K8s node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。

**一、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. 本节点依赖master和etcd服务的正常运行，需要先行对etcd和master的服务状态进行检查。检查命令如下：

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. 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 |

二、**安装nginx服务。**

将nginx配置文件nginx.conf拷贝至/etc/nginx/下，并修改apiserver地址。

将nginx启动文件nginx-proxy.service拷贝至/usr/lib/systemd/system下。

启动nginx服务: systemctl start nginx-proxy.service

nginx-proxy.service服务启动文件内容如下：

[Unit]

Description=nginx - high performance web server

Documentation=<http://nginx.org/en/docs/>

After=network-online.target remote-fs.target nss-lookup.target

Wants=network-online.target

[Service]

Type=forking

PIDFile=/var/run/nginx.pid

ExecStartPre=/usr/sbin/nginx -t -c /etc/nginx/nginx.conf

ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf

ExecReload=/bin/kill -s HUP $MAINPID

ExecStop=/bin/kill -s TERM $MAINPID

[Install]

WantedBy=multi-user.target

[root@fp-web-15 system]# cat 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:1.13.12-alpine

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

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

Restart=always

Nginx 配置文件内容如下：

cat /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 10.129.51.132:443;

server 10.129.51.117:6443;

}

server {

listen 127.0.0.1:6443;

proxy\_pass kube\_apiserver;

proxy\_timeout 10m;

proxy\_connect\_timeout 1s;

}

}

node节点配置的相关目录（所有节点配置目录需预生成）

|  |  |  |
| --- | --- | --- |
| 目录 | 内容 | 备注 |
| /etc/kubernetes/ | k8s相关配置文件及ca证书 |  |
| /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 |
| /var/lib/kubelet | kubelet默认工作目录 | 存放volume文件（包括emptyDir volume)、plugin文件等，默认挂载在系统盘。 |

node主机远程在节点上执行：

ssh [root@10.129.51.26](mailto: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/" /var/lib/kubelet

**三、创建node节点相关证书及配置文件**

由于node节点上面kube-proxy的证书生成依赖master节点上面的ca证书。所以需要拷贝master上面的ca证书，根据master上面的ca证书生成kube-proxy证书。命令如下：

scp -rp root@10.129.51.132:/etc/kubernetes/ssl/master-ca\*

/etc/kubernetes/ssl/

1. 创建kube-proxy ssl证书

cat kube-proxy-csr.json

{

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

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

生成kube-proxy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 node-proxy-csr.json | cfssljson --bare node-proxy

生成的kubectl证书文件如下：

node-proxy.csr node-proxy-key.pem node-proxy.pem

1. 创建bootstrap.kubeconfig。

KUBE\_APISERVER="<https://127.0.0.1:6443>"

BOOTSTRAP\_TOKEN=$(head -c 16 /dev/urandom | od -An -t x | tr -d ' ')

echo "Tokne: ${BOOTSTRAP\_TOKEN}"

# 设置集群参数

kubectl config set-cluster kubernetes \

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

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=bootstrap.kubeconfig

# 设置客户端认证参数

kubectl config set-credentials kubelet-bootstrap \

--token=${BOOTSTRAP\_TOKEN} \

--kubeconfig=bootstrap.kubeconfig

# 设置上下文参数

kubectl config set-context default \

--cluster=kubernetes \

--user=kubelet-bootstrap \

--kubeconfig=bootstrap.kubeconfig

# 设置默认上下文

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

1. 创建kube-proxy.kubeconfig的配置文件。

KUBE\_APISERVER="<https://127.0.0.1:6443>"

# 设置集群参数,设置集群的访问方式。

kubectl config set-cluster kubernetes \

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

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=kube-proxy.kubeconfig

# 设置客户端认证参数，客户端用户kube-proxy写入文件中。

kubectl config set-credentials kube-proxy \

--client-certificate=node-proxy.pem \

--client-key=node-proxy-key.pem \

--embed-certs=true \

--kubeconfig=kube-proxy.kubeconfig

# 设置上下文参数，上下文为default参数关联cluster和user,使用户能够关联集群访问。

kubectl config set-context default \

--cluster=kubernetes \

--user=kube-proxy \

--kubeconfig=kube-proxy.kubeconfig

# 设置上下文defalut 关联的kubeconfig文件为kube-proxy.kubeconfig

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

四、修改kubelet配置文件。

检查kubelet配置文件/etc/kubernetes/conf/kubelet，修改ip地址KUBELET\_ADDRESS="--node-ip=10.129.51.26"

