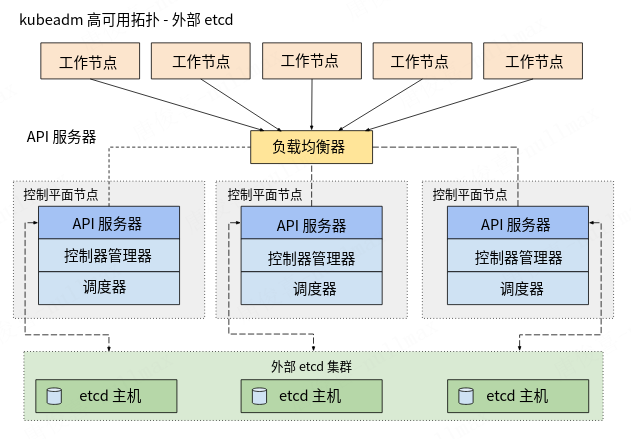
**Kubernetes高可用集群搭建-02**



**一、成员角色**

**1.示意图**



**2.IP地址以及角色划分**

|  |
| --- |
| Bash #etcd节点 192.168.135.149 k8-etcd-01 192.168.135.150 k8-etcd-02 192.168.135.147 k8-etcd-03 #...  #主节点 192.168.135.151 k8-master-01 192.168.135.155 k8-master-02 192.168.135.156 k8-master-03 #...  #负载均衡器 192.168.135.153 k8-apiserver-01 192.168.135.150 k8-apiserver-02 #...   #工作节点 192.168.135.152 k8-slave-01 192.168.135.159 k8-slave-02 192.168.135.160 k8-slave-03 #... |

**一、配置网络环境**

**1.配置kubernetest ipv4转发流量**

|  |
| --- |
| Bash sudo tee /etc/modules-load.d/k8s.conf <<EOF overlay br\_netfilter EOF |

|  |
| --- |
| Bash sudo modprobe overlay sudo modprobe br\_netfilter |

|  |
| --- |
| Bash sudo tee /etc/sysctl.d/k8s.conf <<EOF net.bridge.bridge-nf-call-ip6tables = 1 net.bridge.bridge-nf-call-iptables = 1 net.ipv4.ip\_forward = 1 EOF |

|  |
| --- |
| Bash sudo sysctl --system |

|  |
| --- |
| Bash sysctl net.bridge.bridge-nf-call-iptables net.bridge.bridge-nf-call-ip6tables net.ipv4.ip\_forward |

**2.关闭swap分区**

|  |
| --- |
| Bash sudo nano /etc/fstab #nano #Ctrl+O 保存 #Ctrl+X 退出 #一般系统有两种挂载磁盘方式 注释掉  #UUID=xxxx-xxxx-xxxx-xxxx swap swap defaults 0 0  或者   #/swapfile none swap sw 0 0 #非LVM安装ubuntu  #linux-desktop  sed -i 's@/swap@#/swap@g' /etc/fstab  #linux-server  sed -i 's@/swap.img@#/swap.img@g' /etc/fstab #切勿注释掉系统分区 #否则无法正常开机 sudo swapoff -a #重启系统以确保更改持久生效。 #请注意，在编辑/etc/fstab文件时要小心，因为不正确的更改可能会导致系统无法启动。如果你不确定，请先在非生产系统上尝试这些步骤。 |

**3.安装配置容器运行时**

**1.下载containerd**

|  |
| --- |
| Bash #使用Debain原生 sudo apt update  sudo apt install -y containerd |

**2.修改containerd默认配置**

|  |
| --- |
| Bash sudo mkdir /etc/containerd sudo touch /etc/containerd/config.toml  #把containerd默认配置导入config.toml文件 containerd config default > /etc/containerd/config.toml |

|  |
| --- |
| Bash #修改配置 nano /etc/containerd/config.toml  #修改镜像源 把sandbox\_image = "registry.k8s.io/pause:3.8" 修改为sandbox\_image = "registry.aliyuncs.com/google\_containers/pause:3.9"  #启用系统文件系统 把SystemdCgroup设置为true SystemdCgroup = true |

|  |
| --- |
| Bash #直接使用指令 sed -i 's@registry.k8s.io/pause:3.8@registry.aliyuncs.com/google\_containers/pause:3.9@g' /etc/containerd/config.toml sed -i 's@SystemdCgroup = false@SystemdCgroup = true@g' /etc/containerd/config.toml |

