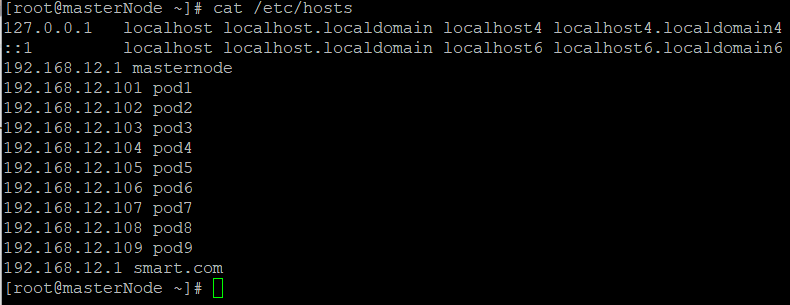
部署K8S

1 环境说明：

masternode：192.168.12.1

pods：192.168.12.101-109



2 安装配置etcd

2.1 etcd安装（https://github.com/coreos/etcd/releases）

wget https://github.com/coreos/etcd/releases/download/v3.2.11/etcd-v3.2.11-linux-amd64.tar.gz

tar zxvf etcd-v3.2.11-linux-amd64.tar.gz

cd etcd-v3.2.11-linux-amd64

mv etcd etcdctl /usr/bin/

2.2 创建etcd证书

cd /opt/ssl/

vi etcd-csr.json

{

"CN": "smartCloud",

"hosts": [

"127.0.0.1",

"192.168.12.1",

"192.168.12.101",

"192.168.12.102",

"192.168.12.103",

"192.168.12.104",

"192.168.12.105",

"192.168.12.106",

"192.168.12.107",

"192.168.12.108",

"192.168.12.109"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"L": "Shaanxi",

"ST": "Xi'an",

"O": "k8s",

"OU": "System"

}

]

}

2.3 生成etcd密钥

# 生成密钥

/opt/local/cfssl/cfssl gencert -ca=/opt/ssl/ca.pem \

-ca-key=/opt/ssl/ca-key.pem \

-config=/opt/ssl/config.json \

-profile=kubernetes etcd-csr.json | /opt/local/cfssl/cfssljson -bare etcd

# 查看生成

ls etcd\*



# 拷贝到etcd服务器

cp etcd\*.pem /etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.101:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.102:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.103:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.104:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.105:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.106:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.107:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.108:/etc/kubernetes/ssl/

scp etcd\*.pem 192.168.12.109:/etc/kubernetes/ssl/

2.4 修改 etcd配置（所有主机）

# 授予修改权限

useradd etcd

mkdir -p /opt/etcd

chown -R etcd:etcd /opt/etcd

# etcd-1

cd /usr/lib/systemd/system

vi etcd.service

[Unit]

Description=Etcd Server

After=network.target

After=network-online.target

Wants=network-online.target

[Service]

Type=notify

WorkingDirectory=/var/lib/etcd/

User=etcd

# set GOMAXPROCS to number of processors

ExecStart=/usr/bin/etcd \

--name=etcd1 \

--cert-file=/etc/kubernetes/ssl/etcd.pem \

--key-file=/etc/kubernetes/ssl/etcd-key.pem \

--peer-cert-file=/etc/kubernetes/ssl/etcd.pem \

--peer-key-file=/etc/kubernetes/ssl/etcd-key.pem \

--trusted-ca-file=/etc/kubernetes/ssl/ca.pem \

--peer-trusted-ca-file=/etc/kubernetes/ssl/ca.pem \

--initial-advertise-peer-urls=https://192.168.12.1:2380 \

--listen-peer-urls=https://192.168.12.1:2380 \

--listen-client-urls=https://192.168.12.1:2379,http://127.0.0.1:2379 \

--advertise-client-urls=https://192.168.12.1:2379 \

--initial-cluster-token=k8s-etcd-cluster \

--initial-cluster=etcd1=https://192.168.12.1:2380,etcd2=https://192.168.12.101:2380,etcd3=https://192.168.12.102:2380,etcd4=https://192.168.12.103:2380,etcd5=https://192.168.12.104:2380,etcd6=https://192.168.12.105:2380,etcd7=https://192.168.12.106:2380,etcd8=https://192.168.12.107:2380,etcd9=https://192.168.12.108:2380,etcd10=https://192.168.12.109:2380 \

