cluster-domain cluster.local \
    --runtime-cgroups=/systemd/system.slice \
    --kubelet-cgroups=/systemd/system.slice \
    --fail-swap-on="false" \
    --client-ca-file ./certs/ca.pem \
    --tls-cert-file ./certs/kubelet.pem \
    --tls-private-key-file ./certs/kubelet-key.pem \
    --hostname-override hdss7-21.host.com \
    --image-gc-high-threshold 20 \
    --image-gc-low-threshold 10 \
    --kubeconfig ../../conf/kubelet.kubeconfig \
    --log-dir /data/logs/kubernetes/kube-kubelet \
    --pod-infra-container-image harbor.od.com/public/pause:latest \
    --root-dir /data/kubelet
[root@hdss7-21 ~]# chmod u+x /opt/apps/kubernetes/server/bin/kubelet-startup.sh
[root@hdss7-21 ~]# mkdir -p /data/logs/kubernetes/kube-kubelet /data/kubelet

[root@hdss7-21 ~]# vim /etc/supervisord.d/kube-kubelet.ini
[program:kube-kubelet-7-21]
command=/opt/apps/kubernetes/server/bin/kubelet-startup.sh
numprocs=1
directory=/opt/apps/kubernetes/server/bin
autostart=true
autorestart=true
startsecs=30
startretries=3
exitcodes=0,2
stopsignal=QUIT
stopwaitsecs=10
user=root
redirect_stderr=true
stdout_logfile=/data/logs/kubernetes/kube-kubelet/kubelet.stdout.log
stdout_logfile_maxbytes=64MB
stdout_logfile_backups=5
stdout_capture_maxbytes=1MB
stdout_events_enabled=false
```

```
[root@hdss7-21 ~]# supervisorctl update
[root@hdss7-21 ~]# supervisorctl status
etcd-server-7-21                 RUNNING   pid 23637, uptime 1 day, 14:56:25
kube-apiserver-7-21              RUNNING   pid 32591, uptime 16:35:54
kube-controller-manager-7-21     RUNNING   pid 33357, uptime 14:40:09
kube-kubelet-7-21                RUNNING   pid 37232, uptime 0:01:08
kube-scheduler-7-21              RUNNING   pid 33450, uptime 14:30:50
[root@hdss7-21 ~]# kubectl get node
```

```
NAME                STATUS   ROLES    AGE     VERSION
hdss7-21.host.com   Ready    <none>   3m13s   v1.15.2
hdss7-22.host.com   Ready    <none>   3m13s   v1.15.2
```

4.1.6. 修改节点角色

使用 kubectl get nodes 获取的Node节点角色为空，可以按照以下方式修改

```
[root@hdss7-21 ~]# kubectl get node
NAME                STATUS   ROLES    AGE     VERSION
hdss7-21.host.com   Ready    <none>   3m13s   v1.15.2
hdss7-22.host.com   Ready    <none>   3m13s   v1.15.2
[root@hdss7-21 ~]# kubectl label node hdss7-21.host.com node-role.kubernetes.io/node=
node/hdss7-21.host.com labeled
[root@hdss7-21 ~]# kubectl label node hdss7-21.host.com node-role.kubernetes.io/master=
node/hdss7-21.host.com labeled
[root@hdss7-21 ~]# kubectl label node hdss7-22.host.com node-role.kubernetes.io/master=
node/hdss7-22.host.com labeled
[root@hdss7-21 ~]# kubectl label node hdss7-22.host.com node-role.kubernetes.io/node=
node/hdss7-22.host.com labeled
[root@hdss7-21 ~]# kubectl get node
NAME                STATUS   ROLES         AGE     VERSION
hdss7-21.host.com   Ready    master,node   7m44s   v1.15.2
hdss7-22.host.com   Ready    master,node   7m44s   v1.15.2
```

4.2. kube-proxy部署

4.2.1. 签发证书

证书签发在 hdss7-200 操作

```
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# vim kube-proxy-csr.json  # CN 其实是k8s中的角色
{
    "CN": "system:kube-proxy",
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "jiangsu",
            "L": "wuxi",
            "O": "JNU",
            "OU": "AI"
        }
    ]
}
[root@hdss7-200 certs]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=client kube-proxy-csr.json |cfssl-json -bare kube-proxy-client
2020/01/07 21:45:53 [INFO] generate received request
2020/01/07 21:45:53 [INFO] received CSR
2020/01/07 21:45:53 [INFO] generating key: rsa-2048
2020/01/07 21:45:53 [INFO] encoded CSR
2020/01/07 21:45:53 [INFO] signed certificate with serial number 620191685968917036075463174423999296907693104226
2020/01/07 21:45:53 [WARNING] This certificate lacks a "hosts" field. This makes it unsuitable for
websites. For more information see the Baseline Requirements for the Issuance and Management
of Publicly-Trusted Certificates, v.1.1.6, from the CA/Browser Forum (https://cabforum.org);
[root@hdss7-200 certs]# ls kube-proxy-c* -l  # 因为kube-proxy使用的用户是kube-proxy，不能使用client证书，必须要重新签发自己的证书
-rw-r--r-- 1 root root 1005 Jan  7 21:45 kube-proxy-client.csr
```

```
-rw------- 1 root root 1675 Jan  7 21:45 kube-proxy-client-key.pem
-rw-r--r-- 1 root root 1375 Jan  7 21:45 kube-proxy-client.pem
-rw-r--r-- 1 root root  267 Jan  7 21:45 kube-proxy-csr.json

[root@hdss7-200 certs]# scp kube-proxy-client-key.pem kube-proxy-client.pem hdss7-21:/opt/apps/kubernetes/server/bin/certs/
100% 1375   870.6KB/s   00:00
[root@hdss7-200 certs]# scp kube-proxy-client-key.pem kube-proxy-client.pem hdss7-22:/opt/apps/kubernetes/server/bin/certs/
```

4.2.2. 创建kube-proxy配置

在所有node节点创建，涉及服务器：hdss7-21 ，hdss7-22

```
[root@hdss7-21 ~]# kubectl config set-cluster myk8s \
--certificate-authority=/opt/apps/kubernetes/server/bin/certs/ca.pem \
--embed-certs=true \
--server=https://10.4.7.10:7443 \
--kubeconfig=/opt/apps/kubernetes/conf/kube-proxy.kubeconfig

