K8s master节点部署

1. master节点服务部署内容

master节点在部署时包含的服务为apiserver、scheduler controller-manager、docker。

1. master节点证书部署。
2. ca证书生成,先创建ca请求文件。

[root@fp-web-15 ssl]# cat k8s-root-ca-csr.json

{

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 4096

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

1. 生成ca文件

# 生成 CA

cfssl gencert --initca=true k8s-root-ca-csr.json | cfssljson --bare k8s-root-ca

1. 创建api请求文件。

{

"CN": "kubernetes",

"hosts": [

"127.0.0.1",

"172.17.0.1",

"10.129.55.61",

"10.129.55.65",

"10.129.55.155",

"master1",

"master2",

"master3",

"\*.kubernetes.master",

"localhost",

"kubernetes",

"kubernetes.default",

"kubernetes.default.svc",

"kubernetes.default.svc.cluster",

"kubernetes.default.svc.cluster.local"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

api文件中需要写入所有服务器ip地址，同时需要写入主机名。

1. 生成admin证书请求。

{

"CN": "admin",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "system:masters",

"OU": "System"

}

]

}

1. 创建CA签名配置文件。

{

"signing": {

"default": {

"expiry": "87600h"

},

"profiles": {

"kubernetes": {

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth"

],

"expiry": "87600h"

}

}

}

}

1. 生成书证内容，包含证书的私钥和证书和csr。

# 依次生成其他组件证书

for targetName in kube-apiserver admin; do

cfssl gencert --ca k8s-root-ca.pem --ca-key k8s-root-ca-key.pem --config k8s-gencert.json --profile kubernetes $targetName-csr.json | cfssljson --bare $targetName

Done

1. 创建kube-api配置文件apiserver。

KUBE\_API\_ADDRESS="--advertise-address=10.129.55.65 --bind-address=10.129.55.65"

KUBE\_API\_PORT="--secure-port=6443"

KUBE\_ETCD\_SERVERS="--etcd-servers=https://10.129.55.61:2379,https://10.129.55.65:2379,https://10.129.55.155:2379"

KUBE\_SERVICE\_ADDRESSES="--service-cluster-ip-range=172.17.0.0/16"

KUBE\_ADMISSION\_CONTROL="--enable-admission-plugins=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,DefaultTolerationSeconds,MutatingAdmissionWebhook,ValidatingAdmissionWebhook,ResourceQuota,NodeRestriction"

KUBE\_API\_ARGS=" --anonymous-auth=false \

--apiserver-count=3 \

--audit-log-maxage=30 \

--audit-log-maxbackup=3 \

--audit-log-maxsize=100 \

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

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

--authorization-mode=Node,RBAC \

--client-ca-file=/etc/kubernetes/ssl/k8s-root-ca.pem \

--enable-bootstrap-token-auth \

--enable-garbage-collector \

--enable-logs-handler \

--enable-swagger-ui \

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

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

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

--etcd-compaction-interval=5m0s \

--etcd-count-metric-poll-period=1m0s \

--event-ttl=48h0m0s \

--kubelet-https=true \

--kubelet-timeout=3s \

--max-requests-inflight=800 \

--log-flush-frequency=5s \

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

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

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

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

--service-account-key-file=/etc/kubernetes/ssl/k8s-root-ca.pem \

--storage-backend=etcd3 \

--enable-swagger-ui=true"

1. 创建连接api的配置文件config。

cat config

KUBE\_LOGTOSTDERR="--logtostderr=true"

KUBE\_LOG\_LEVEL="--v=2"

KUBE\_ALLOW\_PRIV="--allow-privileged=true"

KUBE\_MASTER="--master=http://127.0.0.1:8080"

1. 创建controller-manager配置文件。

KUBE\_CONTROLLER\_MANAGER\_ARGS=" --bind-address=0.0.0.0 \

--cluster-name=kubernetes \

--cluster-signing-cert-file=/etc/kubernetes/ssl/k8s-root-ca.pem \

--cluster-signing-key-file=/etc/kubernetes/ssl/k8s-root-ca-key.pem \

--controllers=\*,bootstrapsigner,tokencleaner \

--deployment-controller-sync-period=10s \

--experimental-cluster-signing-duration=86700h0m0s \

--leader-elect=true \

--node-monitor-grace-period=40s \

--node-monitor-period=5s \

--pod-eviction-timeout=1m0s \

--terminated-pod-gc-threshold=50 \

--root-ca-file=/etc/kubernetes/ssl/k8s-root-ca.pem \

--service-account-private-key-file=/etc/kubernetes/ssl/k8s-root-ca-key.pem \

--feature-gates=RotateKubeletServerCertificate=true"

1. 创建scheduler配置文件。

KUBE\_SCHEDULER\_ARGS=" --address=0.0.0.0 \

--leader-elect=true \ --algorithm-provider=DefaultProvider"

1. 安装docker服务。

安装docker依赖

yum install -y yum-utils device-mapper-persistent-data lvm2

安装docker服务

Yum localinstall /root/k8s/package/docker-ce-18.03.0.ce-1.el7.centos.x86\_64.rpm -y

1. 创建 TLS Bootstrapping Toke。

TLS bootstrapping 功能就是让 kubelet 先使用一个预定的低权限用户连接到 apiserver，然后向 apiserver 申请证书，kubelet 的证书由 apiserver 动态签署。

Token可以是任意的包涵128 bit的字符串，可以使用安全的随机数发生器生成。

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

f1fbd57f88caadfd7f1d69d79b6d7275

#f1fbd57f88caadfd7f1d69d79b6d7275:随机字符串,自定义生成; kubelet-bootstrap:用户名; 10001:UID; system:kubelet-bootstrap：用户组

cat /etc/kubernetes/conf/token.csv

f1fbd57f88caadfd7f1d69d79b6d7275,kubelet-bootstrap,10001,"system:bootstrappers"

1. 一次性导入本地下载好的k8部署所需全部docker镜像。

ssh root@10.129.55.65 "for i in \$(ls /root/k8s/image);do docker load -i /root/k8s/image/\${i};done"

1. 拷贝文件到其他两台服务器。

复制10.129.55.165上面的二进制文件/root/k8s/package/hyperkube和/root/k8s/ssl下的证书文件和/root/k8s/conf到三台服务器。

1. 启动服务。

Docker启动

chkconfig docker on

service docker start

systemctl start docker.service

启动k8s 服务

systemctl start kube-apiserver.service

systemctl start kube-scheduler.service

systemctl start kube-controller-manager.service

1. master节点部署完成后检查集群状态：

[root@fpNet-WEB-0 system]# 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"}

1. 创建clusterrolebinding将用户kubelet-bootstrap绑定系统权限system:node-bootstrapper。

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