k8s-master1 10.206.240.188

k8s-master2 10.206.240.189

k8s-node1 10.206.240.111

k8s-node2 10.206.240.112

1. 安装nginxֵ stream模块 4层

{

user nginx;

worker\_processes 4;

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

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 10.206.240.188:6443;

server 10.206.240.189:6443;

}

server {

listen 6443;

proxy\_pass k8s-apiserver;

}

}

3.etcd集群

下载cfssl工具，证书生成

ֵ# wget <https://pkg.cfssl.org/R1.2/cfssl_linux-amd64>

# wget <https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64>

# wget <https://pkg.cfssl.org/R1.2/cfssl-certinfo_linux-amd64>

# chmod +x cfssl\_linux-amd64 cfssljson\_linux-amd64 cfssl-certinfo\_linuxamd64

# mv cfssl\_linux-amd64 /usr/local/bin/cfssl

# mv cfssljson\_linux-amd64 /usr/local/bin/cfssljson

# mv cfssl-certinfo\_linux-amd64 /usr/bin/cfssl-certinfo

生成etcd证书，创建以下三个文件

# cat ca-config.json

{

"signing": {

"default": {

"expiry": "87600h"

},

"profiles": { "www": {

"expiry": "87600h",

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth" ]

}

}

}

}

# cat ca-csr.json

{ "CN": "etcd CA",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "Beijing",

"ST": "Beijing"

}

]

}

# cat server-csr.json

{ "CN": "etcd",

"hosts": [

"10.206.240.188",

"10.206.240.189",

"10.206.240.111"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "BeiJing",

"ST": "BeiJing" }

]

}

生成证书

# cfssl gencert -initca ca-csr.json | cfssljson -bare ca -

# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=www server-csr.json | cfssljson -bare server

# ls \*pem ca-key.pem ca.pem server-key.pem server.pem

安装etcd安装包,二进制包下载地址: <https://github.com/coreos/etcd/releases/tag/v3.2.12>

Etcd服务器的IP要写当前的。

# mkdir /opt/etcd/{bin,cfg,ssl} -p

# tar zxvf etcd-v3.2.12-linux-amd64.tar.gz

# mv etcd-v3.2.12-linux-amd64/{etcd,etcdctl} /opt/etcd/bin/

# cat /opt/etcd/cfg/etcd

#[Member]

ETCD\_NAME="etcd01"

ETCD\_DATA\_DIR="/var/lib/etcd/default.etcd" ETCD\_LISTEN\_PEER\_URLS="https://10.206.240.189:2380" ETCD\_LISTEN\_CLIENT\_URLS="https://10.206.240.189:2379"

#[Clustering]

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://10.206.240.189:2380" ETCD\_ADVERTISE\_CLIENT\_URLS="https://10.206.240.189:2379" ETCD\_INITIAL\_CLUSTER="etcd01=https://10.206.240.189:2380,etcd02=https:// 10.206.240.188:2380,etcd03=https://10.206.240.111:2380"

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

ETCD\_INITIAL\_CLUSTER\_STATE="new“

\* ETCD\_NAME 节点名称

\* ETCD\_DATA\_DIR 数据目录

\* ETCD\_LISTEN\_PEER\_URLS 集群通信监听地址

\* ETCD\_LISTEN\_CLIENT\_URLS 客户端访问监听地址

\* ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS 集群通告地址

\* ETCD\_ADVERTISE\_CLIENT\_URLS 客户端通告地址

\* ETCD\_INITIAL\_CLUSTER 集群节点地址

\* ETCD\_INITIAL\_CLUSTER\_TOKEN 集群Token

\* ETCD\_INITIAL\_CLUSTER\_STATE 加入集群当前状态，new为新加入

Systemd管理Etcd

# cat /usr/lib/systemd/system/etcd.service

[Unit] Description=Etcd Server

After=network.target

After=network-online.target

Wants=network-online.target

[Service]

Type=notify

EnvironmentFile=/opt/etcd/cfg/etcd ExecStart=/opt/etcd/bin/etcd \

--name=${ETCD\_NAME} \

--data-dir=${ETCD\_DATA\_DIR} \

--listen-peer-urls=${ETCD\_LISTEN\_PEER\_URLS} \

--listen-client-urls=${ETCD\_LISTEN\_CLIENT\_URLS},http://127.0.0.1:2379 \

--advertise-client-urls=${ETCD\_ADVERTISE\_CLIENT\_URLS} \

--initial-advertise-peer-urls=${ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS} \