[root@hdss7-21 ~]# kubectl config set-credentials kube-proxy \
--client-certificate=/opt/apps/kubernetes/server/bin/certs/kube-proxy-client.pem \
--client-key=/opt/apps/kubernetes/server/bin/certs/kube-proxy-client-key.pem \
--embed-certs=true \
--kubeconfig=/opt/apps/kubernetes/conf/kube-proxy.kubeconfig

[root@hdss7-21 ~]# kubectl config set-context myk8s-context \
--cluster=myk8s \
--user=kube-proxy \
--kubeconfig=/opt/apps/kubernetes/conf/kube-proxy.kubeconfig

[root@hdss7-21 ~]# kubectl config use-context myk8s-context --kubeconfig=/opt/apps/kubernetes/conf/kube-proxy.kubeconfig
```

4.2.3. 加载ipvs模块

kube-proxy 共有3种流量调度模式，分别是 namespace，iptables，ipvs，其中ipvs性能最好。

```
[root@hdss7-21 ~]# for i in $(ls /usr/lib/modules/$(uname -r)/kernel/net/netfilter/ipvs|grep -o "^[^.]*");do echo $i; /sbin/modinfo -F filename $i >/dev/null 2>&1 && /sbin/modprobe $i;done
[root@hdss7-21 ~]# lsmod | grep ip_vs  # 查看ipvs模块
```

4.2.4. 创建启动脚本

```
[root@hdss7-21 ~]# vim /opt/apps/kubernetes/server/bin/kube-proxy-startup.sh
#!/bin/sh

WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/kubernetes/server/bin/kube-proxy \
  --cluster-cidr 172.7.0.0/16 \
  --hostname-override hdss7-21.host.com \
  --proxy-mode=ipvs \
  --ipvs-scheduler=nq \
  --kubeconfig ../../conf/kube-proxy.kubeconfig
[root@hdss7-21 ~]# chmod u+x /opt/apps/kubernetes/server/bin/kube-proxy-startup.sh
[root@hdss7-21 ~]# mkdir -p /data/logs/kubernetes/kube-proxy
[root@hdss7-21 ~]# vim /etc/supervisord.d/kube-proxy.ini
[program:kube-proxy-7-21]
command=/opt/apps/kubernetes/server/bin/kube-proxy-startup.sh
numprocs=1
directory=/opt/apps/kubernetes/server/bin
autostart=true
```

```
autorestart=true
startsecs=30
startretries=3
exitcodes=0,2
stopsignal=QUIT
stopwaitsecs=10
user=root
redirect_stderr=true
stdout_logfile=/data/logs/kubernetes/kube-proxy/proxy.stdout.log
stdout_logfile_maxbytes=64MB
stdout_logfile_backups=5
stdout_capture_maxbytes=1MB
stdout_events_enabled=false

[root@hdss7-21 ~]# supervisorctl update
```

4.2.5. 验证集群

```
[root@hdss7-21 ~]# supervisorctl status
etcd-server-7-21                 RUNNING   pid 23637, uptime 2 days, 0:27:18
kube-apiserver-7-21              RUNNING   pid 32591, uptime 1 day, 2:06:47
kube-controller-manager-7-21     RUNNING   pid 33357, uptime 1 day, 0:11:02
kube-kubelet-7-21                RUNNING   pid 37232, uptime 9:32:01
kube-proxy-7-21                  RUNNING   pid 47088, uptime 0:06:19
kube-scheduler-7-21              RUNNING   pid 33450, uptime 1 day, 0:01:43

[root@hdss7-21 ~]# yum install -y ipvsadm
[root@hdss7-21 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.1:443 nq
```

```
  -> 10.4.7.21:6443              Masq    1      0           0
  -> 10.4.7.22:6443              Masq    1      0           0
```

```
[root@hdss7-21 ~]# curl -I 172.7.21.2
HTTP/1.1 200 OK
Server: nginx/1.17.6
Date: Tue, 07 Jan 2020 14:28:46 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 19 Nov 2019 12:50:08 GMT
Connection: keep-alive
ETag: "5dd3e500-264"
Accept-Ranges: bytes

[root@hdss7-21 ~]# curl -I 172.7.22.2  # 缺少网络插件，无法跨节点通信
```

---

5. 核心插件部署

5.1. CNI网络插件

kubernetes设计了网络模型，但是pod之间通信的具体实现交给了CNI往插件。常用的CNI网络插件有：Flannel 、Calico、Canal、Contiv等，其中Flannel和Calico占比接近80%，Flannel占比略多于Calico。本次部署使用Flannel作为网络插件。涉及的机器 hdss7-21,hdss7-22

5.1.1. 安装Flannel

github地址：https://github.com/coreos/flannel/releases
涉及的机器 hdss7-21,hdss7-22

```
[root@hdss7-21 ~]# cd /opt/src/
[root@hdss7-21 src]# wget https://github.com/coreos/flannel/releases/download/v0.11.0/flannel-v0.11.0-linux-amd64.tar.gz
[root@hdss7-21 src]# mkdir /opt/release/flannel-v0.11.0 # 因为flannel压缩包内部没有套目录
[root@hdss7-21 src]# tar -xf flannel-v0.11.0-linux-amd64.tar.gz -C /opt/release/flannel-v0.11.0
[root@hdss7-21 src]# ln -s /opt/release/flannel-v0.11.0 /opt/apps/flannel
[root@hdss7-21 src]# ll /opt/apps/flannel
lrwxrwxrwx 1 root root 28 Jan  9 22:33 /opt/apps/flannel -> /opt/release/flannel-v0.11.0
```

5.1.2. 拷贝证书

```
# flannel 需要以客户端的身份访问etcd，需要相关证书
[root@hdss7-21 src]# mkdir /opt/apps/flannel/certs
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# scp ca.pem client-key.pem client.pem hdss7-21:/opt/apps/flannel/certs/
```

5.1.3. 创建启动脚本

涉及的机器 hdss7-21,hdss7-22

