再也不踩坑的Kubernetes实战指南

# 安装前必读

请不要使用带中文的服务器和克隆的虚拟机

生产环境建议使用二进制安装方式

请将该文档复制一份，然后进行更改安装，并记录每一个步骤的返回信息，有问题可以直接发送部署文档进行问答，解决更加迅速

# kubeadm高可用安装k8s集群最新版

## 基本环境配置

Kubeadm安装方式自1.14版本以后，安装方法几乎没有任何变化，此文档可以尝试安装最新的k8s集群

K8S官网：<https://kubernetes.io/docs/setup/>

最新版高可用安装：<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/>

表1-1 高可用Kubernetes集群规划

|  |  |  |
| --- | --- | --- |
| 主机名 | IP地址 | 说明 |
| k8s-master01 ~ 03 | 10.103.236.201 ~ 203 | master节点 \* 3 |
| k8s-master-lb | 10.103.236.236 | keepalived虚拟IP |
| k8s-node01 ~ 02 | 10.103.236.204 ~ 205 | worker节点 \* 2 |

VIP（虚拟IP）不要和公司内网IP重复，首先去ping一下，不通才可用。VIP需要和主机在同一个局域网内！

所有节点配置hosts，修改/etc/hosts如下：

[root@k8s-master01 ~]# cat /etc/hosts

10.103.236.201 k8s-master01

10.103.236.202 k8s-master02

10.103.236.203 k8s-master03

10.103.236.236 k8s-master-lb # 如果不是高可用集群，该IP为Master01的IP

10.103.236.204 k8s-node01

10.103.236.205 k8s-node02

CentOS 7安装yum源如下：

curl -o /etc/yum.repos.d/CentOS-Base.repo <https://mirrors.aliyun.com/repo/Centos-7.repo>

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

yum-config-manager --add-repo <https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

sed -i -e '/mirrors.cloud.aliyuncs.com/d' -e '/mirrors.aliyuncs.com/d' /etc/yum.repos.d/CentOS-Base.repo

必备工具安装

yum install wget jq psmisc vim net-tools telnet yum-utils device-mapper-persistent-data lvm2 git -y

所有节点关闭防火墙、selinux、dnsmasq、swap。服务器配置如下：

systemctl disable --now firewalld

systemctl disable --now dnsmasq

systemctl disable --now NetworkManager

setenforce 0

sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/sysconfig/selinux

关闭swap分区

swapoff -a && sysctl -w vm.swappiness=0

sed -ri '/^[^#]\*swap/s@^@#@' /etc/fstab

安装ntpdate

rpm -ivh http://mirrors.wlnmp.com/centos/wlnmp-release-centos.noarch.rpm

yum install ntpdate -y

所有节点同步时间。时间同步配置如下：

ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

echo 'Asia/Shanghai' >/etc/timezone

ntpdate time2.aliyun.com

# 加入到crontab

\*/5 \* \* \* \* ntpdate time2.aliyun.com

所有节点配置limit：

ulimit -SHn 65535

vim /etc/security/limits.conf

# 末尾添加如下内容

\* soft nofile 655360

\* hard nofile 131072

\* soft nproc 655350

\* hard nproc 655350

\* soft memlock unlimited

\* hard memlock unlimited

Master01节点免密钥登录其他节点，安装过程中生成配置文件和证书均在Master01上操作，集群管理也在Master01上操作，阿里云或者AWS上需要单独一台kubectl服务器。密钥配置如下：

ssh-keygen -t rsa

for i in k8s-master01 k8s-master02 k8s-master03 k8s-node01 k8s-node02;do ssh-copy-id -i .ssh/id\_rsa.pub $i;done