--initial-cluster=${ETCD\_INITIAL\_CLUSTER} \

--initial-cluster-token=${ETCD\_INITIAL\_CLUSTER\_TOKEN} \

--initial-cluster-state=new \

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

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

--peer-cert-file=/opt/etcd/ssl/server.pem \

--peer-key-file=/opt/etcd/ssl/server-key.pem \

--trusted-ca-file=/opt/etcd/ssl/ca.pem \

--peer-trusted-ca-file=/opt/etcd/ssl/ca.pem

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

上面生成的证书拷贝到ssl

# cp ca\*pem server\*pem /opt/etcd/ssl

# /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://10.206.240.189:2379,https://10.206.240.188:2379, https ://10.206.240.111:2379" \

cluster-health

member 18218cfabd4e0dea is healthy: got healthy result from https://10.206.240.111:2379

member 541c1c40994c939b is healthy: got healthy result from <https://10.206.240.189:2379>

member a342ea2798d20705 is healthy: got healthy result from <https://10.206.240.188:2379>

cluster is healthy

在node节点上安装docker  
# yum install -y yum-utils device-mapper-persistent-data lvm2

# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo   
# yum install docker-ce -y   
#curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | sh -s <http://bc437cce.m.daocloud.io>  
# systemctl start docker

# systemctl enable docker

部署Falnnel网络  
Falnnel 要用etcd存储自身的一个子网信息，所以需要预设定义连接etcd  
# /opt/etcd/bin/etcdctl \ --ca-file=ca.pem --cert-file=server.pem --key-file=server-key.pem \

-endpoints="https://10.206.240.189:2379,https://10.206.240.188:2379,https ://10.206.240.111:2379" \

set /coreos.com/network/config '{ "Network": "172.17.0.0/16", "Backend": {"Type": "vxlan"}}'  
  
在每个node节点都要操作  
# wget https://github.com/coreos/flannel/releases/download/v0.10.0/flannelv0.10.0-linux-amd64.tar.gz   
# tar zxvf flannel-v0.10.0-linux-amd64.tar.gz   
# mkdir -pv /opt/kubernetes/bin   
# mv flanneld mk-docker-opts.sh /opt/kubernetes/bin  
  
配置Falnnel  
# mkdir -pv /opt/kubernetes/cfg/   
#cat /opt/kubernetes/cfg/flanneld   
FLANNEL\_OPTIONS="--etcdendpoints=https://10.206.240.189:2379,https://10.206.240.188:2379,https: //10.206.240.111:2379   
-etcd-cafile=/opt/etcd/ssl/ca.pem   
-etcdcertfile=/opt/etcd/ssl/server.pem   
-etcd-keyfile=/opt/etcd/ssl/serverkey.pem  
  
systemd管理Fannel  
# cat /usr/lib/systemd/system/flanneld.service   
[Unit]   
Description=Flanneld overlay address etcd agent   
After=network-online.target network.target   
Before=docker.service

[Service]   
Type=notify   
EnvironmentFile=/opt/kubernetes/cfg/flanneld   
ExecStart=/opt/kubernetes/bin/flanneld --ip-masq $FLANNEL\_OPTIONS   
ExecStartPost=/opt/kubernetes/bin/mk-docker-opts.sh \

-k DOCKER\_NETWORK\_OPTIONS -d /run/flannel/subnet.env Restart=on-failure

[Install]   
WantedBy=multi-user.target  
  
配置Docker启动子网段  
# cat /usr/lib/systemd/system/docker.service   
[Unit] Description=Docker Application Container Engine   
Documentation=https://docs.docker.com   
After=network-online.target firewalld.service   
Wants=network-online.target

[Service]   
Type=notify   
EnvironmentFile=/run/flannel/subnet.env   
ExecStart=/usr/bin/dockerd $DOCKER\_NETWORK\_OPTIONS   
ExecReload=/bin/kill -s HUP $MAINPID   
LimitNOFILE=infinity   
LimitNPROC=infinity   
LimitCORE=infinity   
TimeoutStartSec=0   
Delegate=yes   
KillMode=process   
Restart=on-failure   
StartLimitBurst=3   
StartLimitInterval=60s

[Install]   
WantedBy=multi-user.targe  
  
因为node节点没有证书，所以需要拷贝  
# mkdir -pv /opt/etcd/ssl/   
# scp /opt/etcd/ssl/\* k8s-node2:/opt/etcd/ssl/  
  