|  |
| --- |
| Bash #对于containerd修改/etc/containerd/config.toml文件 #找到 [plugins."io.containerd.grpc.v1.cri".registry.configs] #和 [plugins."io.containerd.grpc.v1.cri".registry.mirrors] #在下面添加配置 nano /etc/containerd/config.toml #忽略证书问题 #找到[plugins."io.containerd.grpc.v1.cri".registry.configs]在下面添加：  [plugins."io.containerd.grpc.v1.cri".registry.configs."registry.docker.io:5000".tls]  insecure\_skip\_verify = true  #**insecure\_skip\_verify**是一个配置选项，用于在TLS连接中忽略证书验证。  #当设置为true时，可以忽略SSL证书的验证，这通常用于测试环境或与自签名证书的服务器通信。     #添加镜像地址 #找到[plugins."io.containerd.grpc.v1.cri".registry.mirrors]在下面添加：  [plugins."io.containerd.grpc.v1.cri".registry.mirrors."docker.io"]  endpoint = ["https://docker.ketches.cn"]  [plugins."io.containerd.grpc.v1.cri".registry.mirrors."registry.docker.io:5000"]  endpoint = ["http://registry.docker.io:5000"] |

**3.保存重启containerd**

|  |
| --- |
| Bash #重载daemon sudo systemctl daemon-reload sudo systemctl restart containerd  #验证版本信息 sudo containerd --veriosn |

**二、下载安装kubernetes**

**1.配置apt-get源**

**1.更新 apt 包索引，并安装使用 Kubernetes apt 仓库所需要的包：**

|  |
| --- |
| Bash sudo apt-get update *# apt-transport-https 可以是一个虚拟包；如果是这样，你可以跳过这个包* sudo apt-get install -y apt-transport-https ca-certificates curl gnupg |

**2.下载 Kubernetes 软件包仓库的公共签名密钥。 同一个签名密钥适用于所有仓库，因此你可以忽略 URL 中的版本信息：**

|  |
| --- |
| Bash *# 如果 `/etc/apt/keyrings` 目录不存在，则应在 curl 命令之前创建它，请阅读下面的注释。* *sudo mkdir -p -m 755 /etc/apt/keyrings* curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28.2/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg *# allow unprivileged APT programs to read this keyring* |

**添加合适的 Kubernetes apt 仓库。如果你想用 v1.32 之外的 Kubernetes 版本， 请将下面命令中的 v1.32 替换为所需的次要版本：**

|  |
| --- |
| Bash *# 这会覆盖 /etc/apt/sources.list.d/kubernetes.list 中的所有现存配置* # 使用阿里云镜像 sudo curl -fsSL https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg |  sudo apt-key add sudo echo "deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main" |  sudo tee /etc/apt/sources.list.d/kubernetes.list |

**2.下载kubernetes**

**修改/etc/apt/sources.list.d/kubernetes.list 中的版本， 再运行 apt-get update 和 apt-get upgrade 命令。 更详细的步骤可以在**[**更改 Kubernetes 软件包存储库**](https://kubernetes.io/zh-cn/docs/tasks/administer-cluster/kubeadm/change-package-repository/)**中找到。**

更新 apt 包索引，然后安装 kubectl、kubeadm、kubelet：

|  |
| --- |
| Bash sudo apt-get update sudo apt-get install -y kubectl kubeadm kubelet sudo apt-mark hold kubelet kubeadm kubectl #固定版本防止自动升级 |

**3.配置kubernetes**

|  |
| --- |
| Bash #导出默认配置 sudo mkdir /etc/kubernetes sudo touch /etc/kubernetes/kubeadm-init.yaml kubeadm config print init-defaults > /etc/kubernetes/kubeadm-init.yaml |

|  |
| --- |
| Bash #目前建议手动 #修改节点名字 nano /etc/kubernetes/kubeadm-init.yaml  #slave节点就改为对应的节点名 name: k8-master-01 #修改版本 kubernetesVersion: 1.28.2 #修改IP advertiseAddress： 192.168.135.136 #your host #修改镜像源 #把镜像修改为： imageRepository: registry.aliyuncs.com/google\_containers #添加podSubnet: podSubnet: 10.244.0.0/16 |

|  |
| --- |
| Bash #使用指令 #k8-master-01 sed -i 's@name: node@name: k8-master-01@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@advertiseAddress: 1.2.3.4@advertiseAddress: 192.168.135.151@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@kubernetesVersion: 1.28.0@kubernetesVersion: 1.28.2@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@imageRepository: registry.k8s.io@imageRepository: registry.aliyuncs.com/google\_containers@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@serviceSubnet: 10.96.0.0/12@serviceSubnet: 10.96.0.0/12\n podSubnet: 10.244.0.0/16@g' /etc/kubernetes/kubeadm-init.yaml |