下载安装所有的源码文件：

cd /root/ ; git clone <https://github.com/dotbalo/k8s-ha-install.git>

所有节点升级系统并重启，此处升级没有升级内核，下节会单独升级内核：

yum update -y --exclude=kernel\* && reboot #CentOS7需要升级，CentOS8可以按需升级系统

## 内核配置

CentOS7 需要升级内核至4.18+，本地升级的版本为4.19

在master01节点下载内核：(购买架构师课程的可以从百度网盘下载)

cd /root

wget http://193.49.22.109/elrepo/kernel/el7/x86\_64/RPMS/kernel-ml-devel-4.19.12-1.el7.elrepo.x86\_64.rpm

wget http://193.49.22.109/elrepo/kernel/el7/x86\_64/RPMS/kernel-ml-4.19.12-1.el7.elrepo.x86\_64.rpm

从master01节点传到其他节点：

for i in k8s-master02 k8s-master03 k8s-node01 k8s-node02;do scp kernel-ml-4.19.12-1.el7.elrepo.x86\_64.rpm kernel-ml-devel-4.19.12-1.el7.elrepo.x86\_64.rpm $i:/root/ ; done

所有节点安装内核

cd /root && yum localinstall -y kernel-ml\*

所有节点更改内核启动顺序

grub2-set-default 0 && grub2-mkconfig -o /etc/grub2.cfg

grubby --args="user\_namespace.enable=1" --update-kernel="$(grubby --default-kernel)"

检查默认内核是不是4.19

[root@k8s-master02 ~]# grubby --default-kernel

/boot/vmlinuz-4.19.12-1.el7.elrepo.x86\_64

所有节点重启，然后检查内核是不是4.19

[root@k8s-master02 ~]# uname -a

Linux k8s-master02 4.19.12-1.el7.elrepo.x86\_64 #1 SMP Fri Dec 21 11:06:36 EST 2018 x86\_64 x86\_64 x86\_64 GNU/Linux

所有节点安装ipvsadm：

yum install ipvsadm ipset sysstat conntrack libseccomp -y

所有节点配置ipvs模块，在内核4.19+版本nf\_conntrack\_ipv4已经改为nf\_conntrack， 4.18以下使用nf\_conntrack\_ipv4即可：

modprobe -- ip\_vs

modprobe -- ip\_vs\_rr

modprobe -- ip\_vs\_wrr

modprobe -- ip\_vs\_sh

modprobe -- nf\_conntrack

vim /etc/modules-load.d/ipvs.conf

# 加入以下内容

ip\_vs

ip\_vs\_lc

ip\_vs\_wlc

ip\_vs\_rr

ip\_vs\_wrr

ip\_vs\_lblc

ip\_vs\_lblcr

ip\_vs\_dh

ip\_vs\_sh

ip\_vs\_fo

ip\_vs\_nq

ip\_vs\_sed

ip\_vs\_ftp

ip\_vs\_sh

nf\_conntrack

ip\_tables

ip\_set

xt\_set

ipt\_set

ipt\_rpfilter

ipt\_REJECT

ipip

然后执行systemctl enable --now systemd-modules-load.service即可

检查是否加载：

[root@k8s-master01 ~]# lsmod | grep -e ip\_vs -e nf\_conntrack\_ipv4

nf\_conntrack\_ipv4 16384 23

nf\_defrag\_ipv4 16384 1 nf\_conntrack\_ipv4

nf\_conntrack 135168 10 xt\_conntrack,nf\_conntrack\_ipv6,nf\_conntrack\_ipv4,nf\_nat,nf\_nat\_ipv6,ipt\_MASQUERADE,nf\_nat\_ipv4,xt\_nat,nf\_conntrack\_netlink,ip\_vs