--initial-cluster-state=new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

# etcd-2

cd /usr/lib/systemd/system

vi etcd.service

[Unit]

Description=Etcd Server

After=network.target

After=network-online.target

Wants=network-online.target

[Service]

Type=notify

WorkingDirectory=/var/lib/etcd/

User=etcd

# set GOMAXPROCS to number of processors

ExecStart=/usr/bin/etcd \

--name=etcd2 \

--cert-file=/etc/kubernetes/ssl/etcd.pem \

--key-file=/etc/kubernetes/ssl/etcd-key.pem \

--peer-cert-file=/etc/kubernetes/ssl/etcd.pem \

--peer-key-file=/etc/kubernetes/ssl/etcd-key.pem \

--trusted-ca-file=/etc/kubernetes/ssl/ca.pem \

--peer-trusted-ca-file=/etc/kubernetes/ssl/ca.pem \

--initial-advertise-peer-urls=https://192.168.12.101:2380 \

--listen-peer-urls=https://192.168.12.101:2380 \

--listen-client-urls=https://192.168.12.101:2379,http://127.0.0.1:2379 \

--advertise-client-urls=https://192.168.12.101:2379 \

--initial-cluster-token=k8s-etcd-cluster \

--initial-cluster=etcd1=https://192.168.12.1:2380,etcd2=https://192.168.12.101:2380,etcd3=https://192.168.12.102:2380,etcd4=https://192.168.12.103:2380,etcd5=https://192.168.12.104:2380,etcd6=https://192.168.12.105:2380,etcd7=https://192.168.12.106:2380,etcd8=https://192.168.12.107:2380,etcd9=https://192.168.12.108:2380,etcd10=https://192.168.12.109:2380 \

--initial-cluster-state=new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

# etc3-etc10 以此类推

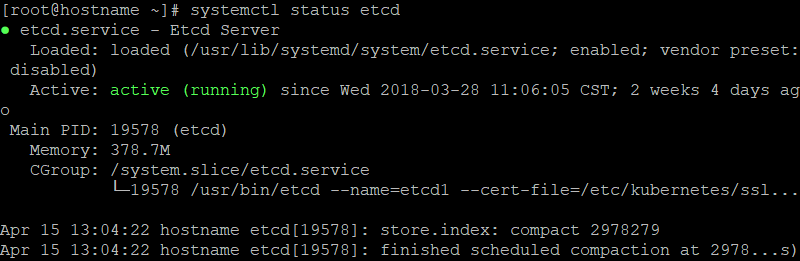
2.4 启动 etcd

systemctl daemon-reload

systemctl enable etcd

systemctl start etcd

systemctl status etcd



2.5 验证 etcd 集群状态

etcdctl --endpoints= https://192.168.12.101:2379,https://192.168.12.101:2379,https://192.168.12.102:2379,https://192.168.12.103:2379,https://192.168.12.104:2379,https://192.168.12.105:2379,https://192.168.12.106:2379,https://192.168.12.107:2379, https://192.168.12.108:2379,https://192.168.12.109:2379\

--cert-file=/etc/kubernetes/ssl/etcd.pem \

--ca-file=/etc/kubernetes/ssl/ca.pem \

--key-file=/etc/kubernetes/ssl/etcd-key.pem \

cluster-health

3 配置 Flannel 网络

kubernetes 要求集群内各节点能通过 Pod 网段互联互通，本章节介绍使用 Flannel 在**所有节点** (Master、Node) 上创建互联互通的 Pod 网段的步骤。

