

阿良微信

## 服务器规划

|  |  |  |
| --- | --- | --- |
| 角色 | IP | 组件 |
| k8s-master1 | 192.168.31.63 | kube-apiserver  kube-controller-manager  kube-scheduler  etcd |
| k8s-master2 | 192.168.31.64 | kube-apiserver  kube-controller-manager  kube-scheduler |
| k8s-node1 | 192.168.31.65 | kubelet  kube-proxy  docker  etcd |
| k8s-node2 | 192.168.31.66 | kubelet  kube-proxy  docker  etcd |
| Load Balancer（Master） | 192.168.31.61  192.168.31.60 (VIP) | Nginx L4 |
| Load Balancer（Backup） | 192.168.31.62 | Nginx L4 |

1. **系统初始化**

关闭防火墙：

# systemctl stop firewalld

# systemctl disable firewalld

关闭selinux：

# setenforce 0 # 临时

# sed -i 's/enforcing/disabled/' /etc/selinux/config # 永久

关闭swap：

# swapoff -a # 临时

# vim /etc/fstab # 永久

同步系统时间：

# ntpdate time.windows.com

添加hosts：

# vim /etc/hosts

192.168.31.63 k8s-master1

192.168.31.64 k8s-master2

192.168.31.65 k8s-node1

192.168.31.66 k8s-node2

修改主机名：

hostnamectl set-hostname k8s-master1

1. **Etcd集群**

可在任意节点完成以下操作。

### 2.1 生成etcd证书

# cd TLS/etcd

安装cfssl工具：

# ./cfssl.sh

修改请求文件中hosts字段包含所有etcd节点IP：

# vi server-csr.json

{

"CN": "etcd",

"hosts": [

"192.168.31.63",

"192.168.31.64",

"192.168.31.65"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "BeiJing",

"ST": "BeiJing"

}

]

}

# ./generate\_etcd\_cert.sh

# ls \*pem

ca-key.pem ca.pem server-key.pem server.pem

### 2.2 部署三个Etcd节点

# tar zxvf etcd.tar.gz

# cd etcd

# cp TLS/etcd/ssl/{ca,server,server-key}.pem ssl

分别拷贝到Etcd三个节点：

# scp –r etcd root@192.168.31.63:/opt

# scp etcd.service root@192.168.31.63:/usr/lib/systemd/system

登录三个节点修改配置文件 名称和IP：

# vi /opt/etcd/cfg/etcd.conf

#[Member]

ETCD\_NAME="etcd-1"

ETCD\_DATA\_DIR="/var/lib/etcd/default.etcd"

ETCD\_LISTEN\_PEER\_URLS="https://192.168.31.63:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https://192.168.31.63:2379"

#[Clustering]

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://192.168.31.63:2380"

ETCD\_ADVERTISE\_CLIENT\_URLS="https://192.168.31.63:2379"

ETCD\_INITIAL\_CLUSTER="etcd-1=https://192.168.31.63:2380,etcd-2=https://192.168.31.64:2380,etcd-3=https://192.168.31.65:2380"

ETCD\_INITIAL\_CLUSTER\_TOKEN="etcd-cluster"

ETCD\_INITIAL\_CLUSTER\_STATE="new"

# systemctl start etcd

# systemctl enable etcd

### 2.3 查看集群状态

# /opt/etcd/bin/etcdctl \

> --ca-file=/opt/etcd/ssl/ca.pem --cert-file=/opt/etcd/ssl/server.pem --key-file=/opt/etcd/ssl/server-key.pem \

> --endpoints="https://192.168.31.63:2379,https://192.168.31.64:2379,https://192.168.31.65:2379" \