**4.配置外部etcd（kubelet etcd服务集群）**

**1.添加如下配置**

|  |
| --- |
| Bash cat << EOF > /usr/lib/systemd/system/kubelet.service.d/20-etcd-service-manager.conf [Service] ExecStart= ExecStart=/usr/bin/kubelet --address=127.0.0.1 --cgroup-driver=systemd --pod-manifest-path=/etc/kubernetes/manifests/ Restart=always EOF |

|  |
| --- |
| Bash #重载kubelet服务 systemctl daemon-reload systemctl restart kubelet |

**2.编写为每个etcd成员创建证书的脚本**

|  |
| --- |
| Bash cat *<< EOF > /etc/kubernetes/create-crts.sh*  *# Update HOST0, HOST1 and HOST2 with the IPs of your hosts*  *export HOST0=192.168.135.149*  *export HOST1=192.168.135.150*  *export HOST2=192.168.135.147*  *# Update NAME0, NAME1 and NAME2 with the hostnames of your hosts*  *export NAME0="k8-etcd-01"*  *export NAME1="k8-etcd-02"*  *export NAME2="k8-etcd-03"*  *# Create temp directories to store files that will end up on other hosts*  *mkdir -p /tmp/****${****HOST0****}****/ /tmp/****${****HOST1****}****/ /tmp/****${****HOST2****}****/*  *HOSTS=(****${****HOST0****}******${****HOST1****}******${****HOST2****}****)*  *NAMES=(****${****NAME0****}******${****NAME1****}******${****NAME2****}****)*  ***for*** *i in "****${****!HOSTS[@]****}****";*   ***do***  *HOST=****${****HOSTS[$i]****}***  *NAME=****${****NAMES[$i]****}***  *cat << EOF > /tmp/${HOST}/kubeadmcfg.yaml*  *---*  *apiVersion: "kubeadm.k8s.io/v1beta3"*  *kind: InitConfiguration*  *nodeRegistration:*   *name: ${NAME}*  *localAPIEndpoint:  advertiseAddress: ${HOST}*  *---*  *apiVersion: "kubeadm.k8s.io/v1beta3"*  *kind: ClusterConfiguration*  *etcd:  local:  serverCertSANs:  - "${HOST}"  peerCertSANs:  - "${HOST}"  extraArgs:  initial-cluster: ${NAMES[0]}=https://${HOSTS[0]}:2380,${NAMES[1]}=https://${HOSTS[1]}:2380,${NAMES[2]}=https://${HOSTS[2]}:2380  initial-cluster-state: new  name: ${NAME}  listen-peer-urls: https://${HOST}:2380  listen-client-urls: https://${HOST}:2379  advertise-client-urls: https://${HOST}:2379  initial-advertise-peer-urls: https://${HOST}:2380*  *imageRepository: registry.aliyuncs.com/google\_containers*  *kubernetesVersion: v1.28.2*  *EOF*  ***done*** EOF |

|  |
| --- |
| Bash #执行脚本 sudo chmod +x */etc/kubernetes/create-crts.sh* *bash /etc/kubernetes/create-crts.sh* |

**3.为你首个执行操作的etcd服务器创建证书**

|  |
| --- |
| Bash kubeadm init phase certs etcd-ca --kubernetes-version v1.28.2 |

