链滴

kubeadm 部署 k8s

作者：fish2018

原文链接：https://ld246.com/article/1566377508465

来源网站：链滴

角色分配说明，k8s版本1.14.5

三台master 172.19.168.2527

三台node 172.19.168.2830

环境初始化

修改主机名

在各自主机执行命令设置主机名

```
hostnamectl set-hostname test-k8s-master01
。。。。。。
hostnamectl set-hostname test-k8s-node01
。。。。。。
```

配置ssh免秘钥

[root@k8s01 ~]# cat sshconfig.sh

```
#!/bin/bash
yum -y install expect
# 密钥对不存在则创建密钥
[ ! -f /root/.ssh/id_rsa.pub ] && ssh-keygen -t rsa -P '' &>/dev/null
# 要设置免密登录的机器IP 前3台master，后3台node
SERVERS="172.19.168.25 172.19.168.26 172.19.168.27 172.19.168.28 172.19.168.29 172.19.168
30"
# 目标机器要统一密码
PASSWORD=clPL35IZLKRRKyUp
PORT=9833
auto_ssh_copy_id() {
    expect -c "set timeout -1;
        spawn ssh-copy-id $1 -p$3;
        expect {
            *(yes/no)* {send -- yes\r;exp_continue;}
            *assword:* {send -- $2\r;exp_continue;}
            eof        {exit 0;}
        }";
}

ssh_copy_id_to_all() {
    for SERVER in $SERVERS
    do
        auto_ssh_copy_id $SERVER $PASSWORD $PORT
    done
}

ssh_copy_id_to_all
```

```
sh sshconfig.sh
```

关闭selinux,firewalld,iptables

```
setenforce 0 && sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
systemctl stop firewalld.service iptables.service && systemctl disable firewalld.service iptables
service
```

关闭swap

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

配置主机时间、时区、系统语言

```
ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
echo 'LANG="en_US.UTF-8"' >> /etc/profile;source /etc/profile
```

同步时间

```
systemctl start chronyd.service && systemctl enable chronyd.service
```

nofile调整

```
cat >> /etc/security/limits.conf <<EOF
* soft nofile 65535
* hard nofile 65536
EOF
```

Kernel性能调优

```
echo "
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.ipv4.conf.all.forwarding = 1
net.ipv4.neigh.default.gc_thresh1 = 4096
net.ipv4.neigh.default.gc_thresh2 = 6144
net.ipv4.neigh.default.gc_thresh3 = 8192
net.ipv4.neigh.default.gc_interval=60
net.ipv4.neigh.default.gc_stale_time=120
" >> /etc/sysctl.conf
sysctl -p
```

启用ipvs内核模块

vi ipvs_mod.sh

```
#!/bin/sh
ipvs_mods_dir="/usr/lib/modules/$(uname -r)/kernel/net/netfilter/ipvs"
for i in $(ls $ipvs_mods_dir | grep -o "^[^.]*");do
  /sbin/modinfo -F filename $i &> /dev/null
  if [ $? -eq 0 ]; then
```

```
    /sbin/modprobe $i
  fi
done

for N in `seq 26 30`;do scp -P9833 -r ipvs_mod.sh 172.19.168.$N:/root;done
# 在各主机执行脚本
sh ipvs_mod.sh
```

下面脚本创建了的/etc/sysconfig/modules/ipvs.modules文件，保证在节点重启后能自动加载所需块。

```
cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
EOF
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.mod
les && lsmod | grep -e ip_vs -e nf_conntrack_ipv4
```

Docker安装

```
# 定义安装版本
export docker_version=18.06.3
# step 1: 安装必要的一些系统工具
yum update -y;
yum install -y yum-utils device-mapper-persistent-data \
    lvm2 bash-completion;
# Step 2: 添加软件源信息
yum-config-manager --add-repo \
    http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo;
# Step 3: 更新并安装 Docker-CE
yum makecache all;
version=$(yum list docker-ce.x86_64 --showduplicates | sort -r|grep ${docker_version}|awk '{pr
nt $2}');
yum -y install --setopt=obsoletes=0 docker-ce-${version} docker-ce-selinux-${version};
```

优化配置

配置镜像下载和上传并发数

配置镜像加速地址

配置cgroups驱动

配置Docker存储驱动

配置日志驱动

```
mkdir /etc/docker/
cat <<EOF >> /etc/docker/daemon.json
{
"max-concurrent-downloads": 3,
```

```
"max-concurrent-uploads": 5,
"registry-mirrors": ["https://7bezldxe.mirror.aliyuncs.com/"],
"exec-opts": ["native.cgroupdriver=systemd"],
"storage-driver": "overlay2",
"storage-opts": ["overlay2.override_kernel_check=true"],
"log-driver": "json-file",
"log-opts": {
    "max-size": "100m",
    "max-file": "3"
    }
}
EOF
```

vi /usr/lib/systemd/system/docker.service，在[Service]下添加

```
# 防止docker服务OOM：
OOMScoreAdjust=-1000
# 开启iptables转发链：
ExecStartPost=/usr/sbin/iptables -P FORWARD ACCEPT
```