开启一些k8s集群中必须的内核参数，所有节点配置k8s内核：

cat <<EOF > /etc/sysctl.d/k8s.conf

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

fs.may\_detach\_mounts = 1

vm.overcommit\_memory=1

vm.panic\_on\_oom=0

fs.inotify.max\_user\_watches=89100

fs.file-max=52706963

fs.nr\_open=52706963

net.netfilter.nf\_conntrack\_max=2310720

net.ipv4.tcp\_keepalive\_time = 600

net.ipv4.tcp\_keepalive\_probes = 3

net.ipv4.tcp\_keepalive\_intvl =15

net.ipv4.tcp\_max\_tw\_buckets = 36000

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_max\_orphans = 327680

net.ipv4.tcp\_orphan\_retries = 3

net.ipv4.tcp\_syncookies = 1

net.ipv4.tcp\_max\_syn\_backlog = 16384

net.ipv4.ip\_conntrack\_max = 65536

net.ipv4.tcp\_max\_syn\_backlog = 16384

net.ipv4.tcp\_timestamps = 0

net.core.somaxconn = 16384

EOF

sysctl --system

所有节点配置完内核后，重启服务器，保证重启后内核依旧加载

reboot

lsmod | grep --color=auto -e ip\_vs -e nf\_conntrack

## 基本组件安装

本节主要安装的是集群中用到的各种组件，比如Docker-ce、Kubernetes各组件等。

所有节点安装Docker-ce 19.03

yum install docker-ce-19.03.\* -y

温馨提示：

由于新版kubelet建议使用systemd，所以可以把docker的CgroupDriver改成systemd

mkdir /etc/docker

cat > /etc/docker/daemon.json <<EOF

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

EOF

所有节点设置开机自启动Docker：

systemctl daemon-reload && systemctl enable --now docker

安装k8s组件：

yum list kubeadm.x86\_64 --showduplicates | sort -r

所有节点安装最新版本kubeadm：

yum install kubeadm -y

默认配置的pause镜像使用gcr.io仓库，国内可能无法访问，所以这里配置Kubelet使用阿里云的pause镜像：

cat >/etc/sysconfig/kubelet<<EOF

KUBELET\_EXTRA\_ARGS="--cgroup-driver=systemd --pod-infra-container-image=registry.cn-hangzhou.aliyuncs.com/google\_containers/pause-amd64:3.2"

EOF

设置Kubelet开机自启动：

systemctl daemon-reload

systemctl enable --now kubelet

## 高可用组件安装

（注意：如果不是高可用集群，haproxy和keepalived无需安装）

所有Master节点通过yum安装HAProxy和KeepAlived：

yum install keepalived haproxy -y

所有Master节点配置HAProxy（详细配置参考HAProxy文档，所有Master节点的HAProxy配置相同）：

[root@k8s-master01 etc]# mkdir /etc/haproxy

[root@k8s-master01 etc]# vim /etc/haproxy/haproxy.cfg

global

maxconn 2000

ulimit-n 16384

log 127.0.0.1 local0 err

stats timeout 30s

defaults

log global

mode http

option httplog

timeout connect 5000

timeout client 50000

timeout server 50000

timeout http-request 15s

timeout http-keep-alive 15s

frontend monitor-in

bind \*:33305

mode http

option httplog

monitor-uri /monitor

frontend k8s-master

bind 0.0.0.0:16443

bind 127.0.0.1:16443

mode tcp

option tcplog

tcp-request inspect-delay 5s

default\_backend k8s-master

backend k8s-master

mode tcp

option tcplog

option tcp-check

balance roundrobin

default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250 maxqueue 256 weight 100

server k8s-master01 10.103.236.201:6443 check

server k8s-master02 10.103.236.202:6443 check

server k8s-master03 10.103.236.203:6443 check

　所有Master节点配置KeepAlived，配置不一样，注意区分 [root@k8s-master01 pki]# vim /etc/keepalived/keepalived.conf ，注意每个节点的IP和网卡（interface参数）