> cluster-health

member 37f20611ff3d9209 is healthy: got healthy result from https://192.168.31.63:2379

member b10f0bac3883a232 is healthy: got healthy result from https://192.168.31.64:2379

member b46624837acedac9 is healthy: got healthy result from https://192.168.31.65:2379

cluster is healthy

1. **部署Master Node**

### 生成apiserver证书

# cd TLS/k8s

修改请求文件中hosts字段包含所有etcd节点IP：

# vi server-csr.json

{

"CN": "kubernetes",

"hosts": [

"10.0.0.1",

"127.0.0.1",

"kubernetes",

"kubernetes.default",

"kubernetes.default.svc",

"kubernetes.default.svc.cluster",

"kubernetes.default.svc.cluster.local",

"192.168.31.60",

"192.168.31.61",

"192.168.31.62",

"192.168.31.63",

"192.168.31.64",

"192.168.31.65",

"192.168.31.66"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "BeiJing",

"ST": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

# ./generate\_k8s\_cert.sh

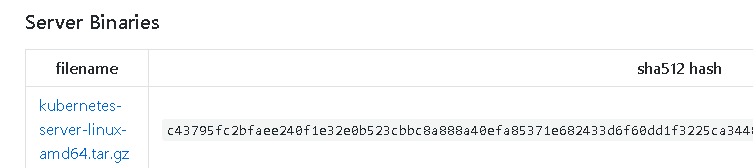
# ls \*pem

ca-key.pem ca.pem kube-proxy-key.pem kube-proxy.pem server-key.pem server.pem

### 3.2 部署apiserver，controller-manager和scheduler

在Master节点完成以下操作。

二进制包下载地址：<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.16.md#v1161>



二进制文件位置：kubernetes/serverr/bin

# tar zxvf k8s-master.tar.gz

# cd kubernetes

# cp TLS/k8s/ssl/\*.pem ssl

# cp –rf kubernetes /opt

# cp kube-apiserver.service kube-controller-manager.service kube-scheduler.service /usr/lib/systemd/system

# cat /opt/kubernetes/cfg/kube-apiserver.conf

KUBE\_APISERVER\_OPTS="--logtostderr=false \

--v=2 \

--log-dir=/opt/kubernetes/logs \

--etcd-servers=https://192.168.31.63:2379,https://192.168.31.64:2379,https://192.168.31.65:2379 \

--bind-address=192.168.31.63 \

--secure-port=6443 \

--advertise-address=192.168.31.63 \

……

# systemctl start kube-apiserver

# systemctl start kube-controller-manager

# systemctl start kube-scheduler

# systemctl enable kube-apiserver

# systemctl enable kube-controller-manager

# systemctl enable kube-scheduler

### 3.3 启用TLS Bootstrapping

为kubelet TLS Bootstrapping 授权：

# cat /opt/kubernetes/cfg/token.csv

c47ffb939f5ca36231d9e3121a252940,kubelet-bootstrap,10001,"system:node-bootstrapper"

格式：token,用户,uid,用户组

给kubelet-bootstrap授权：

kubectl create clusterrolebinding kubelet-bootstrap \

--clusterrole=system:node-bootstrapper \

--user=kubelet-bootstrap

token也可自行生成替换：

head -c 16 /dev/urandom | od -An -t x | tr -d ' '

