**单master2个node节点集群架构部署演示**

**一、准备实验环境**

1.准备三台centos7虚拟机，用来安装k8s集群，下面是三台虚拟机的配置情况

**master1192.168.1.10）配置：**

操作系统：centos7.4、centos7.5、centos7.6以及更高版本都可以

配置：4核cpu，6G内存，两块60G硬盘

网络：桥接网络

**Node1（192.168.1.11）配置：**

操作系统：centos7.4、centos7.5、centos7.6以及更高版本都可以

配置：4核cpu，6G内存，两块60G硬盘

网络：桥接网络

**Node2（192.168.1.12）配置：**

操作系统：centos7.4、centos7.5、centos7.6以及更高版本都可以

配置：4核cpu，6G内存，两块60G硬盘

网络：桥接网络

**二、初始化实验环境**

#### **设置主机名**

hostnamectl set-hostname k8s-master

#### 1、让每台机子可以相互解析 在添加

vi /etc/hosts

192.168.1.10 k8s-master

192.168.1.11 k8s-node1

192.168.1.12 k8s-node2

**2.安装基础软件包，各个节点操作**

yum install -y conntrack ntpdate ntp ipvsadm ipset jq iptables curl sysstat libseccomp wget vim net-tools git ntpdate

**3.关闭firewalld防火墙，各个节点操作，centos7系统默认使用的是firewalld防火墙，停止firewalld防火墙，并禁用这个服务**

systemctl stop firewalld && systemctl disable firewalld

swapoff -a && sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

setenforce 0 && sed -i 's/^SELINUX=.\*/SELINUX=disabled/' /etc/selinux/config

**5.时间同步，各个节点操作**

5.1 时间同步

ntpdate cn.pool.ntp.org

5.2 编辑计划任务，每小时做一次同步

crontab -e

\* \*/1 \* \* \* /usr/sbin/ntpdate cn.pool.ntp.org

1. **修改内核参数，各个节点操作**

#### **调整内核参数，对于K8S**

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

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

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

net.ipv4.ip\_forward=1

net.ipv4.ip\_nonlocal\_bind = 1

net.ipv4.tcp\_tw\_recycle=0

vm.swappiness=0 # 禁止使用 swap 空间，只有当系统 OOM 时才允许使用它

vm.overcommit\_memory=1 # 不检查物理内存是否够用

vm.panic\_on\_oom=0 # 开启 OOM

fs.inotify.max\_user\_instances=8192

fs.inotify.max\_user\_watches=1048576

fs.file-max=52706963

fs.nr\_open=52706963

net.ipv6.conf.all.disable\_ipv6=1

EOF

sysctl -p /etc/sysctl.d/kubernetes.conf

#### **升级系统内核为 4.44（可选）**

# CentOS 7.x 系统自带的 3.10.x 内核存在一些 Bugs，导致运行的 Docker、Kubernetes 不稳定，查看内核命令uname -r，升级步骤如下：

rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm#安装完成后检查 /boot/grub2/grub.cfg 中对应内核 menuentry 中是否包含 initrd16 配置，如果没有，再安装一次!

yum --enablerepo=elrepo-kernel install -y kernel-lt#设置开机从新内核启动

grub2-set-default "CentOS Linux (4.4.182-1.el7.elrepo.x86\_64) 7 (Core)"# 重启

reboot# 查看内核变化啦

**三、安装kubernetes1.18高可用集群**

#### **1、 安装docker**

#如果未按照wget命令

yum install wget

wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -O /etc/yum.repos.d/docker-ce.repo

yum -y install docker-ce-18.06.1.ce-3.el7

systemctl enable docker && systemctl start docker #启动并设置开机启动

docker --version #查看是否安装成功

# 创建/etc/docker目录

配置docker阿里云镜像加速

sudo mkdir -p /etc/docker

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

{

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

"registry-mirrors": ["https://yywkvob3.mirror.aliyuncs.com"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

}

}

EOF

#重启docker

systemctl daemon-reload

systemctl restart docker

systemctl enable docker

#### **2、 安装Kubeadm、 kubectl和 kubelet**

# 配置K8S的yum源

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

[kubernetes]

name=Kubernetes

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

enabled=1

gpgcheck=0

repo\_gpgcheck=0

gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg

http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

# 安装kubeadm, kubectl, and kubelet.

sudo yum install -y kubelet-1.18.0 kubeadm-1.18.0 kubectl-1.18.0

sudo systemctl restart kubelet

sudo systemctl enable kubelet

#### **设置集群主节点配置--master**

[root@master keepalived]# cat kubeadm-config.yaml

apiVersion: kubeadm.k8s.io/v1beta2

kind: ClusterConfiguration

kubernetesVersion: v1.18.0

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

controlPlaneEndpoint: "vip:6443"

networking:

serviceSubnet: "10.96.0.0/16"

podSubnet: "10.100.0.1/16"

dnsDomain: "cluster.local"

