**K8s master集群部署**

K8smaster集群各节点部署前，需要按照k8s集群部署要求进行检查。

需下载安装初始化工具。

1. 需要通过yum安装的软件

net-tools vim wget lrzsz git cfssl cfssljson expect

yum-utils device-mapper-persistent-data lvm2 telnet ntpdate

1. 安装包中已存在，无需下载的软件

hyperkube docker-ce calico-node calico-cni calico-kube-controllers

1. docker镜像

Nginx-1.13.12-alpine.tar calico-node.tar calico-kube-controllers.tar pause-amd64.3.0.tar dashboard.tar coredns.tar cni-v3.1.3.tar

1. 关闭防火墙、selinx。
2. 优化内核。
3. 保证网络正常通讯。
4. 保证网卡、路由配置正确。
5. 保证时区和时间同步正常。
6. 保证操作系统版本符合要求。
7. 保证各个服务器主机文件配置主机名。
8. 保证ssh设置免密互通（k8s集群内所有主机设置免密）。
9. k8s集群部署拓扑结构。

|  |  |  |  |
| --- | --- | --- | --- |
| IP | 系统版本 | 作用 | 备注 |
| 10.129.51.132 | CentOS Linux release 7.2.1511 | Master1 | 该机器作为操作节点 |
| 10.129.51.117 | CentOS Linux release 7.2.1511 | Master2 |  |
| 10.129.51.26 | CentOS Linux release 7.2.1511 | Node | Node节点 |
| 10.129.52.142 | CentOS Linux release 7.2.1511 | Etcd1 | Etcd集群 |
| 10.129.52.162 | CentOS Linux release 7.2.1511 | Etcd2 |
| 10.129.52.194 | CentOS Linux release 7.2.1511 | Etcd3 |
| 10.129.52.120 | CentOS Linux release 7.2.1511 | ansible | 用于远程批量操作，如：监测网络互通性 |

1. master节点10.129.51.132免密配置。

需要进行免密的服务器

|  |  |  |
| --- | --- | --- |
| Ip | 用途 | 备注 |
| 10.129.51.132 | 主操作节点master | 以142为例，其他节点复制142的生成证书和二进制文件 |
| 10.129.51.117 | Master |  |
| 10.129.51.26 | Node |  |

expect -c "

spawn ssh-copy-id -i /root/.ssh/id\_rsa.pub root@$i

expect {

\"\*yes/no\*\" {send \"yes\r\"; exp\_continue}

\"\*password\*\" {send \"$mypass\r\"; exp\_continue}

\"\*Password\*\" {send \"$mypass\r\";}

}"

1. 时区设置。

设置所有节点的时区为Shanghai

\cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime -rf

##时区设置和基础包安装，关闭防火墙selinux，ntp时间同步。

ssh root@$i "yum install -y ntpdate net-tools vim wget lrzsz git telnet bash-completion;ntpdate -u 10.129.80.80;systemctl stop firewalld;systemctl disable firewalld;sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config;cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime -rf;reboot"