3.1 安装&配置 Flannel

rpm -ivh flannel-0.9.1-1.x86\_64.rpm

# 或

mkdir flannel

wget https://github.com/coreos/flannel/releases/download/v0.7.1/flannel-v0.7.1-linux-amd64.tar.gz

tar -xzvf flannel-v0.7.1-linux-amd64.tar.gz -C flannel

sudo cp flannel/{flanneld,mk-docker-opts.sh} /root/local/bin

3.2 配置 Flannel

# 配置 flannel， 由于我们docker更改了 docker.service.d 的路径， 所以这里把 flannel.conf 的配置拷贝到 这个目录去

mv /usr/lib/systemd/system/docker.service.d/flannel.conf /etc/systemd/system/docker.service.d

# 配置 flannel 网段

etcdctl --endpoints= https://192.168.12.101:2379,https://192.168.12.101:2379,https://192.168.12.102:2379,https://192.168.12.103:2379,https://192.168.12.104:2379,https://192.168.12.105:2379,https://192.168.12.106:2379,https://192.168.12.107:2379, https://192.168.12.108:2379,https://192.168.12.109:2379\

--cert-file=/etc/kubernetes/ssl/etcd.pem \

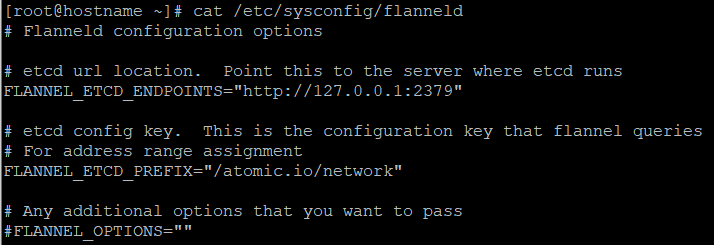
--ca-file=/etc/kubernetes/ssl/ca.pem \

--key-file=/etc/kubernetes/ssl/etcd-key.pem \

set /flannel/network/config \ '{"Network":"10.254.64.0/18","SubnetLen":24,"Backend":{"Type":"host-gw"}}'

# 修改 flanneld 配置

vi /etc/sysconfig/flannel



# etcd 地址

FLANNEL\_ETCD\_ENDPOINTS=" https://192.168.12.101:2379,https://192.168.12.101:2379,https://192.168.12.102:2379,https://192.168.12.103:2379,https://192.168.12.104:2379,https://192.168.12.105:2379,https://192.168.12.106:2379,https://192.168.12.107:2379, https://192.168.12.108:2379,https://192.168.12.109:2379"

# 配置为上面的路径 flannel/network

FLANNEL\_ETCD\_PREFIX="/flannel/network"