Master01节点的配置：

[root@k8s-master01 etc]# mkdir /etc/keepalived

[root@k8s-master01 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

script\_user root

enable\_script\_security

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 3

weight -5

fall 2

}

vrrp\_instance VI\_1 {

state MASTER

interface ens192

mcast\_src\_ip 10.103.236.201

virtual\_router\_id 51

priority 101

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

10.103.236.236

}

# track\_script {

# chk\_apiserver

# }

}

Master02节点的配置：

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

script\_user root

enable\_script\_security

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 3

weight -5

fall 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens192

mcast\_src\_ip 10.103.236.202

virtual\_router\_id 51

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

10.103.236.236

}

# track\_script {

# chk\_apiserver

# }

}

Master03节点的配置：

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

script\_user root

enable\_script\_security

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 3

weight -5

fall 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens192

mcast\_src\_ip 10.103.236.203

virtual\_router\_id 51

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

10.103.236.236

}

# track\_script {

# chk\_apiserver

# }

}

注意上述的健康检查是关闭的，集群建立完成后再开启：

# track\_script {

# chk\_apiserver

# }

配置KeepAlived健康检查文件：

[root@k8s-master01 keepalived]# cat /etc/keepalived/check\_apiserver.sh

#!/bin/bash

err=0

for k in $(seq 1 2)

do

check\_code=$(curl -k -s https://127.0.0.1:6443/healthz --connect-timeout 1)

if [[ $check\_code != "ok" ]]; then

err=$(expr $err + 1)

sleep 1

continue

else

err=0

break

fi

done

if [[ $err != "0" ]]; then

echo "systemctl stop keepalived"

/usr/bin/systemctl stop keepalived

exit 1

else

exit 0

fi

chmod +x /etc/keepalived/check\_apiserver.sh

启动haproxy和keepalived

[root@k8s-master01 keepalived]# systemctl daemon-reload

[root@k8s-master01 keepalived]# systemctl enable --now haproxy

[root@k8s-master01 keepalived]# systemctl enable --now keepalived

重要：如果安装了keepalived和haproxy，需要测试keepalived是否是正常的

测试VIP

[root@k8s-master01 ~]# ping 10.103.236.236 -c 4

PING 10.103.236.236 (10.103.236.236) 56(84) bytes of data.

64 bytes from 10.103.236.236: icmp\_seq=1 ttl=64 time=0.464 ms

64 bytes from 10.103.236.236: icmp\_seq=2 ttl=64 time=0.063 ms

64 bytes from 10.103.236.236: icmp\_seq=3 ttl=64 time=0.062 ms

64 bytes from 10.103.236.236: icmp\_seq=4 ttl=64 time=0.063 ms

--- 10.103.236.236 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3106ms

rtt min/avg/max/mdev = 0.062/0.163/0.464/0.173 ms

[root@k8s-master01 ~]# telnet 10.103.236.236 16443

Trying 10.103.236.236...

Connected to 10.103.236.236.

Escape character is '^]'.

Connection closed by foreign host.

如果ping不通且telnet没有出现 ] ，则认为VIP不可以，不可在继续往下执行，需要排查keepalived的问题，比如防火墙和selinux，haproxy和keepalived的状态，监听端口等

所有节点查看防火墙状态必须为disable和inactive：systemctl status firewalld

所有节点查看selinux状态，必须为disable：getenforce

master节点查看haproxy和keepalived状态：systemctl status keepalived haproxy

master节点查看监听端口：netstat -lntp

## 集群初始化

官方初始化文档：

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/>

Master01节点创建kubeadm-config.yaml配置文件如下：

Master01：（# 注意，如果不是高可用集群，10.103.236.236:16443改为master01的地址，16443改为apiserver的端口，默认是6443，注意更改v1.18.5自己服务器kubeadm的版本：kubeadm version）

apiVersion: kubeadm.k8s.io/v1beta2

bootstrapTokens:

- groups:

- system:bootstrappers:kubeadm:default-node-token

token: 7t2weq.bjbawausm0jaxury

ttl: 24h0m0s

usages:

- signing

- authentication

kind: InitConfiguration

localAPIEndpoint:

advertiseAddress: 10.103.236.201

bindPort: 6443

nodeRegistration:

criSocket: /var/run/dockershim.sock

name: k8s-master01

taints:

- effect: NoSchedule

key: node-role.kubernetes.io/master

---

apiServer:

certSANs:

