**Kubernetes多主节点集群搭建（Ubuntu 22.04.1）**

Copyright@2025-01-10

Wrote by JunXiTang

**参考文档**

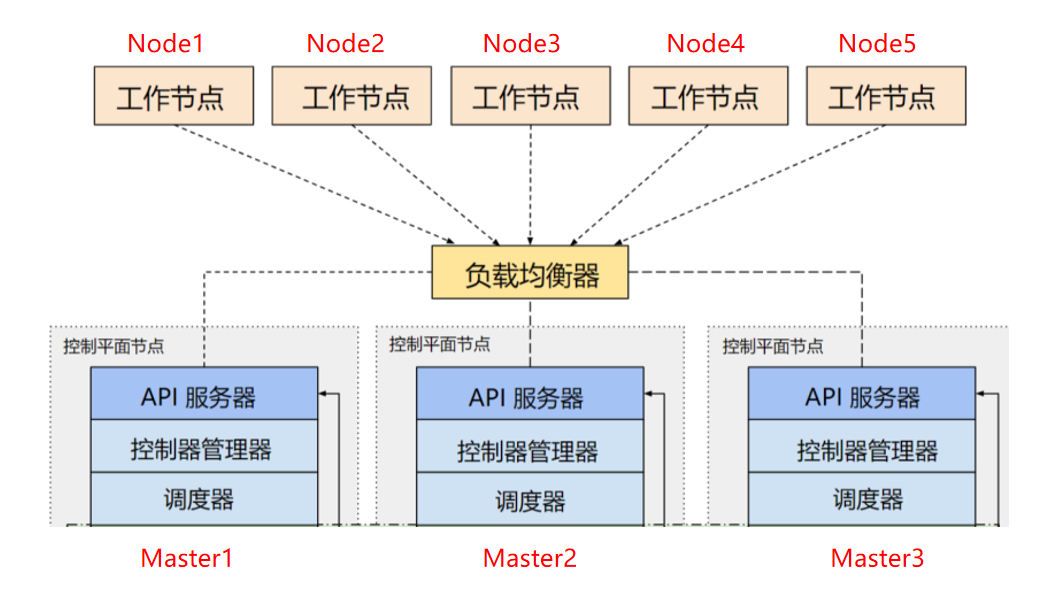
|  |
| --- |
| Bash 1.配置高可用kubernetes集群（真只是参考） https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/ 2.配置负载均衡器 https://github.com/kubernetes/kubeadm/blob/main/docs/ha-considerations.md#options-for-software-load-balancing  不使用加速器可用先配置hosts文件：  20.205.243.166 github.com 下载calico和flannel等网络插件： 185.199.108.133 raw.githubusercontent.com |