3.3 启动 flannel

# 启动 flannel

systemctl daemon-reload

systemctl enable flanneld

systemctl start flanneld

systemctl status flannel

# 重启 kubelet

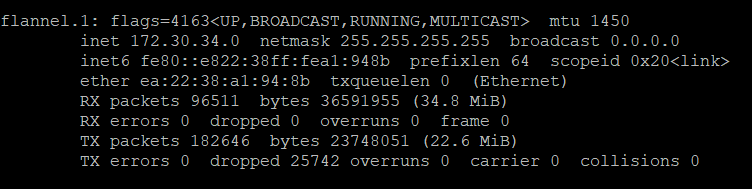
systemctl daemon-reload

systemctl restart kubelet

systemctl status kubelet

3.4 查看 flannel 验证 网络

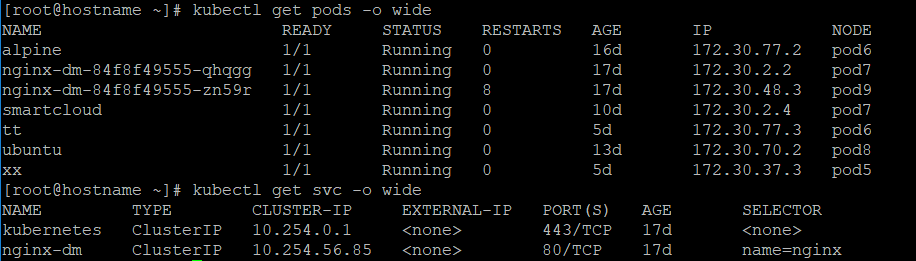
ifconfig 查看 docker0 网络 是否已经更改为配置IP网段



# 测试集群

kubectl get pods -o wide

kubectl get svc -o wide



4 配置 Kubernetes 集群

* kubectl 安装在所有需要进行操作的机器上
* Master 需要部署 kube-apiserver , kube-scheduler , kube-controller-manager 这三个组件。
* kube-scheduler 作用是调度pods分配到那个node里，简单来说就是资源调度。
* 同时只能有一个 kube-scheduler、kube-controller-manager 进程处于工作状态，如果运行多个，则需要通过选举产生一个 leader；
* kube-controller-manager 作用是 对 deployment controller , replication controller, endpoints controller, namespace controller, and serviceaccounts controller等等的循环控制，与kube-apiserver交互。

4.1 组件安装

cd /tmp

wget https://dl.k8s.io/v1.9.0/kubernetes-server-linux-amd64.tar.gz

tar -xzvf kubernetes-server-linux-amd64.tar.gz

cd kubernetes

cp -r server/bin/{kube-apiserver,kube-controller-manager,kube-scheduler,kubectl} /usr/local/bin/

scp server/bin/{kube-apiserver,kube-controller-manager,kube-scheduler,kubectl,kube-proxy,kubelet} 192.168.12.1:/usr/local/bin/

scp server/bin/{kube-proxy,kubelet} 192.168.12.101:/usr/local/bin/

4.2 创建admin证书

cd /opt/ssl/

vi admin-csr.json

{

"CN": "admin",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Shaanxi",

"L": "Xi'an",

"O": "system:masters",

"OU": "System"

}

]

}

# 生成 admin 证书和私钥

cd /opt/ssl/

/opt/local/cfssl/cfssl gencert -ca=/etc/kubernetes/ssl/ca.pem \

-ca-key=/etc/kubernetes/ssl/ca-key.pem \

-config=/opt/ssl/config.json \

-profile=kubernetes admin-csr.json | /opt/local/cfssl/cfssljson -bare admin

# 查看生成

ls admin\*

admin.csr admin-csr.json admin-key.pem admin.pem

cp admin\*.pem /etc/kubernetes/ssl/

scp admin\*.pem 192.168.12.1:/etc/kubernetes/ssl/

4.3 配置 kubectl kubeconfig 文件

# 配置 kubernetes 集群

kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=https://127.0.0.1:6443

# 配置 客户端认证

kubectl config set-credentials admin \

--client-certificate=/etc/kubernetes/ssl/admin.pem \

--embed-certs=true \

--client-key=/etc/kubernetes/ssl/admin-key.pem

kubectl config set-context kubernetes \

--cluster=kubernetes \

--user=admin

kubectl config use-context kubernetes

4.4 创建 kubernetes 证书

## 这里 hosts 字段中 三个 IP 分别为 127.0.0.1 本机， 192.168.12.1为 Master 的IP，多个Master需要写多个。10.254.0.1 为 kubernetes SVC 的 IP， 一般是 部署网络的第一个IP , 如: 10.254.0.1 ， 在启动完成后，我们使用 kubectl get svc ， 就可以查看到

cd /opt/ssl

vi kubernetes-csr.json

{

"CN": "kubernetes",

"hosts": [

"127.0.0.1",

"192.168.12.1",

"10.254.0.1",

"kubernetes",

"k8s-api.virtual.local",

"kubernetes.default",

"kubernetes.default.svc",

"kubernetes.default.svc.cluster",

"kubernetes.default.svc.cluster.local"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Shaanxi",

"L": "Xi'an",

"O": "k8s",

"OU": "System"

}

]

}

4.5 生成 kubernetes 证书和私钥

/opt/local/cfssl/cfssl gencert -ca=/etc/kubernetes/ssl/ca.pem \

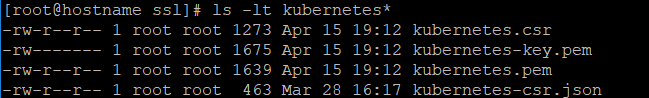
-ca-key=/etc/kubernetes/ssl/ca-key.pem \