- 10.103.236.236

timeoutForControlPlane: 4m0s

apiVersion: kubeadm.k8s.io/v1beta2

certificatesDir: /etc/kubernetes/pki

clusterName: kubernetes

controlPlaneEndpoint: 10.103.236.236:16443

controllerManager: {}

dns:

type: CoreDNS

etcd:

local:

dataDir: /var/lib/etcd

imageRepository: registry.cn-hangzhou.aliyuncs.com/google\_containers

kind: ClusterConfiguration

kubernetesVersion: v1.20.0

networking:

dnsDomain: cluster.local

podSubnet: 172.168.0.0/16

serviceSubnet: 10.96.0.0/12

scheduler: {}

更新kubeadm文件

kubeadm config migrate --old-config kubeadm-config.yaml --new-config new.yaml

将new.yaml文件复制到其他master节点，之后所有Master节点提前下载镜像，可以节省初始化时间：

kubeadm config images pull --config /root/new.yaml

所有节点设置开机自启动kubelet

systemctl enable --now kubelet（如果启动失败无需管理，初始化成功以后即可启动）

Master01节点初始化，初始化以后会在/etc/kubernetes目录下生成对应的证书和配置文件，之后其他Master节点加入Master01即可：

kubeadm init --config /root/new.yaml --upload-certs

如果初始化失败，重置后再次初始化，命令如下：

kubeadm reset -f ; ipvsadm --clear ; rm -rf ~/.kube

初始化成功以后，会产生Token值，用于其他节点加入时使用，因此要记录下初始化成功生成的token值（令牌值）：

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of the control-plane node running the following command on each as root:

kubeadm join 10.103.236.236:16443 --token 7t2weq.bjbawausm0jaxury \

--discovery-token-ca-cert-hash sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908 \

--control-plane --certificate-key ac2854de93aaabdf6dc440322d4846fc230b290c818c32d6ea2e500fc930b0aa

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!

As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use

"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.103.236.236:16443 --token 7t2weq.bjbawausm0jaxury \

--discovery-token-ca-cert-hash sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908

Master01节点配置环境变量，用于访问Kubernetes集群：

cat <<EOF >> /root/.bashrc

export KUBECONFIG=/etc/kubernetes/admin.conf

EOF

source /root/.bashrc

查看节点状态：

[root@k8s-master01 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master01 NotReady control-plane,master 74s v1.20.0

采用初始化安装方式，所有的系统组件均以容器的方式运行并且在kube-system命名空间内，此时可以查看Pod状态：

[root@k8s-master01 ~]# kubectl get pods -n kube-system -o wide

NAME READY STATUS RESTARTS AGE IP NODE

coredns-777d78ff6f-kstsz 0/1 Pending 0 14m <none> <none>

coredns-777d78ff6f-rlfr5 0/1 Pending 0 14m <none> <none>

etcd-k8s-master01 1/1 Running 0 14m 10.103.236.201 k8s-master01

kube-apiserver-k8s-master01 1/1 Running 0 13m 10.103.236.201 k8s-master01

kube-controller-manager-k8s-master01 1/1 Running 0 13m 10.103.236.201 k8s-master01

kube-proxy-8d4qc 1/1 Running 0 14m 10.103.236.201 k8s-master01

kube-scheduler-k8s-master01 1/1 Running 0 13m 10.103.236.201 k8s-master01

## 高可用Master

Token过期后生成新的token：

kubeadm token create --print-join-command

Master需要生成--certificate-key

kubeadm init phase upload-certs --upload-certs

初始化其他master加入集群

kubeadm join 10.103.236.236:16443 --token 7t2weq.bjbawausm0jaxury \

--discovery-token-ca-cert-hash sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908 \

--control-plane --certificate-key ac2854de93aaabdf6dc440322d4846fc230b290c818c32d6ea2e500fc930b0aa

## Node节点的配置

Node节点上主要部署公司的一些业务应用，生产环境中不建议Master节点部署系统组件之外的其他Pod，测试环境可以允许Master节点部署Pod以节省系统资源。

kubeadm join 10.103.236.236:16443 --token 7t2weq.bjbawausm0jaxury \

--discovery-token-ca-cert-hash sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908