```
[root@hdss7-21 src]# vim /opt/apps/flannel/subnet.env # 创建子网信息，7-22的subnet需要修改
FLANNEL_NETWORK=172.7.0.0/16
FLANNEL_SUBNET=172.7.21.1/24
FLANNEL_MTU=1500
FLANNEL_IPMASQ=false
[root@hdss7-21 src]# /opt/apps/etcd/etcdctl set /coreos.com/network/config '{"Network": "172.7.0.0/16", "Backend": {"Type": "host-gw"}}'
```

```
[root@hdss7-21 src]# /opt/apps/etcd/etcdctl get /coreos.com/network/config # 只需要在一台etcd机器上设置就可以了
{"Network": "172.7.0.0/16", "Backend": {"Type": "host-gw"}}
```

```
# public-ip 为本机IP，iface 为当前宿主机对外网卡
[root@hdss7-21 src]# vim /opt/apps/flannel/flannel-startup.sh
#!/bin/sh

WORK_DIR=$(dirname $(readlink -f $0))
[ $? -eq 0 ] && cd $WORK_DIR || exit

/opt/apps/flannel/flanneld \
    --public-ip=10.4.7.21 \
    --etcd-endpoints=https://10.4.7.12:2379,https://10.4.7.21:2379,https://10.4.7.22:2379 \
    --etcd-keyfile=./certs/client-key.pem \
    --etcd-certfile=./certs/client.pem \
    --etcd-cafile=./certs/ca.pem \
    --iface=ens32 \
    --subnet-file=./subnet.env \
    --healthz-port=2401
[root@hdss7-21 src]# chmod u+x /opt/apps/flannel/flannel-startup.sh
```

```
[root@hdss7-21 src]# vim /etc/supervisord.d/flannel.ini
[program:flanneld-7-21]
command=/opt/apps/flannel/flannel-startup.sh                 ; the program (relative uses PATH, can take args)
numprocs=1                                                   ; number of processes copies to start (def 1)
directory=/opt/apps/flannel                                  ; directory to cwd to before exec (def no cwd)
autostart=true                                               ; start at supervisord start (default: true)
autorestart=true                                             ; retstart at unexpected quit (default: true)
startsecs=30                                                 ; number of secs prog must stay running (def. 1)
```

```
startretries=3                                               ; max # of serial start failures (default 3)
exitcodes=0,2                                                ; 'expected' exit codes for process (default 0,2)
stopsignal=QUIT                                              ; signal used to kill process (default TERM)
stopwaitsecs=10                                              ; max num secs to wait b4 SIGKILL (default 10)
user=root                                                    ; setuid to this UNIX account to run the program
redirect_stderr=true                                         ; redirect proc stderr to stdout (default false)
stdout_logfile=/data/logs/flanneld/flanneld.stdout.log       ; stderr log path, NONE for none; default AUTO
stdout_logfile_maxbytes=64MB                                 ; max # logfile bytes b4 rotation (default 50MB)
stdout_logfile_backups=5                                     ; # of stdout logfile backups (default 10)
stdout_capture_maxbytes=1MB                                  ; number of bytes in 'capturemode' (default 0)
stdout_events_enabled=false                                  ; emit events on stdout writes (default false)
[root@hdss7-21 src]# mkdir -p /data/logs/flanneld/
[root@hdss7-21 src]# supervisorctl update
flanneld-7-21: added process group
[root@hdss7-21 src]# supervisorctl status
etcd-server-7-21                 RUNNING   pid 1058, uptime -1 day, 16:33:25
flanneld-7-21                    RUNNING   pid 13154, uptime 0:00:30
kube-apiserver-7-21              RUNNING   pid 1061, uptime -1 day, 16:33:25
kube-controller-manager-7-21     RUNNING   pid 1068, uptime -1 day, 16:33:25
kube-kubelet-7-21                RUNNING   pid 1052, uptime -1 day, 16:33:25
kube-proxy-7-21                  RUNNING   pid 1082, uptime -1 day, 16:33:25
kube-scheduler-7-21              RUNNING   pid 1089, uptime -1 day, 16:33:25
```

5.1.4. 验证跨网络访问

```
[root@hdss7-21 src]# kubectl get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE    IP           NODE                NOMINATED NODE   READINESS GATES
nginx-ds-7db29   1/1     Running   1          2d     172.7.22.2   hdss7-22.host.com   <none>           <none>
nginx-ds-vvsz7   1/1     Running   1          2d     172.7.21.2   hdss7-21.host.com   <none>           <none>
[root@hdss7-21 src]# curl -I 172.7.22.2
HTTP/1.1 200 OK
Server: nginx/1.17.6
Date: Thu, 09 Jan 2020 14:55:21 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 19 Nov 2019 12:50:08 GMT
Connection: keep-alive
ETag: "5dd3e500-264"
Accept-Ranges: bytes
```

5.1.5. 解决pod间IP透传问题

所有Node上操作，即优化NAT网络

```
# 从pod a跨宿主机访问pod b时，在pod b中能看到的地址为 pod a 宿主机地址
[root@nginx-ds-jdp7q /]# tail -f /usr/local/nginx/logs/access.log
10.4.7.22 - - [13/Jan/2020:13:13:39 +0000] "GET / HTTP/1.1" 200 12 "-" "curl/7.29.0"
10.4.7.22 - - [13/Jan/2020:13:14:27 +0000] "GET / HTTP/1.1" 200 12 "-" "curl/7.29.0"
10.4.7.22 - - [13/Jan/2020:13:54:20 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/7.29.0"
10.4.7.22 - - [13/Jan/2020:13:54:25 +0000] "HEAD / HTTP/1.1" 200 0 "-
```

```
" "curl/7.29.0"
[root@hdss7-21 ~]# iptables-save |grep POSTROUTING|grep docker # 引发问题的规则
-A POSTROUTING -s 172.7.21.0/24 ! -o docker0 -j MASQUERADE
```

```
[root@hdss7-21 ~]# yum install -y iptables-services
[root@hdss7-21 ~]# systemctl start iptables.service ; systemctl enable iptables.service
# 需要处理的规则:
[root@hdss7-21 ~]# iptables-save |grep POSTROUTING|grep docker
-A POSTROUTING -s 172.7.21.0/24 ! -o docker0 -j MASQUERADE
[root@hdss7-21 ~]# iptables-save | grep -i reject
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
# 处理方式:
[root@hdss7-21 ~]# iptables -t nat -D POSTROUTING -s 172.7.21.0/24 ! -o docker0 -j MASQUERADE
[root@hdss7-21 ~]# iptables -t nat -I POSTROUTING -s 172.7.21.0/24 ! -d 172.7.0.0/16 ! -o docker0 -j MASQUERADE