**基础机器和软件配置**

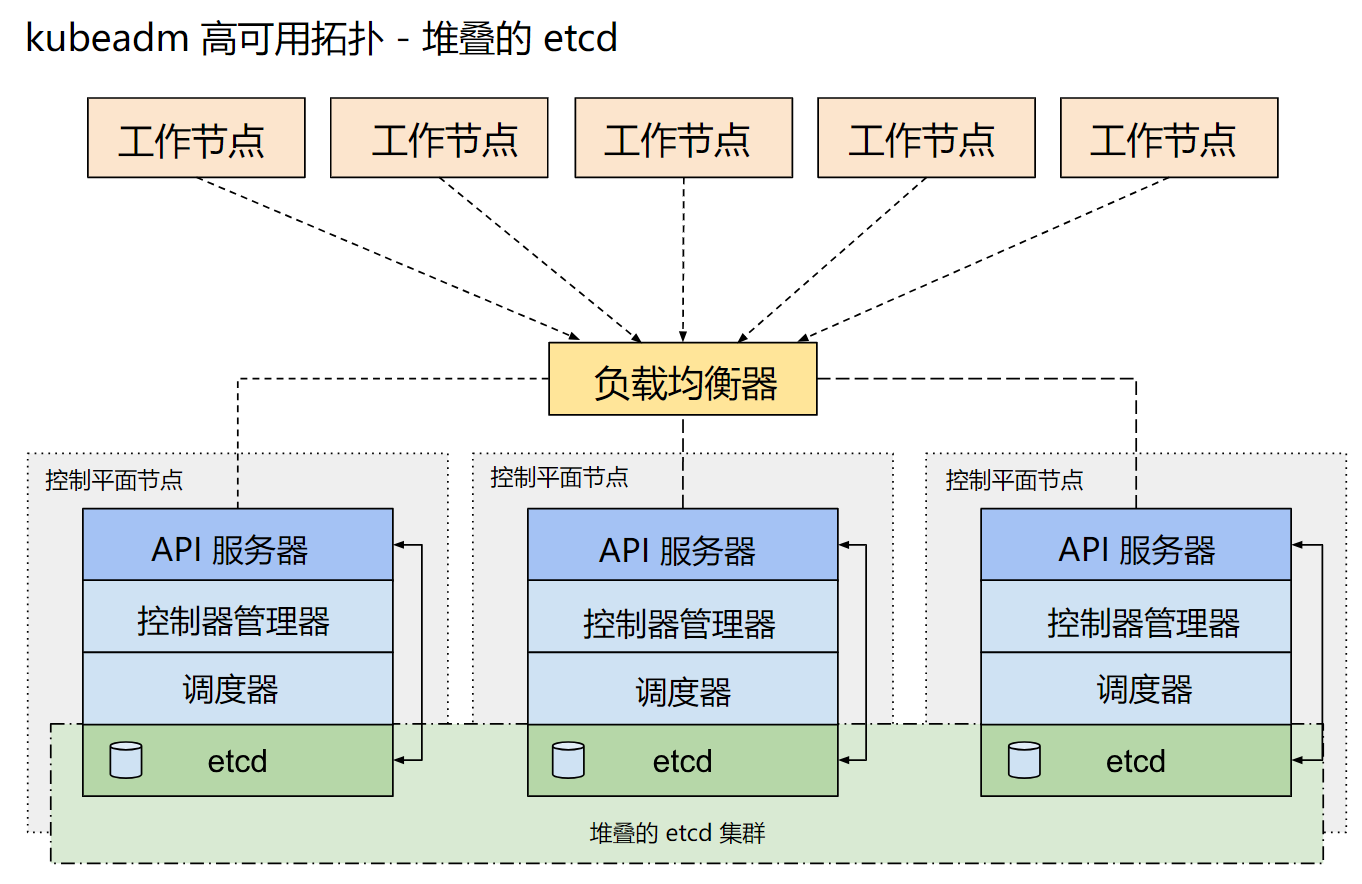
|  |
| --- |
| Bash Linux操作系统： Ubuntu 22.04.1   内核版本：  6.5.0-18-generic   基础软件：  Kubernetes版本：  V1.28.2  容器containerd版本：  github.com/containerd/containerd 1.7.12  网络插件版本：  calico版本：  v3.29.1  git工具：  薅官方脚本更新证书有效期时间  下载：  sudo apt install -y git  ssh服务：  ssh客户端和服务端  客户端自带  服务端下载：  sudo apt install -y openss-server |

**一、多主节点架构概念图**

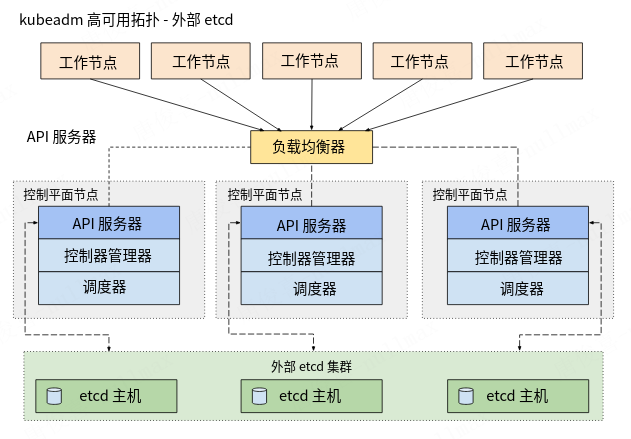
**1.基本架构图(摘至Kubernetes.io)**



**2.高可用etcd集群架构图（摘至kubernetes.io）**



**3.外部高可用ectd集群（摘至kubernetes.io）**



**二、角色规划**

**1.角色分配**

|  |
| --- |
| Bash 角色：  控制节点：  k8-master-01   k8-master-02  k8-master-03  负载均衡：   k8-apiserver-01  k8-apiserver-02  外部etcd存储：   k8-etcd-01  k8-etcd-02  k8-etcd-03  工作节点：   k8-slave-01  k8-slave-02 |