**4.在首个执行操作的etcd服务器上，为每个etcd集群创建证书**

|  |
| --- |
| Bash #HOST2 kubeadm init phase certs etcd-server --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml  kubeadm init phase certs etcd-peer --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml  kubeadm init phase certs etcd-healthcheck-client --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml  kubeadm init phase certs apiserver-etcd-client --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml  cp -R /etc/kubernetes/pki /tmp/**${**HOST2**}**/ *# cleanup non-reusable certificates* find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  #HOST1 kubeadm init phase certs etcd-server --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml  kubeadm init phase certs etcd-peer --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml  kubeadm init phase certs etcd-healthcheck-client --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml  kubeadm init phase certs apiserver-etcd-client --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml  cp -R /etc/kubernetes/pki /tmp/**${**HOST1**}**/ find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  #HOST0 kubeadm init phase certs etcd-server --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml  kubeadm init phase certs etcd-peer --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml  kubeadm init phase certs etcd-healthcheck-client --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml  kubeadm init phase certs apiserver-etcd-client --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml  *# No need to move the certs because they are for HOST0* *# clean up certs that should not be copied off this host* find /tmp/**${**HOST2**}** -name ca.key -type f -delete find /tmp/**${**HOST1**}** -name ca.key -type f -delete |

|  |
| --- |
| Bash #确保证书存在 tree /tmp/${HOST0}/ /etc/kubernetes/pki /tmp/${HOST1}/ /tmp/${HOST2}/ |

|  |
| --- |
| Bash /tmp/192.168.135.149/ └── kubeadmcfg.yaml /etc/kubernetes/pki/ ├── apiserver-etcd-client.crt ├── apiserver-etcd-client.key └── etcd  ├── ca.crt  ├── ca.key  ├── healthcheck-client.crt  ├── healthcheck-client.key  ├── peer.crt  ├── peer.key  ├── server.crt  └── server.key /tmp/192.168.135.150/ ├── kubeadmcfg.yaml └── pki  ├── apiserver-etcd-client.crt  ├── apiserver-etcd-client.key  └── etcd  ├── ca.crt  ├── healthcheck-client.crt  ├── healthcheck-client.key  ├── peer.crt  ├── peer.key  ├── server.crt  └── server.key /tmp/192.168.135.147/ ├── kubeadmcfg.yaml └── pki  ├── apiserver-etcd-client.crt  ├── apiserver-etcd-client.key  └── etcd  ├── ca.crt  ├── healthcheck-client.crt  ├── healthcheck-client.key  ├── peer.crt  ├── peer.key  ├── server.crt  └── server.key |

**5.在首个执行操作的etcd服务器上，把证书copy到对应的etcd集群成员**

|  |
| --- |
| Bash USER=${NAME2} HOST=${HOST2} scp -r /tmp/${HOST}/\* ${USER}@${HOST}: ssh ${USER}@${HOST} USER@HOST $ sudo -Es root@HOST $ chown -R root:root pki root@HOST $ mv pki /etc/kubernetes/ |

**6.在每个etcd集群成员服务器上，创建etcd 静态 pod**

|  |
| --- |
| Bash root@HOST0 $ kubeadm init phase etcd local --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml root@HOST1 $ kubeadm init phase etcd local --config=/home/k8-etcd-02/kubeadmcfg.yaml root@HOST2 $ kubeadm init phase etcd local --config=/home/k8-etcd-03/kubeadmcfg.yaml |

**7.检查etcd集群是否正常工作**

|  |
| --- |
| Bash #查看集群成员是否正常运行 docker run --rm -it \ --net host \ -v /etc/kubernetes:/etc/kubernetes registry.aliyuncs.com/google\_containers/etcd:3.5.9-0 etcdctl \ --cert /etc/kubernetes/pki/etcd/peer.crt **\** --key /etc/kubernetes/pki/etcd/peer.key **\** --cacert /etc/kubernetes/pki/etcd/ca.crt **\** --endpoints https://**${HOST0}**:2379 endpoint health  #查看etcd集群成员 docker run --rm -it \ --net host \ -v /etc/kubernetes:/etc/kubernetes registry.aliyuncs.com/google\_containers/etcd:3.5.9-0 etcdctl \  --endpoints 192.168.135.149:2379 \  --cert=/etc/kubernetes/pki/etcd/server.crt \  --key=/etc/kubernetes/pki/etcd/server.key \  --cacert=/etc/kubernetes/pki/etcd/ca.crt \  member list |