[root@hdss7-21 ~]# iptables -t filter -D INPUT -j REJECT --reject-with icmp-host-prohibited
[root@hdss7-21 ~]# iptables -t filter -D FORWARD -j REJECT --reject-with icmp-host-prohibited

[root@hdss7-21 ~]# iptables-save > /etc/sysconfig/iptables
```

```
# 此时跨宿主机访问pod时，显示pod的IP
[root@nginx-ds-jdp7q /]# tail -f /usr/local/nginx/logs/access.log
172.7.22.2 - - [13/Jan/2020:14:15:39 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/7.29.0"
172.7.22.2 - - [13/Jan/2020:14:15:47 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/7.29.0"
172.7.22.2 - - [13/Jan/2020:14:15:48 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/7.29.0"
172.7.22.2 - - [13/Jan/2020:14:15:48 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/7.29.0"
```

5.2. CoreDNS

CoreDNS用于实现 service --> cluster IP 的DNS解析。以容器的方式交付到k8s集群，由k8s自行管理，降低人为操作的复杂度。

5.2.1. 配置yaml文件库

在hdss7-200中配置yaml文件库，后期通过Http方式去使用yaml清单文件。

- 配置nginx虚拟主机(hdss7-200)

```
[root@hdss7-200 ~]# vim /etc/nginx/conf.d/k8s-yaml.od.com.conf
server {
    listen       80;
    server_name  k8s-yaml.od.com;

    location / {
        autoindex on;
        default_type text/plain;
        root /data/k8s-yaml;
    }
}
[root@hdss7-200 ~]# mkdir /data/k8s-yaml;
[root@hdss7-200 ~]# nginx -qt && nginx -s reload
```

- 配置dns解析(hdss7-11)

```
[root@hdss7-11 ~]# vim /var/named/od.com.zone
[root@hdss7-11 ~]# cat /var/named/od.com.zone
$ORIGIN od.com.
$TTL 600    ; 10 minutes
@           IN SOA  dns.od.com. dnsadmin.od.com. (
                2020011301 ; serial
                10800      ; refresh (3 hours)
                900        ; retry (15 minutes)
                604800     ; expire (1 week)
                86400      ; minimum (1 day)
                )
                NS   dns.od.com.
$TTL 60 ; 1 minute
dns                A    10.4.7.11
harbor             A    10.4.7.200
k8s-yaml           A    10.4.7.200
[root@hdss7-11 ~]# systemctl restart named
```

5.2.2. coredns的资源清单文件

清单文件存放到 hdss7-200:/data/k8s-yaml/coredns/coredns_1.6.1/

- rabc.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: coredns
  namespace: kube-system
  labels:
      kubernetes.io/cluster-service: "true"
      addonmanager.kubernetes.io/mode: Reconcile
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
    addonmanager.kubernetes.io/mode: Reconcile
  name: system:coredns
rules:
- apiGroups:
  - ""
  resources:
```

```
  - endpoints
  - services
  - pods
  - namespaces
  verbs:
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
    addonmanager.kubernetes.io/mode: EnsureExists
  name: system:coredns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:coredns
subjects:
- kind: ServiceAccount
  name: coredns
  namespace: kube-system
```

- configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
data:
  Corefile: |
    .:53 {
        errors
        log
        health
        ready
        kubernetes cluster.local 192.168.0.0/16
        forward . 10.4.7.11
        cache 30
        loop
        reload
        loadbalance
    }
```

- deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coredns
  namespace: kube-system
  labels:
    k8s-app: coredns
    kubernetes.io/name: "CoreDNS"
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: coredns
  template:
    metadata:
      labels:
        k8s-app: coredns
    spec:
      priorityClassName: system-cluster-critical
      serviceAccountName: coredns
      containers:
      - name: coredns
        image: harbor.od.com/public/coredns:v1.6.1
        args:
        - -conf
        - /etc/coredns/Corefile
```

```yaml
        volumeMounts:
        - name: config-volume
          mountPath: /etc/coredns
        ports:
        - containerPort: 53
          name: dns
          protocol: UDP
        - containerPort: 53
          name: dns-tcp
          protocol: TCP
        - containerPort: 9153
          name: metrics
          protocol: TCP
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 60
          timeoutSeconds: 5
          successThreshold: 1
          failureThreshold: 5
      dnsPolicy: Default
      volumes:
        - name: config-volume
          configMap:
            name: coredns
            items:
            - key: Corefile
              path: Corefile
```

- service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: coredns
  namespace: kube-system
  labels:
    k8s-app: coredns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: "CoreDNS"
spec:
  selector:
    k8s-app: coredns
  clusterIP: 192.168.0.2
  ports:
  - name: dns
    port: 53
    protocol: UDP
  - name: dns-tcp
    port: 53
  - name: metrics
    port: 9153
    protocol: TCP
```

5.2.3. 交付coredns到K8s

```
# 准备镜像
[root@hdss7-200 ~]# docker pull coredns/coredns:1.6.1
[root@hdss7-200 ~]# docker image tag coredns/coredns:1.6.1 harbor.od.com/public/coredns:v1.6.1
[root@hdss7-200 ~]# docker image push harbor.od.com/public/coredns:v1.6.1
```

```
# 交付coredns
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/coredns/coredns_1.6.1/rbac.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/coredns/coredns_1.6.1/configmap.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/coredns/coredns_1.6.1/deployment.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/coredns/c
```

```
oredns_1.6.1/service.yaml
[root@hdss7-21 ~]# kubectl get all -n kube-system -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP           NODE                NOMINATED NODE   READINESS GATES
pod/coredns-6b6c4f9648-4vtcl   1/1     Running   0          38s   172.7.21.3   hdss7-21.host.com   <none>           <none>

NAME              TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                  AGE   SELECTOR
service/coredns   ClusterIP   192.168.0.2   <none>        53/UDP,53/TCP,9153/TCP   29s   k8s-app=coredns