重启Falnnel和Docker  
# systemctl daemon-reload；systemctl enable flanneld   
# systemctl start flanneld；systemctl restart docker  
检查  
# ps -ef |grep docker  
# ip addr  
  
生成证书，创建CA证书  
# cat ca-config.json {

"signing": {

"default": {

"expiry": "87600h"

},

"profiles": {

"kubernetes": {

"expiry": "87600h",

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth" ]

}

}

}

}  
# cat ca-csr.json {

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "Beijing",

"ST": "Beijing",

"O": "k8s",

"OU": "System"

}

]

}  
  
# cfssl gencert -initca ca-csr.json | cfssljson -bare ca -  
生成api-server的证书  
# cat server-csr.json {

"CN": "kubernetes",

"hosts": [

"10.0.0.1",

"127.0.0.1",

"10.206.176.19",

"10.206.240.188",

"10.206.240.189",

"kubernetes",

"kubernetes.default",

"kubernetes.default.svc",

"kubernetes.default.svc.cluster",

"kubernetes.default.svc.cluster.local"

], "

key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "BeiJing",

"ST": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}  
# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes server-csr.json | cfssljson -bare server  
  
生成kube-proxy证书  
# cat kube-proxy-csr.json

{

"CN": "system:kube-proxy",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "BeiJing",

"ST": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}  
# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kube-proxy-csr.json | cfssljson -bare kube-proxy  
# ls \*pem ca-key.pem ca.pem kube-proxy-key.pem kube-proxy.pem server-key.pem server.pem  
  
  
  
  
部署api-server组件  
<https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.11.md>  
kubernetes-server-linux-amd64.tar.gz 包含所有组件  
  
# mkdir /opt/kubernetes/{bin,cfg,ssl} -pv  
# tar zxvf kubernetes-server-linux-amd64.tar.gz   
# cd kubernetes/server/bin   
# cp kube-apiserver kube-scheduler kube-controller-manager kubectl /opt/kubernetes/bin  
  
将前面做好的证书拷贝到此  
# scp server.pem server-key.pem ca.pem ca-key.pem k8smaster1:/opt/kubernetes/ssl/   
# scp server.pem server-key.pem ca.pem ca-key.pem k8smaster2:/opt/kubernetes/ssl/  
  
创建Token文件  
# cat /opt/kubernetes/cfg/token.csv

674c457d4dcf2eefe4920d7dbb6b0ddc,kubeletbootstrap,10001,"system:kubelet-bootstrap"  
  
创建api-server的配置文件  
# cat /opt/kubernetes/cfg/kube-apiserver

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

--v=4 \

--etcdservers=https://10.206.240.189:2379,https://10.206.240.188:2379,https:// 10.206.240.111:2379 \

--bind-address=10.206.240.189 \

--secure-port=6443 \

--advertise-address=10.206.240.189 \

--allow-privileged=true \

--service-cluster-ip-range=10.0.0.0/24 \

--enable-admission-plugins=NamespaceLifecycle,LimitRanger,ServiceAccount,ResourceQuota,Node Restriction \

--authorization-mode=RBAC,Node \

--enable-bootstrap-token-auth \

--token-auth-file=/opt/kubernetes/cfg/token.csv \

--service-node-port-range=30000-50000 \

--tls-cert-file=/opt/kubernetes/ssl/server.pem \

--tls-private-key-file=/opt/kubernetes/ssl/server-key.pem \

--client-ca-file=/opt/kubernetes/ssl/ca.pem \

--service-account-key-file=/opt/kubernetes/ssl/ca-key.pem \

--etcd-cafile=/opt/etcd/ssl/ca.pem \

--etcd-certfile=/opt/etcd/ssl/server.pem \

--etcd-keyfile=/opt/etcd/ssl/server-key.pem"  
  
\* --logtostderr 启动日志  
\* --v 日志等级  
\* --etcd-servers etcd集群地址  
\* --bind-address 监听地址  
\* --secure-port https安全端口  
\* --advertise-address 集群通告地址  
\* --allow-privileged 启用授权  
\* --service-cluster-ip-range serveice虚拟IP地址  
\* --enable-admission-plugins 准入控制模块  
\* --authorization-mode 认证授权  
\* --enable-bootstrap-token-auth 启用TLS  
\* --token-auth-file token文件  
\* --service-node-port-range node端口范围  
  