如下命令进行初始化：

kubeadm init --config=kubeadm-config.yaml --upload-certs

* 拷贝一下这里打印出来的两条kubeadm join命令，后面添加其他master节点以及worker节点时需要用到

#配置环境变量

mkdir -p $HOME/.kube

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

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

#### **安装calico网络 --master节点**

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

#### **5、查看节点情况**

kubectl get node -o wide

#### 6、子节点加入

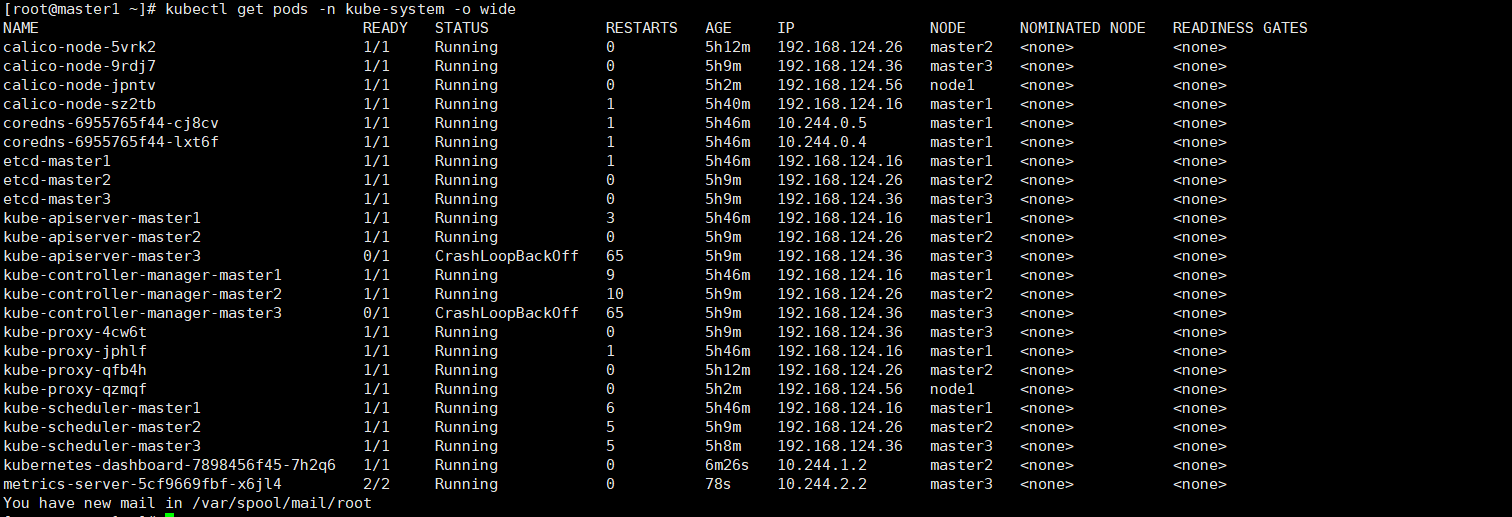
kubeadm join 192.168.183.10:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:f2a9b77cbe8bdaa9dd8cdcd414fb4a9d09059dfb33ced7f13fc2218f425b6273

注意token是master节点初始化时，生产的。如果忘记了。执行以下操作

kubeadm token create --print-join-command

上面组件都安装之后，kubectl get pods -n kube-system -o wide，查看组件安装是否正常，STATUS状态是Running，说明组件正常，如下所示



# **多master多node集群搭建演示**

[root@master keepalived]# cat kubeadm-config.yaml

apiVersion: kubeadm.k8s.io/v1beta2

kind: ClusterConfiguration

kubernetesVersion: v1.18.0

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

controlPlaneEndpoint: "master:6443"

networking:

serviceSubnet: "10.96.0.0/16"

podSubnet: "10.100.0.1/16"

dnsDomain: "cluster.local"