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                                SELECTOR
deployment.apps/coredns   1/1     1            1           39s   coredns      harbor.od.com/public/coredns:v1.6.1   k8s-app=coredns

NAME                                 DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                                SELECTOR
replicaset.apps/coredns-6b6c4f9648   1         1         1       39s   coredns      harbor.od.com/public/coredns:v1.6.1   k8s-app=coredns,pod-template-hash=6b6c4f9648
```

5.2.4. 测试dns

```
# 创建service
[root@hdss7-21 ~]# kubectl create deployment nginx-web --image=harbor.od.com/public/nginx:src_1.14.2
[root@hdss7-21 ~]# kubectl expose deployment nginx-web --port=80 --target-port=80
[root@hdss7-21 ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP        EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   192.168.0.1       <none>        443/TCP   8d
nginx-web    ClusterIP   192.168.164.230   <none>        80/TCP    8s
```

```
# 测试DNS，集群外必须使用FQDN(Fully Qualified Domain Name)，全域名
[root@hdss7-21 ~]# dig -t A nginx-web.default.svc.cluster.local @192.168.0.2 +short # 内网解析OK
192.168.164.230
[root@hdss7-21 ~]# dig -t A www.baidu.com @192.168.0.2 +short # 外网解析OK
www.a.shifen.com.
180.101.49.11
180.101.49.12
```

5.3. Ingress-Controller

service是将一组pod管理起来，提供了一个cluster ip和service name的统一访问入口，屏蔽了pod的ip变化。 ingress 是一种基于七层的流量转发策略，即将符合条件的域名或者location流量转发到特定的service上，而ingress仅仅是一种规则，k8s内部并没有自带代理程序完成这种规则转发。 ingress-controller 是一个代理服务器，将ingress的规则能真正实现的方式，常用的有nginx,traefik,haproxy。但是在k8s集群中，建议使用traefik，性能比haroxy强大，更新配置不需要重载服务，是首选的ingress-controller。github地址：https://github.com/containous/traefik

5.3.1. 配置traefik资源清单

清单文件存放到 hdss7-200:/data/k8s-yaml/traefik/traefik_1.7.2

- rbac.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: traefik-ingress-controller
rules:
```

```yaml
  - apiGroups:
      - ""
    resources:
      - services
      - endpoints
      - secrets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - extensions
    resources:
      - ingresses
    verbs:
      - get
      - list
      - watch
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: traefik-ingress-controller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: traefik-ingress-controller
subjects:
- kind: ServiceAccount
  name: traefik-ingress-controller
  namespace: kube-system
```

- daemonset.yaml

```yaml
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: traefik-ingress
  namespace: kube-system
  labels:
    k8s-app: traefik-ingress
spec:
  template:
    metadata:
      labels:
        k8s-app: traefik-ingress
        name: traefik-ingress
    spec:
      serviceAccountName: traefik-ingress-controller
      terminationGracePeriodSeconds: 60
      containers:
      - image: harbor.od.com/public/traefik:v1.7.2
        name: traefik-ingress
        ports:
        - name: controller
          containerPort: 80
          hostPort: 81
        - name: admin-web
          containerPort: 8080
        securityContext:
          capabilities:
            drop:
            - ALL
            add:
            - NET_BIND_SERVICE
        args:
        - --api
        - --kubernetes
        - --logLevel=INFO
        - --insecureskipverify=true
        - --kubernetes.endpoint=https://10.4.7.10:7443
        - --accesslog
        - --accesslog.filepath=/var/log/traefik_access.log
        - --traefiklog
        - --traefiklog.filepath=/var/log/traefik.log
        - --metrics.prometheus
```

- service.yaml

```yaml
kind: Service
apiVersion: v1
metadata:
  name: traefik-ingress-service
  namespace: kube-system
spec:
  selector:
    k8s-app: traefik-ingress
  ports:
    - protocol: TCP
      port: 80
      name: controller
    - protocol: TCP
      port: 8080
      name: admin-web
```

- ingress.yaml

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: traefik-web-ui
  namespace: kube-system
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: traefik.od.com
    http:
      paths:
        - path: /
        backend:
          serviceName: traefik-ingress-service
          servicePort: 8080
```

- 准备镜像

```
[root@hdss7-200 traefik_1.7.2]# docker pull traefik:v1.7.2-alpine
[root@hdss7-200 traefik_1.7.2]# docker image tag traefik:v1.7.2-alpine harbor.od.com/public/traefik:v1.7.2
[root@hdss7-200 traefik_1.7.2]# docker push harbor.od.com/public/traefik:v1.7.2
```

5.3.2. 交付traefik到k8s

```
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/traefik/traefik_1.7.2/rbac.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/traefik/traefik_1.7.2/daemonset.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/traefik/traefik_1.7.2/service.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/traefik/traefik_1.7.2/ingress.yaml
```