**三、下载负载keepalived均衡器和haproxy代理**

**1.主负载均衡器配置**

**1.安装**

|  |
| --- |
| Bash sudo apt install -y haproxy keepalived |

**2.配置keepalived(标红需要自己配置)**

|  |
| --- |
| Bash #创建keepalived.conf文件 sudo nano /etc/keepalived/keepalived.conf |

|  |
| --- |
| Bash cat <<EOF> /etc/keepalived/keepalived.conf ! /etc/keepalived/keepalived.conf ! Configuration File for keepalived  global\_defs {  script\_user root  router\_id VRRP-01  enable\_script\_security } vrrp\_script check\_apiserver {  script "/etc/keepalived/check\_apiserver.sh"  interval 3  weight -2  fall 10  rise 2 }  vrrp\_instance VI\_1 {  state MASTER  interface ens33  virtual\_router\_id 50  priority 101  authentication {  auth\_type PASS  auth\_pass Kubernetes  }  virtual\_ipaddress {  192.168.135.143  }  track\_script {  check\_apiserver  } }  EOF |

|  |
| --- |
| Bash #创建check\_apiserver.sh脚本文件 #sudo nano /etc/keepalived/check\_apiserver.sh  cat <<EOF>> /etc/keepalived/check\_apiserver.sh #!/bin/sh errorExit() {  echo "\*\*\* $\*" 1>&2  exit 1 }  curl -sfk --max-time 2 https://localhost:6443/healthz -o /dev/null || errorExit "Error GET https://localhost:6443/healthz" EOF |

**3.配置haproxy(标红需要自己配置)**

|  |
| --- |
| Bash  #修改haproxy.cfg文件  sudo nano /etc/haproxy/haproxy.cfg  #清除掉原有内容 |

|  |
| --- |
| Bash cat <<EOF> /etc/haproxy/haproxy.cfg # /etc/haproxy/haproxy.cfg #--------------------------------------------------------------------- # Global settings #--------------------------------------------------------------------- global  log stdout format raw local0  daemon  #--------------------------------------------------------------------- # common defaults that all the 'listen' and 'backend' sections will # use if not designated in their block #--------------------------------------------------------------------- defaults  mode http  log global  option httplog  option dontlognull  option http-server-close  option forwardfor except 127.0.0.0/8  option redispatch  retries 1  timeout http-request 10s  timeout queue 20s  timeout connect 5s  timeout client 35s  timeout server 35s  timeout http-keep-alive 10s  timeout check 10s  #--------------------------------------------------------------------- # apiserver frontend which proxys to the control plane nodes #--------------------------------------------------------------------- frontend apiserver  #bind \*:${APISERVER\_DEST\_PORT}  #修改绑定端口  bind \*:6443  mode tcp  option tcplog  default\_backend apiserverbackend  #--------------------------------------------------------------------- # round robin balancing for apiserver #--------------------------------------------------------------------- backend apiserverbackend  option httpchk   http-check connect ssl  http-check send meth GET uri /healthz  http-check expect status 200   mode tcp  balance roundrobin    #server ${HOST1\_ID} ${HOST1\_ADDRESS}:${APISERVER\_SRC\_PORT} check verify none  #修改kubernetes服务器地址以及端口  server k8-master-01 192.168.135.136:6443 check verify none  server k8-master-02 192.168.135.134:6443 check verify none  server k8-master-03 192.168.135.145:6443 check verify none  # [...] EOF |

**4.启用keepalived和haproxy**

|  |
| --- |
| Bash sudo systemctl enable/disable haproxy --now sudo systemctl enable/disable keepalived --now |

**2.配置备用负载均衡器**

**1.安装**

|  |
| --- |
| Bash sudo apt install -y haproxy keepalived |

**2.配置keepalived(标红需要自己配置)**

|  |
| --- |
| Bash cat <<EOF> /etc/keepalived/keepalived.conf ! /etc/keepalived/keepalived.conf ! Configuration File for keepalived  global\_defs {  script\_user root  router\_id VRRP-02  enable\_script\_security } vrrp\_script check\_apiserver {  script "/etc/keepalived/check\_apiserver.sh"  interval 3  weight -2  fall 10  rise 2 }  vrrp\_instance VI\_1 {  state BACKUP  interface ens33  virtual\_router\_id 50  priority 100  authentication {  auth\_type PASS  auth\_pass Kubernetes  }  virtual\_ipaddress {  192.168.135.146  }  track\_script {  check\_apiserver  } } EOF |

|  |
| --- |
| Bash #创建check\_apiserver.sh脚本文件 #sudo nano /etc/keepalived/check\_apiserver.sh  cat <<EOF>> /etc/keepalived/check\_apiserver.sh #!/bin/sh errorExit() {  echo "\*\*\* $\*" 1>&2  exit 1 }  curl -sfk --max-time 2 https://localhost:6443/healthz -o /dev/null || errorExit "Error GET https://localhost:6443/healthz" EOF |