1. 主节点10.129.51.132的相关操作步骤。
2. Master配置的相关目录（所有节点配置目录需预生成）

|  |  |  |
| --- | --- | --- |
| 目录 | 内容 | 备注 |
| /etc/kubernetes/ssl | Master的admin(kubelet)、kube-apiserver的ca证书文件 | 1. 通过相关的json文件及master-ca.pem master-ca-key.pem生成相关的证书（拷贝自output目录） 2. 通过kubelet服务第一次启动时自动生成的kubelet-client.crt、kubelet-client.key文件。（如果以克隆方式添加节点，新机器需要删除这两个文件，方能成功启动kubelet服务） |
| /etc/nginx | Nginx反向代理 | 反向代理kube-apiserver |
| /etc/calico | 放置etcd集群生成的证书 | 该目录本机不用创建，由etcd集群创建 |
| /var/lib/calico | 节点名称（nodename） | Nodename文件由calico服务启动时自动生成 |
| /var/run/calico | bird.ctl  bird6.ctl | 由本机calico服务所创建的calico容器产生。   1. bird.ctl是由容器中的bird.cfg所产生； 2. bird6.ctl是由容器中的bird6.cfg产生（calico网络配置文件） |
| /opt/cni/bin | k8s二进制程序 | K8s网络相关命令（该目录是二进制程序，所有node节点是通过载入cni-v3.1.3.tar的镜像后自动生成。只需保证有该目录且每个master节点执行kubectl apply -f calico.yaml） |
| /etc/cni/net.d | CNI默认的网络配置文件目录，当没有特别指定时，CNI就会默认对该目录进行查找，从中加载配置文件进行容器网络的创建 | 通过kubectl apply -f calico.yaml命令创建的calico-node（类型为：DaemonSet）会根据yaml中的配置自动在该目录下生成网络配置文件 |
| /etc/etcd/ssl | Etcd集群创建时生成的证书文件通过远程传输到该目录（非master创建） | 该目录本机不用创建。由etcd集群创建 |
| /var/log/kube-audit/ | 审计的日志文件 | 日志文件 |

主节点操作命令如下：

mkdir -p /etc/kubernetes/ssl /etc/nginx /var/lib/calico /opt/cni/bin /etc/cni/net.d /var/run/calico /var/log/kube-audit/

Master主机远程在节点上执行：

ssh root@10.129.51.117 "mkdir -p /etc/kubernetes/ssl /etc/nginx /var/lib/calico /opt/cni/bin /etc/cni/net.d /var/run/calico /var/log/kube-audit/"

ssh root@10.129.51.26 "mkdir -p /etc/kubernetes/ssl /etc/nginx /var/lib/calico /opt/cni/bin /etc/cni/net.d /var/run/calico /var/log/kube-audit/"

五、创建生成CA证书的基础json文件k8s-gencert.json和**master-ca-csr.json（该文件又名请求文件）**

{

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 4096

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

六、 根据ca的基础json文件生成master的ca证书

1）生成master的ca证书文件，命令如下：

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

生成的文件如下：

master-ca.pem master-ca-key.pem master-ca.csr

1. 生成kube-apiserver组件的证书

创建kube-apiserver的证书请求文件k8s-gencert.json和kube-apiserver-csr.json。

{

"CN": "kubernetes",

"hosts": [

"127.0.0.1",

"172.17.0.1",

"10.129.51.132",

"10.129.51.117",

"master1",

"master2",

"10.129.52.142",

“10.129.52.162”,

“10.129.52.194”,

“etcd1”,

“etcd2”,

“etcd3”,

“10.129.51.26”,

“node”

"\*.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"

}

]

}

k8s-gencert.json。

{

"signing": {

"default": {

"expiry": "87600h"

},

"profiles": {

"kubernetes": {

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth"

],

"expiry": "87600h"

}

}

}

}

生成kube-apiserver组件证书的命令如下：

cfssl gencert --ca master-ca.pem --ca-key master-ca-key.pem --config k8s-gencert.json --profile kubernetes kube-apiserver-csr.json | cfssljson --bare master-apiserver

生成的kube-apiserver组件的证书如下：

master-apiserver.csr master-apiserver-key.pem master-apiserver.pem

1. 创建kubectl的证书请求文件admin-csr.json。

{

"CN": "admin",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "system:masters",

"OU": "System"

}

]

}

生成kubectl证书命令如下：

cfssl gencert --ca master-ca.pem --ca-key master-ca-key.pem --config k8s-gencert.json --profile kubernetes admin-csr.json | cfssljson --bare master-admin

生成的kubectl证书文件如下：

master-admin.csr master-admin-key.pem master-admin.pem

八、 创建集群（master）审计策略文件及token

1.创建审计策略文件 /etc/kubernetes/conf/audit-policy.yaml

# Log all requests at the Metadata level.

apiVersion: audit.k8s.io/v1beta1

kind: Policy

rules:

- level: Metadata

创建token.csv

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

vim /etc/kubernetes/conf/token.csv

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

1. 创建master的kube-apiserver组件的配置文件/etc/kubernetes/conf/apiserver（包含master，etcd除外）

KUBE\_API\_ADDRESS="--advertise-address=10.129.51.132 --bind-address=10.129.51.132"

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

KUBE\_ETCD\_SERVERS="--etcd-servers=https://10.129.52.142:2379,https://10.129.52.162:2379,https://10.129.52.194:2379"

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/master-ca.pem \

--enable-bootstrap-token-auth \

--enable-garbage-collector \

--enable-logs-handler \

--enable-swagger-ui \

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

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

--etcd-keyfile=/etc/etcd/ssl/etcd-server-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/master-apiserver.pem \

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

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

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

--storage-backend=etcd3 \

--enable-swagger-ui=true"

九、 创建kube-apiserver、kube-schedule、kube-controller-manager组件读取的配置文件/etc/kubernetes/conf/config（又名”kubernetes system config”）。

文件内容如下：

KUBE\_LOGTOSTDERR="--logtostderr=true"

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

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

1. 创建kube-controller-manager组件配置文件/etc/kubernetes/conf/controller-manager。

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

--cluster-name=kubernetes \

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

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

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

--feature-gates=RotateKubeletServerCertificate=true"

1. 创建kube-scheduler组件的配置文件/etc/kubernetes/conf/scheduler。

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

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

十二、 创建nginx反向代理配置文件/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 10.129.51.132:6443;

server 10.129.51.117:6443;

}

server {

listen 0.0.0.0:6443;

proxy\_pass kube\_apiserver;

proxy\_timeout 10m;

proxy\_connect\_timeout 1s;

}

}

十三、 拷贝所有二进制文件、生成的ca证书文件、基本配置文件、启动文件到本机及其他master节点上。

1. 拷贝二进制文件

命令如下：

scp -rp /k8sInstallV11013/masterInstall/software/hyperkube root@10.129.51.132：/usr/local/bin

scp -rp /k8sInstallV11013/masterInstall/software/hyperkube root@10.129.51.117：/usr/local/bin

1. 拷贝证书文件

命令如下：

scp -rp /k8sInstallV11013/masterInstall/config/masterConf/ssl/output/\* root@10.129.51.132：/etc/kubernetes/ssl

scp -rp /k8sInstallV11013/masterInstall/config/masterConf/ssl/output/\* root@10.129.51.117：/etc/kubernetes/ssl

1. 拷贝基本配置文件

命令如下：

Scp -rp /k8sInstallV11013/masterInstall/config/masterConf/baseconf root@10.129.51.132：/etc/kubernetes/conf

Scp -rp /k8sInstallV11013/masterInstall/config/masterConf/baseconf root@10.129.51.117：/etc/kubernetes/conf

1. 拷贝启动文件

命令如下：

scp -rp /k8sInstallV11013/masterInstall/config/service/\* root@10.129.51.132:/usr/lib/systemd/system/

scp -rp /k8sInstallV11013/masterInstall/config/service/\* root@10.129.51.117:/usr/lib/systemd/system/

十四、 安装和启动docker服务。

1. 安装docker依赖

本机操作：

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

所有master节点及node节点需操作：

Ssh [root@10.129.51.117](mailto:root@10.129.51.117) “yum install -y yum-utils device-mapper-persistent-data lvm2”

Ssh [root@10.129.51.2](mailto:root@10.129.51.117)6 “yum install -y yum-utils device-mapper-persistent-data lvm2”

1. 安装docker服务

本机操作：

Yum localinstall /k8sInstallV11013/masterInstall/docker/software/docker-ce-18.03.0.ce-1.el7.centos.x86\_64.rpm -y

将docker安装包一次性远程传输到所有其他master节点及node节点上：

Scp -rp /k8sInstallV11013/masterInstall/docker/software/docker-ce-18.03.0.ce-1.el7.centos.x86\_64.rpm root@10.129.51.117:/k8sinstall/