如下命令进行初始化：

kubeadm init --config=kubeadm-config.yaml --upload-certs

* 拷贝一下这里打印出来的两条kubeadm join命令，后面添加其他master节点以及worker节点时需要用到

#配置环境变量

mkdir -p $HOME/.kube

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

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

#### **安装calico网络 --master节点**

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

#### **查看pod信息状态**

kubectl get pod --all-namespaces

# **第二台master加入集群**

**前提第二台master 环境是正常的，docker 和kubelet 安装正常**

**就可以执行加入集群**

**Node在加入集群：**

**加入集群命令刚才都保存好了 执行就好**

# **基于kubeadm搭建k8s高可用集群**

### **部署keepalived - apiserver高可用（任选两个master节点）**

vim /etc/keepalived/keepalived.conf

global\_defs {

router\_id keepalive-master

}

vrrp\_script check\_apiserver {

# 检测脚本路径

script "/etc/keepalived/check-apiserver.sh"

# 多少秒检测一次

interval 3

# 失败的话权重-2

weight -2

}

vrrp\_instance VI-kube-master {

state MASTER # 定义节点角色

interface ens33 # 网卡名称

virtual\_router\_id 68

priority 100

dont\_track\_primary

advert\_int 3

virtual\_ipaddress {

# 自定义虚拟ip

192.168.5.199

}

track\_script {

check\_apiserver

}

}

在m2（角色为backup）上创建配置文件如下：

global\_defs {

router\_id keepalive-master

}

vrrp\_script check\_apiserver {

# 检测脚本路径

script "/etc/keepalived/check-apiserver.sh"

# 多少秒检测一次

interval 3

# 失败的话权重-2

weight -2

}

vrrp\_instance VI-kube-master {

state MASTER # 定义节点角色

interface ens33 # 网卡名称

virtual\_router\_id 68

priority 99

dont\_track\_primary

advert\_int 3

virtual\_ipaddress {

# 自定义虚拟ip

192.168.5.199

}

track\_script {

check\_apiserver

}

}

vim /etc/keepalived/check-apiserver.sh

#!/bin/sh

netstat -ntlp |grep 6443 || exit 1

分别在master和backup上启动keepalived服务

$ systemctl enable keepalived && service keepalived start

### **部署第一个k8s主节点**

kubeadm config images list

使用kubeadm创建的k8s集群，大部分组件都是以docker容器的方式去运行的，所以kubeadm在初始化master节点的时候需要拉取

脚本执行完后，此时查看Docker镜像列表应如下：

创建kubeadm用于初始化master节点的配置文件：

cat <<EOF > ./kubeadm-config.yaml

apiVersion: kubeadm.k8s.io/v1beta2

kind: ClusterConfiguration

kubernetesVersion: v1.18.0

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

controlPlaneEndpoint: "192.168.1.199:6443"

networking:

serviceSubnet: "10.96.0.0/16"

podSubnet: "10.100.0.1/16"

dnsDomain: "cluster.local"

EOF

然后执行如下命令进行初始化：

kubeadm init --config=kubeadm-config.yaml --upload-certs

然后在master节点上执行如下命令拷贝配置文件：

[root@m1 ~]# mkdir -p $HOME/.kube

[root@m1 ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@m1 ~]# chown $(id -u):$(id -g) $HOME/.kube/config

# 查看当前的Pod信息：

kubectl get pod --all-namespaces

#### **安装calico网络 --master节点**

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

curl -ik <https://192.168.5.199:6443/version>

*kubectl create deployment nginx --image=nginx*

*kubectl expose deployment nginx --port=80 --type=NodePort*

访问地址：http://NodeIP:Port ,此例就是：<http://10.10.100.72:32393>

**多master高可用keepalived+lvs负载均衡集群**

***第一台master keepalived配置***

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 80

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass just0kk

}

virtual\_ipaddress {

192.168.5.199

}

}

virtual\_server 192.168.5.199 6443 {

delay\_loop 6

lb\_algo loadbalance

lb\_kind DR

net\_mask 255.255.255.0

persistence\_timeout 0

protocol TCP

real\_server 192.168.5.30 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.5.24 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

***第二台master keepaived***

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 80

priority 99

advert\_int 1

authentication {

auth\_type PASS

auth\_pass just0kk

}

virtual\_ipaddress {

192.168.5.199

}

}

virtual\_server 192.168.5.199 6443 {

delay\_loop 6

lb\_algo loadbalance

lb\_kind DR

net\_mask 255.255.255.0

persistence\_timeout 0

protocol TCP

real\_server 192.168.5.30 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.5.24 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

## 重新初始化master节点

rm -rf /etc/kubernetes/\*

rm -rf ~/.kube/\*

rm -rf /var/lib/etcd/\*

systemctl stop firewalld

### 第五步重置节点

kubeadm reset

注意token是master节点初始化时，生产的。如果忘记了。执行以下操作

kubeadm token create --print-join-command