KUBELET\_ADDRESS="--node-ip=10.129.51.26"

KUBELET\_HOSTNAME="--hostname-override=node1"

KUBELET\_ARGS=" --bootstrap-kubeconfig=/etc/kubernetes/conf/bootstrap.kubeconfig \

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

--cgroup-driver=cgroupfs \

--network-plugin=cni \

--cluster-dns=172.17.0.2 \

--cluster-domain=cluster.local. \

--fail-swap-on=false \

--feature-gates=RotateKubeletClientCertificate=true,RotateKubeletServerCertificate=true \

--node-labels=node-role.kubernetes.io/k8s-node=true \

--image-gc-high-threshold=70 \

--image-gc-low-threshold=50 \

--kube-reserved=cpu=500m,memory=512Mi,ephemeral-storage=1Gi \

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

--system-reserved=cpu=1000m,memory=1024Mi,ephemeral-storage=1Gi \

--serialize-image-pulls=false \

--sync-frequency=30s \

--pod-infra-container-image=k8s.gcr.io/pause-amd64:3.0 \

--resolv-conf=/etc/resolv.conf \

--rotate-certificates"

五、 拷贝证书及相关配置文件

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节点上。

拷贝二进制文件

命令如下：

scp -rp /k8sInstallV11013/masterInstall/software/hyperkube [root@node](mailto:root@10.129.51.132)ip：/usr/local/bin

1.拷贝证书文件

命令如下：

scp -rp /k8sInstallV11013/masterInstall/config/masterConf/ssl/output/\* [root@nodeip](mailto:root@10.129.51.132)：/etc/kubernetes/ssl

2.拷贝基本配置文件

命令如下：

Scp -rp /k8sInstallV11013/masterInstall/config/masterConf/baseconf [root@nodeip](mailto:root@10.129.51.132)：/etc/kubernetes/conf

3.拷贝启动文件

命令如下：

scp -rp /k8sInstallV11013/masterInstall/config/service/\* [root@nodeip](mailto:root@10.129.51.132):/usr/lib/systemd/system/

五、 安装docker服务。

1.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

2.导入docker镜像

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

3.启动服务,依次启动的服务包含nginx、docker、calico、

kubelet、kube-proxy，相关命令如下：

systemctl start nginx.service && systemctl enable nginx.service

systemctl start docker.service && systemctl enable docker.service

systemctl start calico.service && systemctl enable calico.service

systemctl start kubelet.service && systemctl enable kubelet.service

systemctl start kube-proxy.service && systemctl enable kube-proxy.service

4.检查服务运行状态。

最后检查节点上的所有服务运行是否正常。

k8s服务检查状态是否运行。

systemctl list-units -t service -a|egrep "kube|docker|calico"

calico-node.service loaded active running calico node

docker.service loaded active running Docker Application Container Engine

kube-apiserver.service loaded active running Kubernetes API Server

kube-controller-manager.service loaded active running Kubernetes Controller Manager

kube-proxy.service loaded active running Kubernetes Kube-Proxy Server

kube-scheduler.service loaded active running Kubernetes Scheduler Plugin

kubelet.service loaded active running Kubernetes Kubelet Server

1. Master节点查看并审批认证签名csr（ CertificateSigningRequest）请求。

[root@master1 bin]# kubectl get csr

NAME AGE REQUESTOR CONDITION

node-csr--DGwLtY06xigkTakU5TEL2\_2Y\_2kWBiLi0YchZfivgM 31m kubelet-bootstrap Pending

node-csr-r4FISGzsAQ8WG3ZY5GLq9OS\_J162yU9NUslgzYf0BFE 11m kubelet-bootstrap Pending

同意认证签名请求：

[root@master1 bin]# kubectl certificate approve node-csr-r4FISGzsAQ8WG3ZY5GLq9OS\_J162yU9NUslgzYf0BFE

certificatesigningrequest.certificates.k8s.io "node-csr-r4FISGzsAQ8WG3ZY5GLq9OS\_J162yU9NUslgzYf0BFE" approved

最后检查k8s的node状态检查

k8s node状态检查。

[root@master1 ssl]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

master1 Ready k8s-node 46d v1.10.13

master2 Ready k8s-node 46d v1.10.13

node3 Ready k8s-node 46d v1.10.13

六、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

2.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

3.保证k8s集群基础组件和etcd集群运行正常，具体如下：

kubectl get csr

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"}

4.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](mailto: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](mailto: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 |

5.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