创建systemd管理api-server  
# cat /usr/lib/systemd/system/kube-apiserver.service   
[Unit]   
Description=Kubernetes API Server   
Documentation=https://github.com/kubernetes/kubernetes

[Service]   
EnvironmentFile=-/opt/kubernetes/cfg/kube-apiserver   
ExecStart=/opt/kubernetes/bin/kube-apiserver $KUBE\_APISERVER\_OPTS   
Restart=on-failure

[Install]   
WantedBy=multi-user.target  
  
启动  
# systemctl daemon-reload   
# systemctl enable kube-apiserver   
# systemctl start kube-apiserver  
  
以下配置scheduler  
# cat /opt/kubernetes/cfg/kube-scheduler   
KUBE\_SCHEDULER\_OPTS="--logtostderr=true \  
--v=4 \  
--master=127.0.0.1:8080 \  
--leader-elect"  
  
# cat /usr/lib/systemd/system/kube-scheduler.service  
[Unit]   
Description=Kubernetes Scheduler   
Documentation=https://github.com/kubernetes/kubernetes

[Service]   
EnvironmentFile=-/opt/kubernetes/cfg/kube-scheduler   
ExecStart=/opt/kubernetes/bin/kube-scheduler $KUBE\_SCHEDULER\_OPTS   
Restart=on-failure

[Install]   
WantedBy=multi-user.target  
  
启动  
# systemctl daemon-reload   
# systemctl enable kube-scheduler   
# systemctl start kube-scheduler  
部署controller-manager  
# cat /opt/kubernetes/cfg/kube-controller-manager KUBE\_CONTROLLER\_MANAGER\_OPTS="--logtostderr=true \  
--v=4 \   
--master=127.0.0.1:8080 \   
--leader-elect=true \   
--address=127.0.0.1 \   
--service-cluster-ip-range=10.0.0.0/24 \   
--cluster-name=kubernetes \   
--cluster-signing-cert-file=/opt/kubernetes/ssl/ca.pem \   
--cluster-signing-key-file=/opt/kubernetes/ssl/ca-key.pem \   
--root-ca-file=/opt/kubernetes/ssl/ca.pem \   
--service-account-private-key-file=/opt/kubernetes/ssl/ca-key.pem"  
  
# cat /usr/lib/systemd/system/kube-controller-manager.service   
[Unit]   
Description=Kubernetes Controller Manager   
Documentation=https://github.com/kubernetes/kubernetes

[Service]   
EnvironmentFile=-/opt/kubernetes/cfg/kube-controller-manager   
ExecStart=/opt/kubernetes/bin/kube-controller-manager $KUBE\_CONTROLLER\_MANAGER\_OPTS   
Restart=on-failure

[Install]   
WantedBy=multi-user.target  
  
  
启动  
# systemctl daemon-reload   
# systemctl enable kube-controller-manager   
# systemctl start kube-controller-manager  
  
# /opt/kubernetes/bin/kubectl get cs  
NAME STATUS MESSAGE ERROR   
scheduler Healthy ok   
etcd-0 Healthy {"health":"true"}   
etcd-2 Healthy {"health":"true"}   
etcd-1 Healthy {"health":"true"}   
controller-manager Healthy ok  
  
  
master节点上操作   
创建kubelet kubeconfig文件  
# /opt/kubernetes/bin/kubectl create clusterrolebinding kubeletbootstrap \

--clusterrole=system:node-bootstrapper \

--user=kubelet-bootstrap

# KUBE\_APISERVER="https://10.206.176.19:6443"   
# BOOTSTRAP\_TOKEN=674c457d4dcf2eefe4920d7dbb6b0ddc  
  
  
# /opt/kubernetes/bin/kubectl config set-cluster kubernetes \

--certificate-authority=./ca.pem \

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=bootstrap.kubeconfig  
  
# /opt/kubernetes/bin/kubectl config set-credentials kubelet-bootstrap \

--token=${BOOTSTRAP\_TOKEN} \

--kubeconfig=bootstrap.kubeconfig  
  
# /opt/kubernetes/bin/kubectl config set-context default \

--cluster=kubernetes \

--user=kubelet-bootstrap \

--kubeconfig=bootstrap.kubeconfig

# /opt/kubernetes/bin/kubectl config use-context default -kubeconfig=bootstrap.kubeconfig  
----------------  
创建kube-proxy kubeconfig文件  
# /opt/kubernetes/bin/kubectl config set-cluster kubernetes \