所有节点初始化完成后，查看集群状态

[root@k8s-master01]# kubectl get node

NAME STATUS ROLES AGE VERSION

k8s-master01 NotReady control-plane,master 8m53s v1.20.0

k8s-master02 NotReady control-plane,master 2m25s v1.20.0

k8s-master03 NotReady control-plane,master 31s v1.20.0

k8s-node01 NotReady <none> 32s v1.20.0

k8s-node02 NotReady <none> 88s v1.20.0

## Calico组件的安装

注意：如果国内用户下载Calico较慢，所有节点可以配置加速器(如果该文件有其他配置，别忘了加上去)

以下步骤只在master01执行

cd /root/k8s-ha-install && git checkout manual-installation-v1.20.x && cd calico/

修改calico-etcd.yaml的以下位置

sed -i 's#etcd\_endpoints: "http://<ETCD\_IP>:<ETCD\_PORT>"#etcd\_endpoints: "https://10.103.236.201:2379,https://10.103.236.202:2379,https://10.103.236.203:2379"#g' calico-etcd.yaml

ETCD\_CA=`cat /etc/kubernetes/pki/etcd/ca.crt | base64 | tr -d '\n'`

ETCD\_CERT=`cat /etc/kubernetes/pki/etcd/server.crt | base64 | tr -d '\n'`

ETCD\_KEY=`cat /etc/kubernetes/pki/etcd/server.key | base64 | tr -d '\n'`

sed -i "s@# etcd-key: null@etcd-key: ${ETCD\_KEY}@g; s@# etcd-cert: null@etcd-cert: ${ETCD\_CERT}@g; s@# etcd-ca: null@etcd-ca: ${ETCD\_CA}@g" calico-etcd.yaml

sed -i 's#etcd\_ca: ""#etcd\_ca: "/calico-secrets/etcd-ca"#g; s#etcd\_cert: ""#etcd\_cert: "/calico-secrets/etcd-cert"#g; s#etcd\_key: "" #etcd\_key: "/calico-secrets/etcd-key" #g' calico-etcd.yaml

POD\_SUBNET=`cat /etc/kubernetes/manifests/kube-controller-manager.yaml | grep cluster-cidr= | awk -F= '{print $NF}'`

sed -i 's@# - name: CALICO\_IPV4POOL\_CIDR@- name: CALICO\_IPV4POOL\_CIDR@g; s@# value: "192.168.0.0/16"@ value: '"${POD\_SUBNET}"'@g' calico-etcd.yaml