-config=/opt/ssl/config.json \

-profile=kubernetes kubernetes-csr.json | /opt/local/cfssl/cfssljson -bare kubernetes

# 查看生成

ls -lt kubernetes\*



# 拷贝到目录

cp kubernetes\*.pem /etc/kubernetes/ssl/

scp kubernetes\*.pem 192.168.12.1:/etc/kubernetes/ssl/

4.6 配置 kube-apiserver

# 自定义 系统 service 文件一般存于 /etc/systemd/system/ 下

# 配置为 各自的本地 IP

vi /etc/systemd/system/kube-apiserver.service

[Unit]

Description=Kubernetes API Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=network.target

[Service]

User=root

ExecStart=/usr/bin/kube-apiserver \

--admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,ResourceQuota,NodeRestriction \

--advertise-address=**192.168.12.1** \

--allow-privileged=true \

--apiserver-count=3 \

--audit-policy-file=/etc/kubernetes/audit-policy.yaml \

--audit-log-maxage=30 \

--audit-log-maxbackup=3 \

--audit-log-maxsize=100 \

--audit-log-path=/var/log/kubernetes/audit.log \

--authorization-mode=Node,RBAC \

--bind-address=**192.168.12.1** \

--secure-port=6443 \

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

--enable-swagger-ui=true \

--etcd-cafile=/etc/kubernetes/ssl/ca.pem \

--etcd-certfile=/etc/kubernetes/ssl/etcd.pem \

--etcd-keyfile=/etc/kubernetes/ssl/etcd-key.pem \

--etcd-servers=**https://192.168.12.1:2379,https://192.168.12.101:2379, https://192.168.12.102:2379, https://192.168.12.103:2379, https://192.168.12.104:2379, https://192.168.12.105:2379, https://192.168.12.106:2379, https://192.168.12.107:2379, https://192.168.12.108:2379, https://192.168.12.109:2379 \**

--event-ttl=1h \

--kubelet-https=true \

--insecure-bind-address=**192.168.12.1** \

--insecure-port=8080 \

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

--service-cluster-ip-range=10.254.0.0/18 \

--service-node-port-range=30000-32000 \

--tls-cert-file=/etc/kubernetes/ssl/kubernetes.pem \

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

--enable-bootstrap-token-auth \

--token-auth-file=/etc/kubernetes/token.csv \

--v=2

Restart=on-failure

RestartSec=5

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

4.7 启动 kube-apiserver

systemctl daemon-reload

systemctl enable kube-apiserver

systemctl start kube-apiserver

systemctl status kube-apiserver

4.8 配置 kube-controller-manager

# 创建 kube-controller-manager.service 文件

vi /etc/systemd/system/kube-controller-manager.service

[Unit]

Description=Kubernetes Controller Manager

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

[Service]

ExecStart=/usr/bin/kube-controller-manager \

--address=**127.0.0.1** \

--master=http://**192.168.12.1:8080** \

--allocate-node-cidrs=true \

--service-cluster-ip-range=10.254.0.0/16 \

--cluster-cidr=**170.30.0.0/16** \

--cluster-name=kubernetes \

--cluster-signing-cert-file=/etc/kubernetes/ssl/ca.pem \

--cluster-signing-key-file=/etc/kubernetes/ssl/ca-key.pem \

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

--root-ca-file=/etc/kubernetes/ssl/ca.pem \

--leader-elect=true \

--v=2

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

4.9 启动 kube-controller-manager

systemctl daemon-reload

systemctl enable kube-controller-manager

systemctl start kube-controller-manager

systemctl status kube-controller-manager

4.10 配置 kube-scheduler

# 创建 kube-cheduler.service 文件

vi /etc/systemd/system/kube-scheduler.service

[Unit]

Description=Kubernetes Scheduler

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

[Service]