**2.地址解析**

|  |
| --- |
| Bash sudo nano /etc/hosts |

|  |
| --- |
| Bash 192.168.135.136 k8-master-01 #主节点1 192.168.135.135 k8-master-02 #主节点2 192.168.135.145 k8-master-03 #主节点3  192.168.135.143 k8-apiserver-01 #负载均衡器-Master 192.168.135.146 k8-apiserver-02 #负载均衡器-Backup  192.168.135.137 k8-slave-01 #工作节点1 192.168.135.134 k8-slave-02 #工作节点2  192.168.135.149 k8-etcd-01 #高可用etcd集群1 192.168.135.150 k8-etcd-02 #高可用ectd集群2 192.168.135.147 k8-etcd-03 #高可用etcd集群3 |

**三、网络环境准备**

**1.转发 IPv4 并让 iptables 看到桥接流量(所有安装kubernetes的机器)**

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

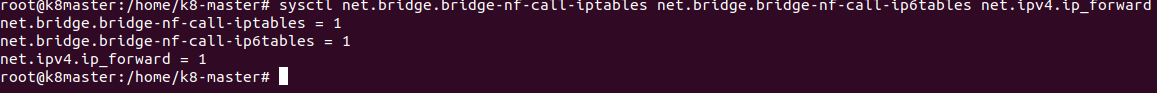
|  |
| --- |
| Bash #加载模块 sudo modprobe overlay sudo modprobe br\_netfilter |

|  |
| --- |
| Bash #设置网络参数 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 #即刻生效 sudo sysctl --system |

**2.确保配置已经启用**

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



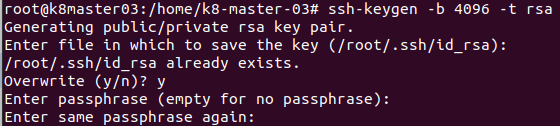
**四、关闭分区**

|  |
| --- |
| 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 sed -i 's@/swapfile@#/swapfile@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。 |

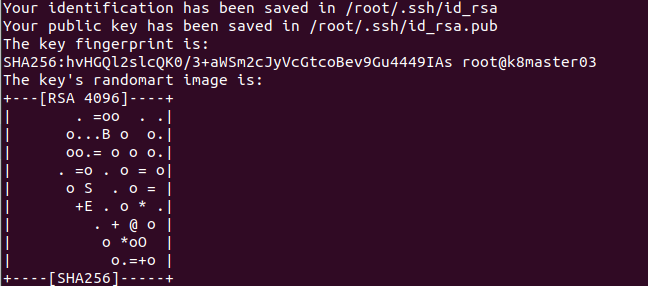
**五、配置ssh传输无密码密钥**

**1.生成秘钥**

|  |
| --- |
| Bash #生成密钥，按提示操作，也可以直接回车以接受默认设置。 #保持默认就行 #生成密钥过程中会让你输入密钥密码，不要输入其他的，回车默认即可，不然仍然需要输入密码 ssh-keygen -b 4096 -t rsa |



|  |
| --- |
| Bash #默认秘钥文件位置 ~ ./ssh/id\_rsa.pub #公钥 ~ ./ssh/id\_rsa #私钥 |



**2.将公钥复制到远程服务器：**

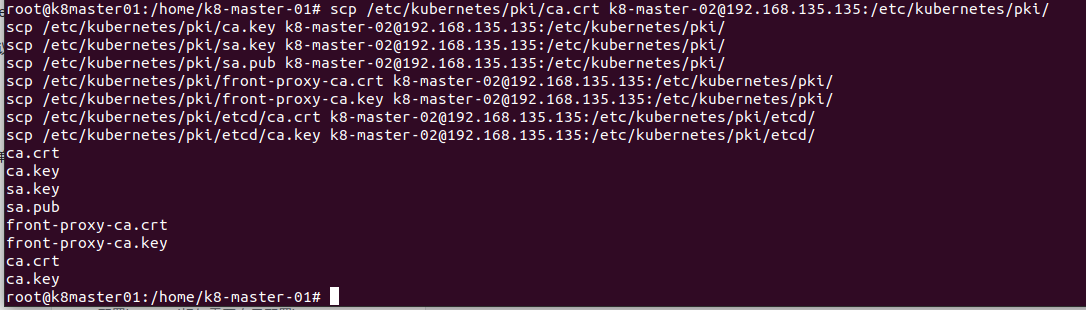
|  |
| --- |
| Bash #复制到远程服务器 #其中user是远程服务器上的用户名，remote\_host是远程服务器的地址。 ssh-copy-id -i ~/.ssh/id\_rsa.pub user@remote\_host |

|  |
| --- |
| Bash #example: #k8-master-03上操作 ssh-copy-id -i ~/.ssh/id\_rsa.pub k8-master-02@192.168.135.135 ssh-copy-id -i ~/.ssh/id\_rsa.pub k8-master-01@192.168.135.136 |