**3.配置haproxy(标红需要自己配置)**

|  |
| --- |
| Bash  #修改haproxy.cfg文件  sudo nano /etc/haproxy/haproxy.cfg  #清除掉原有内容 |

|  |
| --- |
| Bash cat <<EOF> /etc/haproxy/haproxy.cfg # /etc/haproxy/haproxy.cfg #--------------------------------------------------------------------- # Global settings #--------------------------------------------------------------------- global  log stdout format raw local0  daemon  #--------------------------------------------------------------------- # common defaults that all the 'listen' and 'backend' sections will # use if not designated in their block #--------------------------------------------------------------------- defaults  mode http  log global  option httplog  option dontlognull  option http-server-close  option forwardfor except 127.0.0.0/8  option redispatch  retries 1  timeout http-request 10s  timeout queue 20s  timeout connect 5s  timeout client 35s  timeout server 35s  timeout http-keep-alive 10s  timeout check 10s  #--------------------------------------------------------------------- # apiserver frontend which proxys to the control plane nodes #--------------------------------------------------------------------- frontend apiserver  #bind \*:${APISERVER\_DEST\_PORT}  #修改绑定端口  bind \*:6443  mode tcp  option tcplog  default\_backend apiserverbackend  #--------------------------------------------------------------------- # round robin balancing for apiserver #--------------------------------------------------------------------- backend apiserverbackend  option httpchk   http-check connect ssl  http-check send meth GET uri /healthz  http-check expect status 200   mode tcp  balance roundrobin    #server ${HOST1\_ID} ${HOST1\_ADDRESS}:${APISERVER\_SRC\_PORT} check verify none  #修改kubernetes服务器地址以及端口  server k8-master-01 192.168.135.136:6443 check verify none  server k8-master-02 192.168.135.134:6443 check verify none  server k8-master-03 192.168.135.145:6443 check verify none  # [...] EOF |

**4.启用keepalived和haproxy**

|  |
| --- |
| Bash sudo systemctl enable/disable haproxy --now sudo systemctl enable/disable keepalived --now |

5.查看负载均衡器和API SERVER代理情况

|  |
| --- |
| Bash sudo systemctl status haproxy sudo systemctl status keepalived |

**四、初始化kubernetes集群**

**1.配置Kubernetes初始化文件（任意主节点操作）**

|  |
| --- |
| Bash #导出默认配置 sudo mkdir /etc/kubernetes #sudo touch /etc/kubernetes/kubeadm-init.yaml kubeadm config print init-defaults > /etc/kubernetes/kubeadm-init.yaml |

|  |
| --- |
| Bash #目前建议手动 #修改节点名字 nano /etc/kubernetes/kubeadm-init.yaml  #slave节点就改为对应的节点名 name: k8-master  #修改版本 kubernetesVersion: 1.28.2  #修改IP advertiseAddress： 192.168.135.136 #your host  #修改镜像源 #把镜像修改为： imageRepository: registry.aliyuncs.com/google\_containers  #添加podSubnet: podSubnet: 10.244.0.0/16  #添加controlPlaneEndpoint(VRRP虚拟IP): controlPlaneEndpoint: "192.168.135.148:6443" |

|  |
| --- |
| Bash #使用指令 #k8-master-01 sed -i 's@node@k8master01@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@1.2.3.4@192.168.135.136@g' /etc/kubernetes/kubeadm-init.yaml #k8-master-02 sed -i 's@node@k8master02@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@1.2.3.4@192.168.135.135@g' /etc/kubernetes/kubeadm-init.yaml   sed -i 's@1.28.0@1.28.2@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@kubernetesVersion: 1.28.2@kubernetesVersion: 1.28.2\ncontrolPlaneEndpoint: "192.168.135.145:6443"@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@registry.k8s.io@registry.aliyuncs.com/google\_containers@g' /etc/kubernetes/kubeadm-init.yaml sed -i 's@serviceSubnet: 10.96.0.0/12@serviceSubnet: 10.96.0.0/12\n\1podSubnet: 10.244.0.0/16@g' /etc/kubernetes/kubeadm-init.yaml |