3)每台机器一次性导入部署k8s所需全部docker镜像。

命令如下：

ssh root@10.129.51.132 "for i in \$(ls /k8sInstallV11013/masterInstall/docker/dockertar);do docker load -i /k8sInstallV11013/masterInstall/docker/dockertar/\${i};done"

1. 启动Docker服务。

启动所有master节点及node节点的docker服务

systemctl enable docker.service

systemctl start docker.service

十五 启动k8s 的组件服务

systemctl start kube-apiserver.service

systemctl start kube-scheduler.service

systemctl start kube-controller-manager.service

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"}

至此，k8s集群基础组件安装已经完成

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

1. 部署calico服务

部署前提条件：

1. 保证calico相关目录和启动文件已创建，具体目录如下：

/etc/calico /var/lib/calico /var/run/calico /opt/cni/bin /etc/cni/net.d

1. 保证calico-node镜像已完成导入，具体如下

calico-node.tar calico-kube-controllers.tar pause-amd64.3.0.tar cni-v3.1.3.tar

1. 保证k8s集群基础组件和etcd集群运行正常，具体如下：

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. 配置calico-node服务的启动文件

需要修改etcd集群的IP地址，节点的名称，etcd证书名称，已安装docker，

将配置好的/k8sInstallV11013/masterInstall/config/service/calico-node.service启动文件拷贝至所有master节点和node节点的/usr/lib/systemd/system目录下，命令如下：

scp -rp /k8sInstallV11013/masterInstall/config/service/calico-node.service root@10.129.51.132:/usr/lib/systemd/system/

scp -rp /k8sInstallV11013/masterInstall/config/service/calico-node.service [root@10.129.51.117:/usr/lib/systemd/system/](mailto:root@10.129.51.117:/usr/lib/systemd/system/)

scp -rp /k8sInstallV11013/masterInstall/config/service/calico-node.service root@10.129.51.126:/usr/lib/systemd/system/

calico-node.service具体内容如下：

[Unit]

Description= system service name is calico-node

After=docker.service

Requires=docker.service

[Service]

User=root

##calico连接k8s的etcd的集群信息根据需求进行修改。

Environment=ETCD\_ENDPOINTS=https://10.129.51.142:2379,https://10.129.51.162:2379,https://10.129.51.194:2379