--certificate-authority=./ca.pem \

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=kube-proxy.kubeconfig  
  
# /opt/kubernetes/bin/kubectl config set-credentials kube-proxy \

--client-certificate=./kube-proxy.pem \

--client-key=./kube-proxy-key.pem \

--embed-certs=true \

--kubeconfig=kube-proxy.kubeconfig  
  
# /opt/kubernetes/bin/kubectl config set-context default \

--cluster=kubernetes \

--user=kube-proxy \

--kubeconfig=kube-proxy.kubeconfig  
  
# ls bootstrap.kubeconfig kube-proxy.kubeconfig  
将这两个文件复制到node节点下/opt/kubernetes/cfg  
  
node节点上操作  
# vim /opt/kubernetes/cfg/kubelet KUBELET\_OPTS="--logtostderr=true \

--v=4 \

--hostname-override=10.206.240.112 \

--kubeconfig=/opt/kubernetes/cfg/kubelet.kubeconfig \

--bootstrap-kubeconfig=/opt/kubernetes/cfg/bootstrap.kubeconfig \

--config=/opt/kubernetes/cfg/kubelet.config \

--cert-dir=/opt/kubernetes/ssl \

--pod-infra-container-image=registry.cn-hangzhou.aliyuncs.com/googlecontainers/pause-amd64:3.0"  
  
\* --hostname-override 在集群中显示主机名  
\* --kubeconfig 指定kubeconfig的位置，会自动生成  
\* --bootstrap-kubeconfig 指定刚生成的bootstrap.kubeconfig  
\* --cert-dir 颁发证书的存放位置  
\* --pod-infra-container-image 管理pod网络的镜像  
  
# vim /opt/kubernetes/cfg/kubelet.config   
kind: KubeletConfiguration   
apiVersion: kubelet.config.k8s.io/v1beta1   
address: 10.206.240.112   
port: 10250   
readOnlyPort: 10255   
cgroupDriver: cgroupfs   
clusterDNS: ["10.0.0.2"] #不能改  
clusterDomain: cluster.local.   
failSwapOn: false   
authentication:   
 anonymous:   
 enabled: true   
webhook:   
 enabled: false  
  
# vim /usr/lib/systemd/system/kubelet.service   
[Unit]  
Description=Kubernetes Kubelet   
After=docker.service   
Requires=docker.service

[Service]   
EnvironmentFile=/opt/kubernetes/cfg/kubelet  
ExecStart=/opt/kubernetes/bin/kubelet $KUBELET\_OPTS   
Restart=on-failure   
KillMode=process

[Install]   
WantedBy=multi-user.target  
  
启动  
# systemctl daemon-reload   
# systemctl enable kubelet   
# systemctl start kubelet  
  
这时候要去master审批node加入节点  
# /opt/kubernetes/bin/kubectl get csr   
# /opt/kubernetes/bin/kubectl certificate approve XXXXID   
# /opt/kubernetes/bin/kubectl get node  
  
  
  
  
部署kube-proxy  
# cat /opt/kubernetes/cfg/kube-proxy KUBE\_PROXY\_OPTS="--logtostderr=true \

--v=4 \

--hostname-override=10.206.240.111 \

--cluster-cidr=10.0.0.0/24 \ #不能改

--kubeconfig=/opt/kubernetes/cfg/kube-proxy.kubeconfig"  
# cat /usr/lib/systemd/system/kube-proxy.service   
[Unit]   
Description=Kubernetes Proxy   
After=network.target

[Service]   
EnvironmentFile=-/opt/kubernetes/cfg/kube-proxy   
ExecStart=/opt/kubernetes/bin/kube-proxy $KUBE\_PROXY\_OPTS   
Restart=on-failure

[Install]  
WantedBy=multi-user.target  
  
启动  
# systemctl daemon-reload   
# systemctl enable kube-proxy   
# systemctl start kube-proxy

查看集群状态  
# /opt/kubernetes/bin/kubectl get node  
  
# /opt/kubernetes/bin/kubectl get cs  
  
尝试创建pod  
# /opt/kubernetes/bin/kubectl run nginx --image=nginx --replicas=3   
# /opt/kubernetes/bin/kubectl expose deployment nginx --port=88 -target-port=80 --type=NodePort  
  
  
# /opt/kubernetes/bin/kubectl describe pod nginx-64f497f8fd-fjgt2  
# /opt/kubernetes/bin/kubectl get svc  
  
  
集群部署完毕