|  |
| --- |
| Bash #模板 #/etc/kubernetes/kubeadm-init.yaml apiVersion: kubeadm.k8s.io/v1beta3 bootstrapTokens: - groups:  - system:bootstrappers:kubeadm:default-node-token  token: abcdef.0123456789abcdef  ttl: 24h0m0s  usages:  - signing  - authentication kind: InitConfiguration localAPIEndpoint:  advertiseAddress: 192.168.135.151  bindPort: 6443 nodeRegistration:  criSocket: unix:///var/run/containerd/containerd.sock  imagePullPolicy: IfNotPresent  name: k8-master-01  taints: null --- apiServer:  timeoutForControlPlane: 4m0s apiVersion: kubeadm.k8s.io/v1beta3 certificatesDir: /etc/kubernetes/pki clusterName: kubernetes controlPlaneEndpoint: "192.168.135.158:16443" controllerManager: {} dns: {} etcd:  external:  endpoints:  - https://k8-etcd-01:2379  - https://k8-etcd-02:2379  - https://k8-etcd-03:2379  caFile: /etc/kubernetes/pki/etcd/ca.crt  certFile: /etc/kubernetes/pki/apiserver-etcd-client.crt  keyFile: /etc/kubernetes/pki/apiserver-etcd-client.key  imageRepository: registry.aliyuncs.com/google\_containers kind: ClusterConfiguration kubernetesVersion: 1.28.2 networking:  dnsDomain: cluster.local  serviceSubnet: 10.96.0.0/12  podSubnet: 10.244.0.0/16 scheduler: {} |

**2.初始化其中一个master节点**

|  |
| --- |
| Bash #任意一台master节点执行 kubeadm init --config kubeadm-init.yaml |

**3.拷贝证书并加入集群**

**1.控制节点加入前拷贝证书（依情况修改）**

|  |
| --- |
| Bash #k8-master-n mkdir /etc/kubernetes/pki#存储证书 mkdir /etc/kubernetes/pki/etcd #存储证书 sudo chown -R k8-master-n:k8-master-n /etc/kubernetes |

|  |
| --- |
| Bash #k8-master-n -> k8-master-01 #kubernetes集群证书 scp /etc/kubernetes/pki/ca.crt k8-master-01@192.168.135.151:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/ca.key k8-master-01@192.168.135.151:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.key k8-master-01@192.168.135.151:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.pub k8-master-01@192.168.135.151:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.crt k8-master-01@192.168.135.151:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.key k8-master-01@192.168.135.151:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/etcd/ca.crt k8-master-01@192.168.135.151:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/pki/etcd/ca.key k8-master-01@192.168.135.151:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/admin.conf k8-master-01@192.168.135.151:/etc/kubernetes/ #etcd集群证书 scp /etc/kubernetes/pki/apiserver-etcd-client.crt k8-master-01@192.168.135.151:/etc/kubernetes/pki scp /etc/kubernetes/pki/apiserver-etcd-client.key k8-master-01@192.168.135.151:/etc/kubernetes/pki  #k8-master-n -> k8-master-02 scp /etc/kubernetes/pki/ca.crt k8-master-02@192.168.135.155:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/ca.key k8-master-02@192.168.135.155:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.key k8-master-02@192.168.135.155:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.pub k8-master-02@192.168.135.155:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.crt k8-master-02@192.168.135.155:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.key k8-master-02@192.168.135.155:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/etcd/ca.crt k8-master-02@192.168.135.155:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/pki/etcd/ca.key k8-master-02@192.168.135.155:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/admin.conf k8-master-02@192.168.135.155:/etc/kubernetes/  scp /etc/kubernetes/pki/apiserver-etcd-client.crt k8-master-02@192.168.135.155:/etc/kubernetes/pki scp /etc/kubernetes/pki/apiserver-etcd-client.key k8-master-02@192.168.135.155:/etc/kubernetes/pki   #k8-master-n -> k8-master-03 scp /etc/kubernetes/pki/ca.crt k8-master-03@192.168.135.156:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/ca.key k8-master-03@192.168.135.156:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.key k8-master-03@192.168.135.156:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.pub k8-master-03@192.168.135.156:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.crt k8-master-03@192.168.135.156:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.key k8-master-03@192.168.135.156:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/etcd/ca.crt k8-master-03@192.168.135.156:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/pki/etcd/ca.key k8-master-03@192.168.135.156:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/admin.conf k8-master-03@192.168.135.156:/etc/kubernetes/  scp /etc/kubernetes/pki/apiserver-etcd-client.crt k8-master-03@192.168.135.156:/etc/kubernetes/pki scp /etc/kubernetes/pki/apiserver-etcd-client.key k8-master-03@192.168.135.156:/etc/kubernetes/pki |