```
[root@hdss7-21 ~]# kubectl get pods -n kube-system -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP           NODE                NOMINATED NODE   READINESS GATES
coredns-6b6c4f9648-4vtcl   1/1     Running   1          24h   172.7.21.3   hdss7-21.host.com   <none>           <none>
traefik-ingress-4gm4w      1/1     Running   0          77s   172.7.21.5   hdss7-21.host.com   <none>           <none>
traefik-ingress-hwr2j      1/1     Running   0          77s   172.7.22.3   hdss7-22.host.com   <none>           <none>
[root@hdss7-21 ~]# kubectl get ds -n kube-system
NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
traefik-ingress   2         2         2       2            2           <none>          107s
```

5.3.3. 配置外部nginx负载均衡

- 在hdss7-11,hdss7-12 配置nginx L7转发

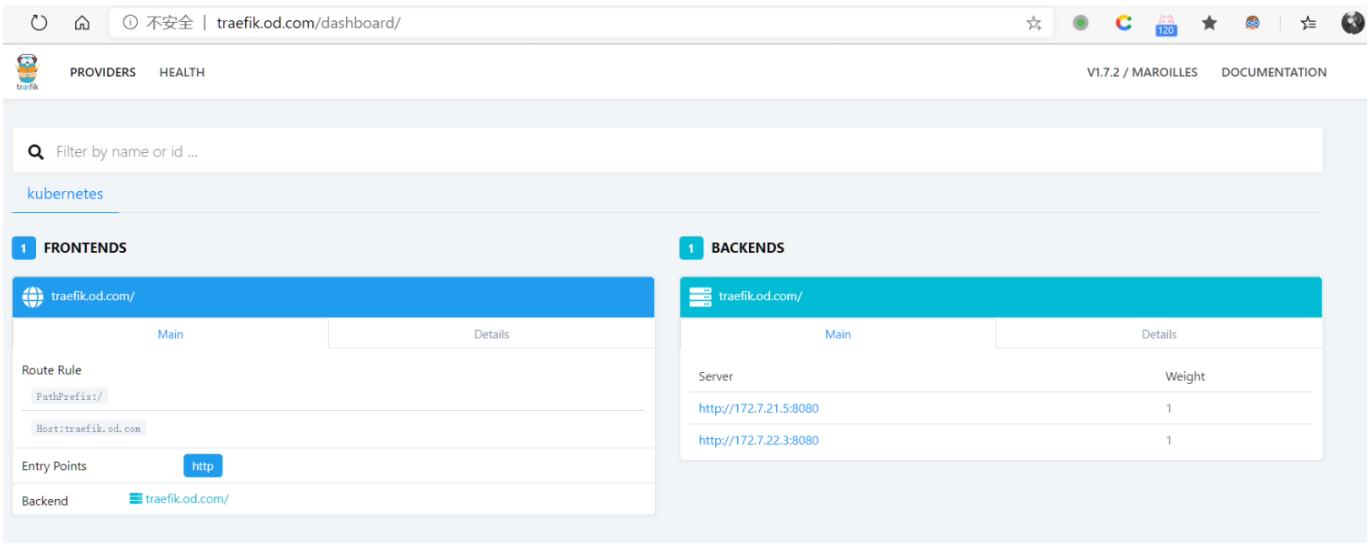
```
[root@hdss7-11 ~]# vim /etc/nginx/conf.d/od.com.conf
server {
    server_name *.od.com;

    location / {
        proxy_pass http://default_backend_traefik;
        proxy_set_header Host       $http_host;
        proxy_set_header x-forwarded-for $proxy_add_x_forwarded_for;
    }
}

upstream default_backend_traefik {
    # 所有的nodes都放到upstream中
    server 10.4.7.21:81    max_fails=3 fail_timeout=10s;
    server 10.4.7.22:81    max_fails=3 fail_timeout=10s;
}
[root@hdss7-11 ~]# nginx -tq && nginx -s reload
```

- 配置dns解析

```
[root@hdss7-11 ~]# vim /var/named/od.com.zone
$ORIGIN od.com.
$TTL 600    ; 10 minutes
@           IN SOA  dns.od.com. dnsadmin.od.com. (
                2020011302 ; serial
                10800      ; refresh (3 hours)
                900        ; retry (15 minutes)
                604800     ; expire (1 week)
                86400      ; minimum (1 day)
                )
                NS   dns.od.com.
$TTL 60 ; 1 minute
```

```
dns                A     10.4.7.11
harbor             A     10.4.7.200
k8s-yaml           A     10.4.7.200
traefik            A     10.4.7.10
[root@hdss7-11 ~]# systemctl restart named
```

- 查看traefik网页



Route Rule: PathPrefix:/ Host:traefik.od.com

Entry Points: http

Backend: traefik.od.com/

Server	Weight
http://172.7.21.5:8080	1
http://172.7.22.3:8080	1

5.4. dashboard

5.4.1. 配置资源清单

清单文件存放到 hdss7-200:/data/k8s-yaml/dashboard/dashboard_1.10.1

- 准备镜像

```
# 镜像准备
# 因不可描述原因，无法访问k8s.gcr.io，改成registry.aliyuncs.com/google_containers
[root@hdss7-200 ~]# docker image pull registry.aliyuncs.com/google_containers/kubernetes-dashboard-amd64:v1.10.1
[root@hdss7-200 ~]# docker image tag f9aed6605b81 harbor.od.com/public/kubernetes-dashboard-amd64:v1.10.1
[root@hdss7-200 ~]# docker image push harbor.od.com/public/kubernetes
```

```
-dashboard-amd64:v1.10.1
```

- rbac.yaml

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
    addonmanager.kubernetes.io/mode: Reconcile
  name: kubernetes-dashboard-admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard-admin
  namespace: kube-system
  labels:
    k8s-app: kubernetes-dashboard
    addonmanager.kubernetes.io/mode: Reconcile
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard-admin
  namespace: kube-system
```

- deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubernetes-dashboard
  namespace: kube-system
  labels:
    k8s-app: kubernetes-dashboard
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  template:
    metadata:
      labels:
        k8s-app: kubernetes-dashboard
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      priorityClassName: system-cluster-critical
      containers:
      - name: kubernetes-dashboard
        image: harbor.od.com/public/kubernetes-dashboard-amd64:v1.10.1
        resources:
          limits:
            cpu: 100m
            memory: 300Mi
          requests:
            cpu: 50m
            memory: 100Mi
        ports:
        - containerPort: 8443
          protocol: TCP
        args:
          # PLATFORM-SPECIFIC ARGS HERE
          - --auto-generate-certificates
        volumeMounts:
        - name: tmp-volume
          mountPath: /tmp
        livenessProbe:
          httpGet:
            scheme: HTTPS
            path: /
```

```
              port: 8443
            initialDelaySeconds: 30
            timeoutSeconds: 30
        volumes:
        - name: tmp-volume
          emptyDir: {}
        serviceAccountName: kubernetes-dashboard-admin
        tolerations:
        - key: "CriticalAddonsOnly"
          operator: "Exists"
```

- service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: kubernetes-dashboard
  namespace: kube-system
  labels:
    k8s-app: kubernetes-dashboard
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  selector:
    k8s-app: kubernetes-dashboard
  ports:
  - port: 443
    targetPort: 8443
```

- ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: kubernetes-dashboard
  namespace: kube-system
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: dashboard.od.com
    http:
      paths:
      - backend:
          serviceName: kubernetes-dashboard
          servicePort: 443
```

5.4.2. 交付dashboard到k8s