**3.登录远程服务器：**

|  |
| --- |
| Bash #这次登录不应再要求输入密码。  ssh user@remote\_host #以上步骤完成后，你就可以无密钥登录远程服务器了。 |

**4.效果：**



**六、安装配置负载均衡器**

**1.安装**

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

**2.配置keepalived(标红需要自己配置)**

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

|  |
| --- |
| Bash ! /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 100  authentication {  auth\_type PASS  auth\_pass Kubernetes  }  virtual\_ipaddress {  192.168.135.143  }  track\_script {  check\_apiserver  } } |

|  |
| --- |
| Bash ! /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  } } |

|  |
| --- |
| Bash #创建check\_apiserver.sh脚本文件 sudo nano /etc/keepalived/check\_apiserver.sh |

|  |
| --- |
| Bash #!/bin/bash #主备均衡器度需要配置 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" |

**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 # /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}    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    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  # [...] |

**4.启用keepalived和haproxy**

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

**七、初始化kubernetes集群**

1.提前下载、安装和配置好cantainerd,kubernetes,以及calico网络插件配置文件

2.准备好kubernetes和calico等网络插件所需镜像

下载安装请参考基础kubernetes集群搭建过程

**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 |

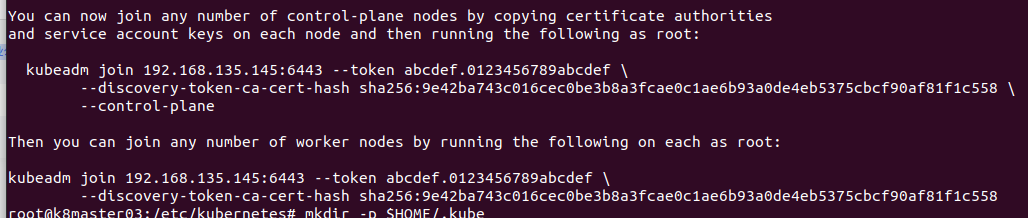
|  |
| --- |
| Bash #目前建议手动 #修改节点名字 nano /etc/kubernetes/kubeadm-init.yaml  #slave节点就改为对应的节点名 name: k8-master  #修改版本 kubernetesVersion: 1.28.2  #修改IP advertiseAddress： 192.168.135.136 #your host  #修改镜像源 #把镜像修改为： imageRepository: registry.aliyuncs.com/google\_containers  #添加podSubnet: podSubnet: 10.244.0.0/16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

|  |
| --- |
| Bash #/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/etcd #存储证书 sudo chown k8-master-n:k8-master-n /etc/kubernetes sudo chown k8-master-n:k8-master-n /etc/kubernetes/pki#修改文件所有者 sudo chown k8-master-n:k8-master-n /etc/kubernetes/pki/etcd |