PermissionsStartOnly=true

##docker启动后修改calico node的容器名称

ExecStart=/usr/bin/docker run --net=host --privileged --name=calico-node \

-e ETCD\_ENDPOINTS=${ETCD\_ENDPOINTS} \

-e ETCD\_CA\_CERT\_FILE=/etc/etcd/ssl/etcd-server.pem \

-e ETCD\_CERT\_FILE=/etc/etcd/ssl/etcd-ca.pem \

-e ETCD\_KEY\_FILE=/etc/etcd/ssl/etcd-ca-key.pem \

-e NODENAME=master1 \

-e IP= \

//指定网卡名称。

-e IP\_AUTODETECTION\_METHOD=can-reach=10.129.51.132 \

-e AS=64512 \

-e CLUSTER\_TYPE=k8s,bgp \

-e CALICO\_IPV4POOL\_CIDR=172.17.0.0/16 \

-e CALICO\_IPV4POOL\_IPIP=always \

-e CALICO\_LIBNETWORK\_ENABLED=true \

-e CALICO\_NETWORKING\_BACKEND=bird \

-e CALICO\_DISABLE\_FILE\_LOGGING=true \

-e FELIX\_IPV6SUPPORT=false \

-e FELIX\_DEFAULTENDPOINTTOHOSTACTION=ACCEPT \

-e FELIX\_LOGSEVERITYSCREEN=info \

-e FELIX\_IPINIPMTU=1440 \

-e FELIX\_HEALTHENABLED=true \

-e CALICO\_K8S\_NODE\_REF=master1 \

-v /etc/calico/etcd-root-ca.pem:/etc/etcd/ssl/etcd-server.pem \

-v /etc/calico/etcd-ca.pem:/etc/etcd/ssl/etcd-ca.pem \

-v /etc/calico/etcd-ca-key.pem:/etc/etcd/ssl/etcd-ca-key.pem \

-v /lib/modules:/lib/modules \

-v /var/lib/calico:/var/lib/calico \

-v /var/run/calico:/var/run/calico \

quay.io/calico/node:v3.1.0

ExecStop=/usr/bin/docker rm -f calico-node

Restart=always

RestartSec=10

[Install]

WantedBy=multi-user.target

|  |  |
| --- | --- |
| 配置项名称 | 备注 |
| Description | 文件描述：系统服务名称是calico-node |
| Environment=ETCD\_ENDPOINTS | ETCD集群ip+端口信息 |
| ExecStart | Nodename 节点的主机名名称 |
| IP\_AUTODETECTION\_METHOD 指定master的ip |
| CALICO\_K8S\_NODE\_REF 节点名称 |
| ETCD\_CA\_CERT\_FILE ETCD\_CERT\_FILE ETCD\_KEY\_FILE  etcd的相关证书名称 |
| --name docker指定容器名称 |
|  |  |

1. 启动并检查calico-node服务。

systemctl start calico-node.service

systemctl status calico-node.service

1. 创建cni插件和calico-kube-controllers服务

1)创建生成calico.yaml部署文件的脚本

/k8sInstallV11013/masterInstall/installSh/calico.sh

ETCD\_CERT=`cat /etc/etcd/ssl/etcd-ca.pem | base64 | tr -d '\n'`

ETCD\_KEY=`cat /etc/etcd/ssl/etcd-ca-key.pem | base64 | tr -d '\n'`

ETCD\_CA=`cat /etc/etcd/ssl/etcd-server.pem | base64 | tr -d '\n'`

ETCD\_ENDPOINTS="https://10.129.52.142:2380,etcd2=https://10.129.52.162:2380,etcd3=https://10.129.52.194:2380""

\cp calico.example.yaml calico.yaml

sed -i "s@.\*etcd\_endpoints:.\*@\ \ etcd\_endpoints:\ \"${ETCD\_ENDPOINTS}\"@gi" calico.yaml

sed -i "s@.\*etcd-cert:.\*@\ \ etcd-cert:\ ${ETCD\_CERT}@gi" calico.yaml

sed -i "s@.\*etcd-key:.\*@\ \ etcd-key:\ ${ETCD\_KEY}@gi" calico.yaml

sed -i "s@.\*etcd-ca:.\*@\ \ etcd-ca:\ ${ETCD\_CA}@gi" calico.yaml

sed -i 's@.\*etcd\_ca:.\*@\ \ etcd\_ca:\ "/calico-secrets/etcd-ca"@gi' calico.yaml

sed -i 's@.\*etcd\_cert:.\*@\ \ etcd\_cert:\ "/calico-secrets/etcd-cert"@gi' calico.yaml

sed -i 's@.\*etcd\_key:.\*@\ \ etcd\_key:\ "/calico-secrets/etcd-key"@gi' calico.yaml

# 注释掉 calico-node 部分(由 systemd 接管)

sed -i '123,219s@.\*@#&@gi' calico.yaml

2) 部署calico-node及calico-kube-controllers服务。

kubectl create -f calico.yaml

3） 部署完成检查服务状态。确保calico-node和calico-kube-controllers服务状态运行正常.

[root@fp-web-11 ~]# kubectl get pod -n kube-system|egrep calico

calico-kube-controllers-98989846-whzmf 1/1 Running 0 41d

calico-node-lzs79 2/2 Running 0 41d

calico-node-n6ptq 2/2 Running 0 41d

calico-node-tklf6 2/2 Running 0 41d

查看日志详情：

kubectl logs -f calico-kube-controllers-98989846-whzmf -n kube-system