```
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/dashboard/dashboard_1.10.1/rbac.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/dashboard/dashboard_1.10.1/deployment.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/dashboard/dashboard_1.10.1/service.yaml
[root@hdss7-21 ~]# kubectl apply -f http://k8s-yaml.od.com/dashboard/dashboard_1.10.1/ingress.yaml
```

5.4.3. 配置DNS解析

```
[root@hdss7-11 ~]# vim /var/named/od.com.zone
$ORIGIN od.com.
$TTL 600    ; 10 minutes
@           IN SOA  dns.od.com. dnsadmin.od.com. (
                2020011303 ; serial
                10800      ; refresh (3 hours)
                900        ; retry (15 minutes)
                604800     ; expire (1 week)
                86400      ; minimum (1 day)
                )
                NS   dns.od.com.
```

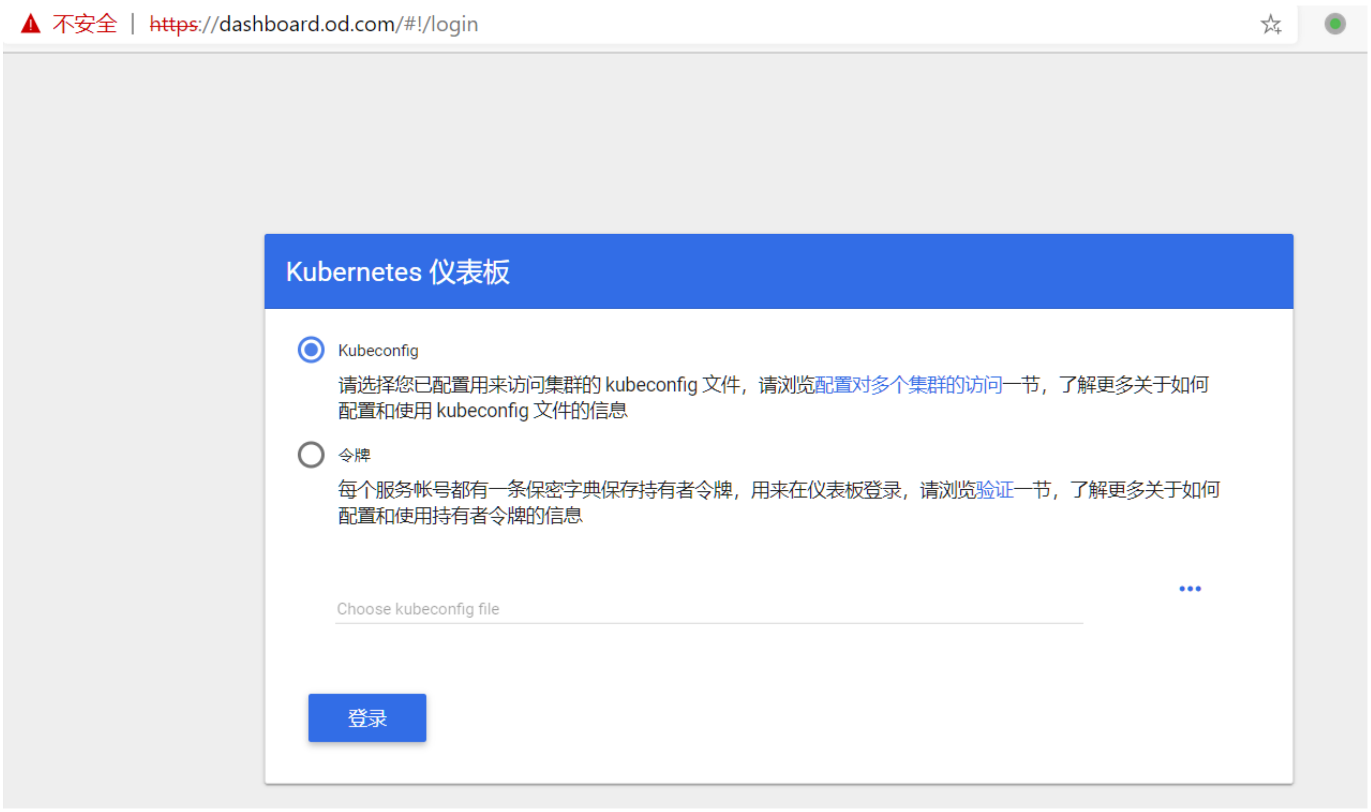
```
$TTL 60 ; 1 minute
dns                 A    10.4.7.11
harbor              A    10.4.7.200
k8s-yaml            A    10.4.7.200
traefik             A    10.4.7.10
dashboard           A    10.4.7.10
[root@hdss7-11 ~]# systemctl restart named.service
```

5.4.4. 签发SSL证书

```
[root@hdss7-200 ~]# cd /opt/certs/
[root@hdss7-200 certs]# (umask 077; openssl genrsa -out dashboard.od.com.key 2048)
[root@hdss7-200 certs]# openssl req -new -key dashboard.od.com.key -out dashboard.od.com.csr -subj "/CN=dashboard.od.com/C=CN/ST=jiangsu/L=wuxi/O=JNU/OU=AI"
[root@hdss7-200 certs]# openssl x509 -req -in dashboard.od.com.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -out dashboard.od.com.crt -days 3650
[root@hdss7-200 certs]# ll dashboard.od.com.*
-rw-r--r-- 1 root root 1196 Jan 29 20:52 dashboard.od.com.crt
-rw-r--r-- 1 root root 1005 Jan 29 20:51 dashboard.od.com.csr
-rw------- 1 root root 1675 Jan 29 20:51 dashboard.od.com.key
[root@hdss7-200 certs]# scp dashboard.od.com.key dashboard.od.com.crt hdss7-11:/etc/nginx/certs/
[root@hdss7-200 certs]# scp dashboard.od.com.key dashboard.od.com.crt hdss7-12:/etc/nginx/certs/
```

5.4.5. 配置Nginx