kubectl apply -f calico-etcd.yaml

查看容器状态

[root@k8s-master01 calico]# kubectl get po -n kube-system

NAME READY STATUS RESTARTS AGE

calico-kube-controllers-5f6d4b864b-pwvnb 1/1 Running 0 3m29s

calico-node-5lz9m 1/1 Running 0 3m29s

calico-node-8z4bg 1/1 Running 0 3m29s

calico-node-lmzvf 1/1 Running 0 3m29s

calico-node-mpngv 1/1 Running 0 3m29s

calico-node-vmqsl 1/1 Running 0 3m29s

coredns-54d67798b7-8525g 1/1 Running 0 39m

coredns-54d67798b7-fxs72 1/1 Running 0 39m

etcd-k8s-master01 1/1 Running 0 39m

etcd-k8s-master02 1/1 Running 0 33m

etcd-k8s-master03 1/1 Running 0 31m

kube-apiserver-k8s-master01 1/1 Running 0 39m

kube-apiserver-k8s-master02 1/1 Running 0 33m

kube-apiserver-k8s-master03 1/1 Running 0 30m

kube-controller-manager-k8s-master01 1/1 Running 1 39m

kube-controller-manager-k8s-master02 1/1 Running 0 33m

kube-controller-manager-k8s-master03 1/1 Running 0 31m

kube-proxy-hnkmj 1/1 Running 0 39m

kube-proxy-jk4dm 1/1 Running 0 32m

kube-proxy-nbcg2 1/1 Running 0 32m

kube-proxy-qv9k7 1/1 Running 0 32m

kube-proxy-x6xdc 1/1 Running 0 33m

kube-scheduler-k8s-master01 1/1 Running 1 39m

kube-scheduler-k8s-master02 1/1 Running 0 33m

kube-scheduler-k8s-master03 1/1 Running 0 30m

## Metrics部署

在新版的Kubernetes中系统资源的采集均使用Metrics-server，可以通过Metrics采集节点和Pod的内存、磁盘、CPU和网络的使用率。

将Master01节点的front-proxy-ca.crt复制到所有Node节点

scp /etc/kubernetes/pki/front-proxy-ca.crt k8s-node01:/etc/kubernetes/pki/front-proxy-ca.crt

scp /etc/kubernetes/pki/front-proxy-ca.crt k8s-node(其他节点自行拷贝):/etc/kubernetes/pki/front-proxy-ca.crt

安装metrics server

cd /root/k8s-ha-install/metrics-server-0.4.x-kubeadm/

[root@k8s-master01 metrics-server-0.4.x-kubeadm]# kubectl create -f comp.yaml

serviceaccount/metrics-server created

clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created

clusterrole.rbac.authorization.k8s.io/system:metrics-server created

rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created

clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created

clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created

service/metrics-server created

deployment.apps/metrics-server created

apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created

查看状态

[root@k8s-master01 metrics-server-0.4.x-kubeadm]# kubectl top node

NAME CPU(cores) CPU% MEMORY(bytes) MEMORY%

k8s-master01 109m 2% 1296Mi 33%

k8s-master02 99m 2% 1124Mi 29%

k8s-master03 104m 2% 1082Mi 28%

k8s-node01 55m 1% 761Mi 19%

k8s-node02 53m 1% 663Mi 17%

## Dashboard部署

Dashboard用于展示集群中的各类资源，同时也可以通过Dashboard实时查看Pod的日志和在容器中执行一些命令等。

### 安装指定版本dashboard

cd /root/k8s-ha-install/dashboard/

[root@k8s-master01 dashboard]# kubectl create -f .

serviceaccount/admin-user created

clusterrolebinding.rbac.authorization.k8s.io/admin-user created

namespace/kubernetes-dashboard created

serviceaccount/kubernetes-dashboard created

service/kubernetes-dashboard created

secret/kubernetes-dashboard-certs created

secret/kubernetes-dashboard-csrf created

secret/kubernetes-dashboard-key-holder created

configmap/kubernetes-dashboard-settings created

role.rbac.authorization.k8s.io/kubernetes-dashboard created

clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created

rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created

clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created

deployment.apps/kubernetes-dashboard created

service/dashboard-metrics-scraper created

deployment.apps/dashboard-metrics-scraper created

### 安装最新版

官方GitHub地址：<https://github.com/kubernetes/dashboard>

可以在官方dashboard查看到最新版dashboard



kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml>

创建管理员用户vim admin.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

annotations:

rbac.authorization.kubernetes.io/autoupdate: "true"

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kube-system

kubectl apply -f admin.yaml -n kube-system

### 登录dashboard

在谷歌浏览器（Chrome）启动文件中加入启动参数，用于解决无法访问Dashboard的问题，参考图1-1：

--test-type --ignore-certificate-errors



图1-1 谷歌浏览器 Chrome的配置

更改dashboard的svc为NodePort：

kubectl edit svc kubernetes-dashboard -n kubernetes-dashboard



将ClusterIP更改为NodePort（如果已经为NodePort忽略此步骤）：

查看端口号：



根据自己的实例端口号，通过任意安装了kube-proxy的宿主机或者VIP的IP+端口即可访问到dashboard：

访问Dashboard：[https://10.103.236.236:18282（请更改18282为自己的端口）](https://192.168.0.200:18282（请更改18282为自己的端口）)，选择登录方式为令牌（即token方式），参考图1-2