ExecStart=/usr/bin/kube-scheduler \

--address=**127.0.0.1** \

--master=**http://192.168.12.1:8080** \

--leader-elect=true \

--v=2

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

4.11 启动 kube-scheduler

systemctl daemon-reload

systemctl enable kube-scheduler

systemctl start kube-scheduler

systemctl status kube-scheduler

# 验证Master节点

kubectl get componentstatuses

4.12 配置 kubelet（for nodes）

# 先创建认证请求 只需创建一次就可以

kubectl create clusterrolebinding kubelet-bootstrap --clusterrole=system:node-bootstrapper --user=kubelet-bootstrap

# 配置集群

kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=https://**192.168.12.1:6443** \

--kubeconfig=bootstrap.kubeconfig

# 配置客户端认证

kubectl config set-credentials kubelet-bootstrap \

--token=d2d7f3a19490ff667fbe94b0f31f9967 \

--kubeconfig=bootstrap.kubeconfig

# 配置关联

kubectl config set-context default \

--cluster=kubernetes \

--user=kubelet-bootstrap \

--kubeconfig=bootstrap.kubeconfig

# 配置默认关联

kubectl config use-context default --kubeconfig=bootstrap.kubeconfig

# 拷贝生成的 bootstrap.kubeconfig 文件

mv bootstrap.kubeconfig /etc/kubernetes/

# 创建 kubelet 目录，配置为 node 本机 IP

mkdir /var/lib/kubelet

vi /etc/systemd/system/kubelet.service

[Unit]

Description=Kubernetes Kubelet

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=docker.service

Requires=docker.service

[Service]

WorkingDirectory=/var/lib/kubelet

ExecStart=/usr/local/bin/kubelet \

--cgroup-driver=cgroupfs \

--hostname-override=**pod** \

--pod-infra-container-image=jicki/pause-amd64:3.0 \

--experimental-bootstrap-kubeconfig=/etc/kubernetes/bootstrap.kubeconfig \

--kubeconfig=/etc/kubernetes/kubelet.kubeconfig \

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

--cluster\_dns=10.254.0.2 \

--cluster\_domain=cluster.local. \

--hairpin-mode promiscuous-bridge \

--allow-privileged=true \

--fail-swap-on=false \

--serialize-image-pulls=false \

--logtostderr=true \

--max-pods=512 \

--v=2

[Install]

WantedBy=multi-user.target

# 启动 kubelet

systemctl daemon-reload

systemctl enable kubelet

systemctl start kubelet

systemctl status kubelet

# 查看 csr 的名称

kubectl get csr

# 增加 认证

kubectl get csr | grep Pending | awk '{print $1}' | xargs kubectl certificate approve

# 验证 nodes

kubectl get nodes

4.13 配置 kube-proxy

# 创建 kube-proxy 证书

cd /opt/ssl

vi kube-proxy-csr.json

{

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

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Shaanxi",

"L": "Xi'an",

"O": "k8s",

"OU": "System"

}

]

}

4.14 生成 kube-proxy 证书和私钥

/opt/local/cfssl/cfssl gencert -ca=/etc/kubernetes/ssl/ca.pem \

-ca-key=/etc/kubernetes/ssl/ca-key.pem \

-config=/opt/ssl/config.json \

-profile=kubernetes kube-proxy-csr.json | /opt/local/cfssl/cfssljson -bare kube-proxy

# 查看生成

ls kube-proxy\*

kube-proxy.csr kube-proxy-csr.json kube-proxy-key.pem kube-proxy.pem

# 拷贝到目录

scp kube-proxy\*.pem **192.168.12.101**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.102**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.103**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.104**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.105**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.106**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.107**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.108**:/etc/kubernetes/ssl/

scp kube-proxy\*.pem **192.168.12.109**:/etc/kubernetes/ssl/

4.15 创建 kube-proxy kubeconfig 文件

# 配置集群

kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=https://127.0.0.1:6443 \

--kubeconfig=kube-proxy.kubeconfig

# 配置客户端认证

kubectl config set-credentials kube-proxy \

--client-certificate=/etc/kubernetes/ssl/kube-proxy.pem \

--client-key=/etc/kubernetes/ssl/kube-proxy-key.pem \

--embed-certs=true \

--kubeconfig=kube-proxy.kubeconfig

# 配置关联

kubectl config set-context default \

--cluster=kubernetes \

--user=kube-proxy \

--kubeconfig=kube-proxy.kubeconfig

# 配置默认关联

kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig

# 拷贝到需要的 node 端里

scp kube-proxy.kubeconfig **192.168.12.101**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.102**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.103**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.104**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.105**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.106**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.107**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.108**:/etc/kubernetes/