分发到其他节点

```
for N in `seq 26 30`;do scp -P9833 -r /usr/lib/systemd/system/docker.service 172.19.168.$N:/u
r/lib/systemd/system/;done
```

设置docker开机启动

```
systemctl daemon-reload
systemctl restart docker
systemctl enable docker
```

hosts修改

```
cat <<EOF >> /etc/hosts
172.19.168.25 test-k8s-master01 k8s-api2.hhotel.com
172.19.168.26 test-k8s-master02 k8s-api2.hhotel.com
172.19.168.27 test-k8s-master03 k8s-api2.hhotel.com
172.19.168.28 test-k8s-node01
172.19.168.29 test-k8s-node02
172.19.168.30 test-k8s-node03
EOF
for N in `seq 25 30`;do scp -P9833 -r /etc/hosts 172.19.168.$N:/etc;done
```

配置阿里k8s的yum源

```
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.
```

```
om/kubernetes/yum/doc/rpm-package-key.gpg
EOF
```

部署kubernetes

所有节点安装kubeadm kubelet kubectl

```
# 查看可用版本yum list kubeadm --showduplicates
echo y | yum list
yum -y install kubeadm-1.14.5 kubelet-1.14.5 kubectl-1.14.5
systemctl enable kubelet.service
systemctl restart kubelet.service
```

创建kubeadm配置文件

```
cat <<EOF > kubeadm-config.yaml
### 1.15.1+用v1beta2, 1.14.5用v1beta1
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: v1.14.5
imageRepository: gcr.azk8s.cn/google_containers
apiServer:
  certSANs:
  - "k8s-api2.hhotel.com"
### apiServer的集群访问地址
controlPlaneEndpoint: "k8s-api2.hhotel.com:6443"
### 网络插件的子网
networking:
  podSubnet: "192.168.0.0/16"
  dnsDomain: cluster.local
  serviceSubnet: "10.96.0.0/12"
EOF

for N in `seq 26 30`;do scp -P9833 -r kubeadm-config.yaml 172.19.168.$N:~;done
```

每台机器执行下面命令拉取用到的镜像

```
kubeadm config images pull --config kubeadm-config.yaml
```

kubeadm初始化，在master01执行

```
kubeadm init --config kubeadm-config.yaml --experimental-upload-certs
```

输出结果：

```
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

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
ach as root:

  kubeadm join api.k8s.hhotel.com:6443 --token 9n6g0e.v7u8nao1isirg2yt \
    --discovery-token-ca-cert-hash sha256:0b1d6f35288fb2640b54e9a0f4017d43ca39420911e
d15a0b1f779a078ae02c \
    --control-plane --certificate-key 6c07dd5fe113850b24e273721a91eaed97ffd520a31b36eee
09e983b4c5d58b

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 api.k8s.hhotel.com:6443 --token 9n6g0e.v7u8nao1isirg2yt \
    --discovery-token-ca-cert-hash sha256:0b1d6f35288fb2640b54e9a0f4017d43ca39420911e
d15a0b1f779a078ae02c
```

第一个join是用来添加master，第二个join用来添加node

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

for N in `seq 28 30`;do scp -P9833 -r $HOME/.kube 172.19.168.$N:~;done
```

安装canal网络插件

修改网络段和kubeadm init中的 --pod-network-cidr=192.168.0.0/16 保持一致

```
wget https://docs.projectcalico.org/v3.8/manifests/canal.yaml
sed -i s@"10.244.0.0/16"@"192.168.0.0/16"@g canal.yaml
kubectl apply -f canal.yaml
```

去除污点，使master可以作为node，只要join master后去除污点这个机器就既是master又是node了

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

token等秘钥失效后重新获取

```
# 默认token有效期是24小时,如果token已经过期,以下命令重新生成
kubeadm token create
# 生成--discovery-token-ca-cert-hash的值
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev
```

```
null | openssl dgst -sha256 -hex | sed 's/^.* //'
# ertificate-key用于其他master节点获取证书文件时验证，有小时间为2小时，超过2小时候需要重生成
kubeadm init phase upload-certs --experimental-upload-certs
# 然后再使用kubeadm join命令加入Master或node
kubeadm join api.k8s.hhotel.com:6443 --token dhn1rz.ij36pu94qpkj43xk \
--discovery-token-ca-cert-hash sha256:0b1d6f35288fb2640b54e9a0f4017d43ca39420911e9d5a0b1f779a078ae02c \
--control-plane --certificate-key e5264a589b2133d328b599dc9c94e3d4f8c51dd6e5e9887752a22b9af5bc622
# 只生成join node的命令
kubeadm token create --print-join-command
```

部署出现问题可以在当前节点重置kubeadm

```
kubeadm reset
rm -rf * ~/.kube/*
```