|  |
| --- |
| Bash #k8-master-n -> k8-master-01 scp /etc/kubernetes/pki/ca.crt k8-master-01@192.168.135.136:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/ca.key k8-master-01@192.168.135.136:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.key k8-master-01@192.168.135.136:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.pub k8-master-01@192.168.135.136:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.crt k8-master-01@192.168.135.136:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.key k8-master-01@192.168.135.136:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/etcd/ca.crt k8-master-01@192.168.135.136:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/pki/etcd/ca.key k8-master-01@192.168.135.136:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/admin.conf k8-master-01@192.168.135.136:/etc/kubernetes/  #k8-master-n -> k8-master-02 scp /etc/kubernetes/pki/ca.crt k8-master-02@192.168.135.135:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/ca.key k8-master-02@192.168.135.135:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.key k8-master-02@192.168.135.135:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.pub k8-master-02@192.168.135.135:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.crt k8-master-02@192.168.135.135:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.key k8-master-02@192.168.135.135:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/etcd/ca.crt k8-master-02@192.168.135.135:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/pki/etcd/ca.key k8-master-02@192.168.135.135:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/admin.conf k8-master-02@192.168.135.135:/etc/kubernetes/  #k8-master-n -> k8-master-03 scp /etc/kubernetes/pki/ca.crt k8-master-03@192.168.135.145:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/ca.key k8-master-03@192.168.135.145:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.key k8-master-03@192.168.135.145:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/sa.pub k8-master-03@192.168.135.145:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.crt k8-master-03@192.168.135.145:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/front-proxy-ca.key k8-master-03@192.168.135.145:/etc/kubernetes/pki/ scp /etc/kubernetes/pki/etcd/ca.crt k8-master-03@192.168.135.145:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/pki/etcd/ca.key k8-master-03@192.168.135.145:/etc/kubernetes/pki/etcd/ scp /etc/kubernetes/admin.conf k8-master-03@192.168.135.145:/etc/kubernetes/ |

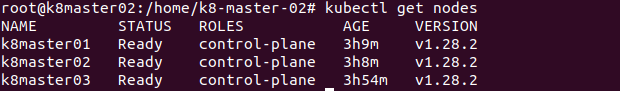
2.加入集群

1.)控制节点加入

|  |
| --- |
| Bash #k8-master-01 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 |

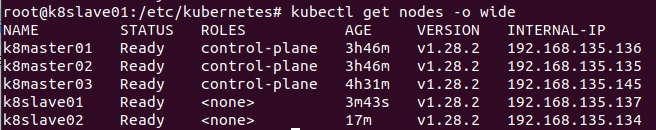
|  |
| --- |
| Bash #k8-master-02 scp k8-master-03@192.168.135.145:/etc/kubernetes/admin.conf /etc/kubernetes/  mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config  #Alternatively, if you are the root user, you can run: export 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 192.168.135.145:6443 --token xxxx.xxxxxx \  --discovery-token-ca-cert-hash sha256: xxxxxxx..xxxx\  --control-plane |



2.)工作节点加入

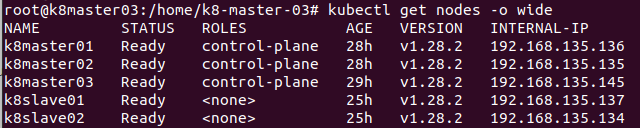
|  |
| --- |
| Bash mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config  #Then you can join any number of worker nodes by running the following on each as root: kubeadm join 192.168.135.145:6443 --token xxxx.xxxxxx \  --discovery-token-ca-cert-hash sha256:xxxxx....xxxxxxx  #example: kubeadm join 192.168.135.145:6443 --token abcdef.0123456789abcdef \  --discovery-token-ca-cert-hash sha256:d77d919dca4a5a9b79362ca70ca2a4d819a0e87b813f56fdffa7df68d865d476 |



**4.删除节点**

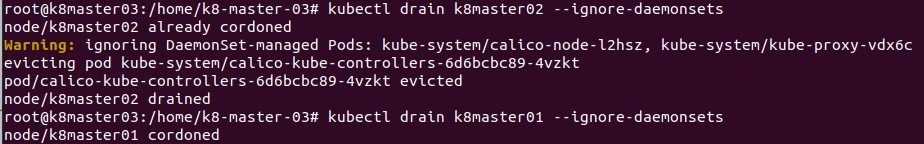
1.)获取所有节点

|  |
| --- |
| 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 |



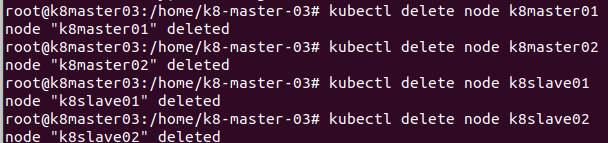
2.）驱除节点

|  |
| --- |
| 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 |

**八、更新证书有效期（依需操作/可选）**

**1.查看目前证书有效期**

|  |
| --- |
| Bash openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text | grep Not |



**2.下载证书**

|  |
| --- |
| Bash git clone https://github.com/yuyicai/update-kube-cert.git cd update-kube-cert chmod 755 update-kubeadm-cert.sh |

**3.更新有效期**

|  |
| --- |
| 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 openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text | grep Not |