```
# hdss7-11和hdss7-12都需要操作
[root@hdss7-11 ~]# vim /etc/nginx/conf.d/dashborad.conf
server {
    listen       80;
    server_name  dashboard.od.com;
    rewrite ^(.*)$ https://${server_name}$1 permanent;
}

server {
    listen       443 ssl;
    server_name  dashboard.od.com;

    ssl_certificate "certs/dashboard.od.com.crt";
    ssl_certificate_key "certs/dashboard.od.com.key";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    location / {
        proxy_pass http://default_backend_traefik;
        proxy_set_header Host       $http_host;
        proxy_set_header x-forwarded-for $proxy_add_x_forwarded_for;
    }
}
[root@hdss7-11 ~]# nginx -t && nginx -s reload
```


不安全 | https://dashboard.od.com/#!/login

Kubernetes 仪表板

● Kubeconfig

请选择您已配置用来访问集群的 kubeconfig 文件，请浏览配置对多个集群的访问一节，了解更多关于如何配置和使用 kubeconfig 文件的信息

○ 令牌

每个服务帐号都有一条保密字典保存持有者令牌，用来在仪表板登录，请浏览验证一节，了解更多关于如何配置和使用持有者令牌的信息

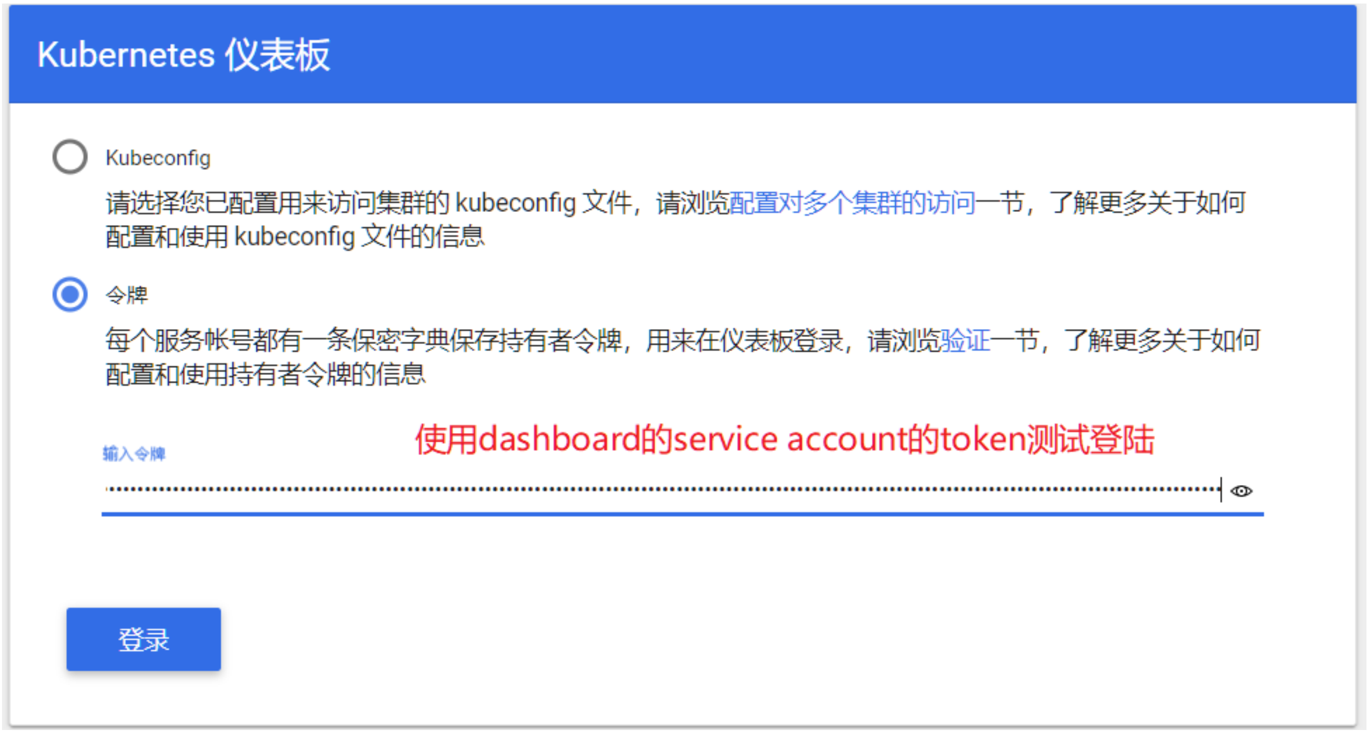
Choose kubeconfig file

登录

5.4.6. 测试token登陆

```
[root@hdss7-21 ~]# kubectl get secret -n kube-system|grep kubernetes-dashboard-token
kubernetes-dashboard-token-hr5rj         kubernetes.io/service-account-token   3      17m
[root@hdss7-21 ~]# kubectl describe secret kubernetes-dashboard-token-hr5rj -n kube-system|grep ^token
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZC10b2tlbi1ocjVyaiIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImZhNzAxZTRmLWVjMGItNDFkNS04NjdmLWY0MGEwYmFkMjFmNSIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3RlbTprdWJlcm5ldGVzLWRhc2hib2FyZCJ9.SDUZEkH_N0B6rjm6bW_jN03F4pHCPafL3uKD2HU0ksM0oenB2425jxvfi16rUbTRCsfcGqYXRrE2x15gpb03fb3jJy-IhnInUnPrw6ZwEdqWagen_Z4tdFhUgCpdjdShHy40ZPfql_iuVKbvv7ASt8w8v13Ar3FxztyDyLScVO3rNEezT7JUqMI4yj5LYQ0IgpSXoH12tlDSTyX8Rk2a_3QlOM_yT5GB_GEZkwIESttQKVr7HXSCrQ2tEdYA4c
```

```
YO2AbF1NgAo_CVBNNvZLvdDukWiQ_b5zwOiO0cUbbiu46x_p6gjNWzVb7zHNro4gh0Shr
4hIhiRQot2DJ-sq94Ag
```



Kubernetes 仪表板

Kubeconfig

请选择您已配置用来访问集群的 kubeconfig 文件，请浏览配置对多个集群的访问一节，了解更多关于如何配置和使用 kubeconfig 文件的信息

[✓ 令牌]

每个服务帐号都有一条保密字典保存持有者令牌，用来在仪表板登录，请浏览验证一节，了解更多关于如何配置和使用持有者令牌的信息

输入令牌

使用dashboard的service account的token测试登陆