但apiserver配置的token必须要与node节点bootstrap.kubeconfig配置里一致。

1. **部署Worker Node**

### 安装Docker

二进制包下载地址：<https://download.docker.com/linux/static/stable/x86_64/>

# tar zxvf k8s-node.tar.gz

# tar zxvf docker-18.09.6.tgz

# mv docker/\* /usr/bin

# mkdir /etc/docker

# mv daemon.json /etc/docker

# mv docker.service /usr/lib/systemd/system

# systemctl start docker

# systemctl enable docker

### 4.2 部署kubelet和kube-proxy

拷贝证书到Node：

# cd TLS/k8s

# scp ca.pem kube-proxy\*.pem root@192.168.31.65:/opt/kubernetes/ssl/

# cp kube-apiserver.service kube-controller-manager.service kube-

# tar zxvf k8s-node.tar.gz

# mv kubernetes /opt

# cp kubelet.service kube-proxy.service /usr/lib/systemd/system

修改以下三个文件中IP地址：

# grep 192 \*

bootstrap.kubeconfig: server: https://192.168.31.63:6443

kubelet.kubeconfig: server: https://192.168.31.63:6443

kube-proxy.kubeconfig: server: https://192.168.31.63:6443

修改以下两个文件中主机名：

# grep hostname \*

kubelet.conf:--hostname-override=k8s-node1 \

kube-proxy-config.yml:hostnameOverride: k8s-node1

# systemctl start kubelet

# systemctl start kube-proxy

# systemctl enable kubelet

# systemctl enable kube-proxy

### 4.3 允许给Node颁发证书

# kubectl get csr

# kubectl certificate approve node-csr-MYUxbmf\_nmPQjmH3LkbZRL2uTO-\_FCzDQUoUfTy7YjI

# kubectl get node

### 4.4 部署CNI网络

二进制包下载地址：<https://github.com/containernetworking/plugins/releases>

# mkdir /opt/cni/bin /etc/cni/net.d

# tar zxvf cni-plugins-linux-amd64-v0.8.2.tgz –C /opt/cni/bin

确保kubelet启用CNI：

# cat /opt/kubernetes/cfg/kubelet.conf

--network-plugin=cni

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

在Master执行：

kubectl apply –f kube-flannel.yaml

# kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

kube-flannel-ds-amd64-5xmhh 1/1 Running 6 171m

kube-flannel-ds-amd64-ps5fx 1/1 Running 0 150m

### 4.5 授权apiserver访问kubelet

为提供安全性，kubelet禁止匿名访问，必须授权才可以。

# cat /opt/kubernetes/cfg/kubelet-config.yml

……

authentication:

anonymous:

enabled: false

webhook:

cacheTTL: 2m0s

enabled: true

x509:

clientCAFile: /opt/kubernetes/ssl/ca.pem

……

# kubectl apply –f apiserver-to-kubelet-rbac.yaml

## 部署Web UI和DNS

<https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/>

# wget https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta4/aio/deploy/recommended.yaml

# vi recommended.yaml

…

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kubernetes-dashboard

spec:

type: NodePort

ports:

- port: 443

targetPort: 8443

nodePort: 30001

selector:

k8s-app: kubernetes-dashboard

…

# kubectl apply -f recommended.yaml

创建service account并绑定默认cluster-admin管理员集群角色：

# cat dashboard-adminuser.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kubernetes-dashboard

获取token：

# kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')

访问地址：<http://NodeIP:30001>

使用输出的token登录Dashboard。

# kubectl apply –f coredns.yaml

# kubectl get pods -n kube-system

## Master高可用

### 5.1 部署Master组件（与Master1一致）

拷贝master1/opt/kubernetes和service文件：

# scp –r /opt/kubernetes root@192.168.31.64:/opt

# scp –r /opt/etcd/ssl root@192.168.31.64:/opt/etcd

# scp /usr/lib/systemd/system/{kube-apiserver,kube-controller-manager,kube-scheduler}.service root@192.168.31.64:/usr/lib/systemd/system

修改apiserver配置文件为本地IP：

# cat /opt/kubernetes/cfg/kube-apiserver.conf

KUBE\_APISERVER\_OPTS="--logtostderr=false \

--v=2 \

--log-dir=/opt/kubernetes/logs \

--etcd-servers=https://192.168.31.63:2379,https://192.168.31.64:2379,https://192.168.31.65:2379 \