图1-2 Dashboard登录方式

查看token值：

[root@k8s-master01 1.1.1]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

Name: admin-user-token-r4vcp

Namespace: kube-system

Labels: <none>

Annotations: kubernetes.io/service-account.name: admin-user

kubernetes.io/service-account.uid: 2112796c-1c9e-11e9-91ab-000c298bf023

Type: kubernetes.io/service-account-token

Data

====

ca.crt: 1025 bytes

namespace: 11 bytes

token: **eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXI0dmNwIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiIyMTEyNzk2Yy0xYzllLTExZTktOTFhYi0wMDBjMjk4YmYwMjMiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.bWYmwgRb-90ydQmyjkbjJjFt8CdO8u6zxVZh-19rdlL\_T-n35nKyQIN7hCtNAt46u6gfJ5XXefC9HsGNBHtvo\_Ve6oF7EXhU772aLAbXWkU1xOwQTQynixaypbRIas\_kiO2MHHxXfeeL\_yYZRrgtatsDBxcBRg-nUQv4TahzaGSyK42E\_4YGpLa3X3Jc4t1z0SQXge7lrwlj8ysmqgO4ndlFjwPfvg0eoYqu9Qsc5Q7tazzFf9mVKMmcS1ppPutdyqNYWL62P1prw\_wclP0TezW1CsypjWSVT4AuJU8YmH8nTNR1EXn8mJURLSjINv6YbZpnhBIPgUGk1JYVLcn47w**

将token值输入到令牌后，单击登录即可访问Dashboard，参考图1-3：



图1-3 Dashboard页面

## 一些必须的配置更改

将Kube-proxy改为ipvs模式，因为在初始化集群的时候注释了ipvs配置，所以需要自行修改一下：

在master01节点执行

kubectl edit cm kube-proxy -n kube-system

mode: “ipvs”

更新Kube-Proxy的Pod：

kubectl patch daemonset kube-proxy -p "{\"spec\":{\"template\":{\"metadata\":{\"annotations\":{\"date\":\"`date +'%s'`\"}}}}}" -n kube-system

验证Kube-Proxy模式

[root@k8s-master01 1.1.1]# curl 127.0.0.1:10249/proxyMode

ipvs

在所有的master节点

开启keepalived的健康检查：

[root@k8s-master01 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 3

weight -5

fall 2

}

vrrp\_instance VI\_1 {

state MASTER

interface ens192

mcast\_src\_ip 10.103.236.201

virtual\_router\_id 51

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

10.103.236.236

}

# 将此块注释打开

track\_script {

chk\_apiserver

}

}

重启keepalived

systemctl daemon-reload

systemctl restart keepalived

# 注意事项

注意：kubeadm安装的集群，证书有效期默认是一年。master节点的kube-apiserver、kube-scheduler、kube-controller-manager、etcd都是以容器运行的。可以通过kubectl get po -n kube-system查看。

启动和二进制不同的是，

kubelet的配置文件在/etc/sysconfig/kubelet和/var/lib/kubelet/config.yaml

其他组件的配置文件在/etc/Kubernetes/manifests目录下，比如kube-apiserver.yaml，该yaml文件更改后，kubelet会自动刷新配置，也就是会重启pod。不能再次创建该文件

Kubeadm安装后，master节点默认不允许部署pod，可以通过以下方式打开：

查看Taints：

[root@k8s-master01 ~]# kubectl describe node -l node-role.kubernetes.io/master= | grep Taints

Taints: node-role.kubernetes.io/master:NoSchedule

Taints: node-role.kubernetes.io/master:NoSchedule

Taints: node-role.kubernetes.io/master:NoSchedule

删除Taint：

[root@k8s-master01 ~]# kubectl taint node -l node-role.kubernetes.io/master node-role.kubernetes.io/master:NoSchedule-

node/k8s-master01 untainted

node/k8s-master02 untainted

node/k8s-master03 untainted

[root@k8s-master01 ~]# kubectl describe node -l node-role.kubernetes.io/master= | grep Taints

Taints: <none>

Taints: <none>

Taints: <none>