**2.加入集群**

**1.控制节点加入**

|  |
| --- |
| Bash #k8-master-01、k8-master-02、k8-master-03... scp k8-master-03@192.168.135.145:/etc/kubernetes/admin.conf /etc/kubernetes/  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 |

|  |
| --- |
| Bash #You can now join any number of control-plane nodes by copying certificate authorities and service account keys on each node and then running the following as root: kubeadm join 192.168.135.158:6443 --token xxxx.xxxxxx \  --discovery-token-ca-cert-hash sha256: xxxxxxx..xxxx\  --control-plane   #example: kubeadm join 192.168.135.158:6443 --token abcdef.0123456789abcdef \  --discovery-token-ca-cert-hash sha256: 81feaef2e840d2c5b3751d1ffa98aa7e79701eefb988685dc07996481bd42108 \  --control-plane |

**2.工作节点加入**

|  |
| --- |
| Bash mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config  #红色部分已经变成负载均衡器上的虚拟控制节点的地址 #Then you can join any number of worker nodes by running the following on each as root: kubeadm join 192.168.135.158:6443 --token xxxx.xxxxxx \  --discovery-token-ca-cert-hash sha256:xxxxx....xxxxxxx  #example: kubeadm join 192.168.135.158:6443 --token abcdef.0123456789abcdef \  --discovery-token-ca-cert-hash sha256:d77d919dca4a5a9b79362ca70ca2a4d819a0e87b813f56fdffa7df68d865d476 |

**4.删除节点**

**1.获取所有节点**

|  |
| --- |
| Bash #在Kubernetes中删除节点的步骤如下： #**获取节点列表并确定要删除的节点**‌： #使用命令kubectl get nodes获取所有节点的名称 kubectl get nodes |

**2.驱除节点**

|  |
| --- |
| Bash #**驱逐节点上的Pod**‌： #使用kubectl drain命令驱逐要删除节点上的Pod，以确保在删除节点时不会中断应用程序。 #例如，kubectl drain <node-name> --ignore-daemonsets，其中<node-name>是要删除的节点名称。 kubectl drain <node-name> --ignore-daemonsets |

**3.删除节点**

|  |
| --- |
| Bash #**删除节点**‌： #使用kubectl delete node <node-name>命令删除节点。例如，kubectl delete node 127.0.0.1，其中127.0.0.1是要删除的节点名称。 kubectl delete node <node-name> |

**4.重置kubernetes**

|  |
| --- |
| Bash #**重置节点**‌： #如果需要在删除节点后重新将其加入集群，可以在该节点上执行kubeadm reset命令来清理Kubernetes集群信 kubectl reset |

**五、证书或者Token过期问题**

**1.token过期，重新生成新的token**

|  |
| --- |
| Bash kubeadm token create --print-join-command |

|  |
| --- |
| Bash #获取token列表 kubeadm token list |

|  |
| --- |
| Bash #查看 --discovery-token-ca-cert-hash哈希值 openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.\* //' |

**2.查看目前证书有效期**

|  |
| --- |
| Bash #查看证书有效期 openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text | grep Not |

|  |
| --- |
| Bash #下载官方证书更新脚本 git clone https://github.com/yuyicai/update-kube-cert.git cd update-kube-cert chmod 755 update-kubeadm-cert.sh |

|  |
| --- |
| Bash #修改过期时间 sed -i '1,$s/CERT\_DAYS=3650/CERT\_DAYS=36500/g' update-kubeadm-cert.sh  ./update-kubeadm-cert.sh all ./update-kubeadm-cert.sh all --cri containerd |

|  |
| --- |
| Bash #再次查看证书过期时间 openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text | grep Not |