--bind-address=192.168.31.64 \

--secure-port=6443 \

--advertise-address=192.168.31.64 \

……

# systemctl start kube-apiserver

# systemctl start kube-controller-manager

# systemctl start kube-scheduler

# systemctl enable kube-apiserver

# systemctl enable kube-controller-manager

# systemctl enable kube-scheduler

### 5.2 部署Nginx负载均衡

nginx rpm包：<http://nginx.org/packages/rhel/7/x86_64/RPMS/>

# rpm -vih <http://nginx.org/packages/rhel/7/x86_64/RPMS/nginx-1.16.0-1.el7.ngx.x86_64.rpm>

# vim /etc/nginx/nginx.conf

……

stream {

log\_format main '$remote\_addr $upstream\_addr - [$time\_local] $status $upstream\_bytes\_sent';

access\_log /var/log/nginx/k8s-access.log main;

upstream k8s-apiserver {

server 192.168.31.63:6443;

server 192.168.31.64:6443;

}

server {

listen 6443;

proxy\_pass k8s-apiserver;

}

}

……

# systemctl start nginx

# systemctl enable nginx

### 5.3 Nginx+Keepalived高可用

**主节点：**

# yum install keepalived

# vi /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id NGINX\_MASTER

}

vrrp\_script check\_nginx {

script "/etc/keepalived/check\_nginx.sh"

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 51 # VRRP 路由 ID实例，每个实例是唯一的

priority 100 # 优先级，备服务器设置 90

advert\_int 1 # 指定VRRP 心跳包通告间隔时间，默认1秒

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.31.60/24

}

track\_script {

check\_nginx

}

}

# cat /etc/keepalived/check\_nginx.sh

#!/bin/bash

count=$(ps -ef |grep nginx |egrep -cv "grep|$$")

if [ "$count" -eq 0 ];then

exit 1

else

exit 0

fi

# systemctl start keepalived

# systemctl enable keepalived

**备节点：**

# cat /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id NGINX\_BACKUP

}

vrrp\_script check\_nginx {

script "/etc/keepalived/check\_nginx.sh"

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens33

virtual\_router\_id 51 # VRRP 路由 ID实例，每个实例是唯一的

priority 90 # 优先级，备服务器设置 90

advert\_int 1 # 指定VRRP 心跳包通告间隔时间，默认1秒

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.31.60/24

}

track\_script {

check\_nginx

}

}

# cat /etc/keepalived/check\_nginx.sh

#!/bin/bash

count=$(ps -ef |grep nginx |egrep -cv "grep|$$")

if [ "$count" -eq 0 ];then

exit 1

else

exit 0

fi

# systemctl start keepalived

# systemctl enable keepalived

**测试：**

# ip a

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 00:0c:29:9d:ee:30 brd ff:ff:ff:ff:ff:ff

inet 192.168.31.63/24 brd 192.168.31.255 scope global noprefixroute ens33

valid\_lft forever preferred\_lft forever

inet 192.168.31.60/24 scope global secondary ens33

valid\_lft forever preferred\_lft forever

inet6 fe80::20c:29ff:fe9d:ee30/64 scope link

valid\_lft forever preferred\_lft forever

关闭nginx测试VIP是否漂移到备节点。

### 5.4 修改Node连接VIP

测试VIP是否正常工作：

# curl -k --header "Authorization: Bearer c47ffb939f5ca36231d9e3121a252940" https://192.168.31.60:6443/version

{

"major": "1",

"minor": "16",

"gitVersion": "v1.16.0",

"gitCommit": "2bd9643cee5b3b3a5ecbd3af49d09018f0773c77",

"gitTreeState": "clean",

"buildDate": "2019-09-18T14:27:17Z",

"goVersion": "go1.12.9",

"compiler": "gc",

"platform": "linux/amd64"

}

将Node连接VIP：

# cd /opt/kubernetes/cfg

# grep 192 \*

bootstrap.kubeconfig: server: https://192.168.31.63:6443

kubelet.kubeconfig: server: https://192.168.31.636443

kube-proxy.kubeconfig: server: https://192.168.31.63:6443

批量修改：

sed -i 's#192.168.31.63#192.168.31.60#g' \*