scp kube-proxy.kubeconfig **192.168.12.109**:/etc/kubernetes/

4.16 创建 kube-proxy.service 文件

# 创建 kube-proxy 目录

mkdir -p /var/lib/kube-proxy

vi /etc/systemd/system/kube-proxy.service

[Unit]

Description=Kubernetes Kube-Proxy Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=network.target

[Service]

WorkingDirectory=/var/lib/kube-proxy

ExecStart=/usr/local/bin/kube-proxy \

--bind-address=**192.168.12.101** \

--hostname-override=pod2 \

--cluster-cidr=10.254.64.0/16 \

--masquerade-all \

--feature-gates=SupportIPVSProxyMode=true \

--proxy-mode=ipvs \

--ipvs-min-sync-period=5s \

--ipvs-sync-period=5s \

--ipvs-scheduler=rr \

--kubeconfig=/etc/kubernetes/kube-proxy.kubeconfig \

--logtostderr=true \

--v=2

Restart=on-failure

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

4.17 启动 kube-proxy

systemctl daemon-reload

systemctl enable kube-proxy

systemctl start kube-proxy

systemctl status kube-proxy

# 检查 ipvs

ipvsadm -L -n

4.18 Node 端配置——nginx

单 Node 部分 需要部署的组件有 docker calico kubelet kube-proxy 这几个组件。 Node 节点 基于 Nginx 负载 API 做 Master HA

# 发布证书 ALL node

mkdir -p /etc/kubernetes/ssl/

scp ca.pem kube-proxy.pem kube-proxy-key.pem node-\*:/etc/kubernetes/ssl/

# 创建Nginx 代理

# 创建配置目录

mkdir -p /etc/nginx

# 写入代理配置

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

error\_log stderr notice;

worker\_processes auto;

events {

multi\_accept on;

use epoll;

worker\_connections 1024;

}

stream {

upstream kube\_apiserver {

least\_conn;

server **192.168.12.101**:6443;

server **192.168.12.102**:6443;

server **192.168.12.103**:6443;

server **192.168.12.104**:6443;

server **192.168.12.105**:6443;

server **192.168.12.106**:6443;

server **192.168.12.107**:6443;

server **192.168.12.108**:6443;

server **192.168.12.109**:6443;

}

server {

listen 0.0.0.0:6443;

proxy\_pass kube\_apiserver;

proxy\_timeout 10m;

proxy\_connect\_timeout 1s;

}

}

EOF

# 更新权限

chmod +r /etc/nginx/nginx.conf

# 配置 Nginx 基于 docker 进程，然后配置 systemd 来启动

cat << EOF >> /etc/systemd/system/nginx-proxy.service

[Unit]

Description=kubernetes apiserver docker wrapper

Wants=docker.socket

After=docker.service

[Service]

User=root

PermissionsStartOnly=true

ExecStart=/usr/bin/docker run -p 127.0.0.1:6443:6443 \\

-v /etc/nginx:/etc/nginx \\

--name nginx-proxy \\

--net=host \\

--restart=on-failure:5 \\

--memory=512M \\

nginx:1.13.7-alpine

ExecStartPre=-/usr/bin/docker rm -f nginx-proxy

ExecStop=/usr/bin/docker stop nginx-proxy

Restart=always

RestartSec=15s

TimeoutStartSec=30s

[Install]

WantedBy=multi-user.target

EOF

# 启动 Nginx

systemctl daemon-reload

systemctl start nginx-proxy

systemctl enable nginx-proxy

systemctl status nginx-proxy

# 配置Kubelet.service & kube-proxy.service文件（略）