步骤：

准备三台linux虚拟机，安装完成后设置ip。或者使用我共享的虚拟机，vmware，复制成3台，分别设置好虚拟机。使用NAT方式。

***虚拟机软件****：windows使用vmware workstation 9，网盘里有共享的安装文件和注册机*

*mac使用vmware fushion，可以搜索下载*

***镜像文件****：可以使用我共享的centos7虚拟机ovf文件导入，需要重新设置ip，copy一下成为3台虚拟机即可。安装完后记得做个快照，防止将系统搞坏。也可以使用我提供的centos7的安装镜像自己重新安装。*

*下载地址：*

*链接:https://pan.baidu.com/s/1-s3b\_4RbthaaSGN81Pf5Tg 密码:iskf*

# 1 centos7虚拟机的配置：

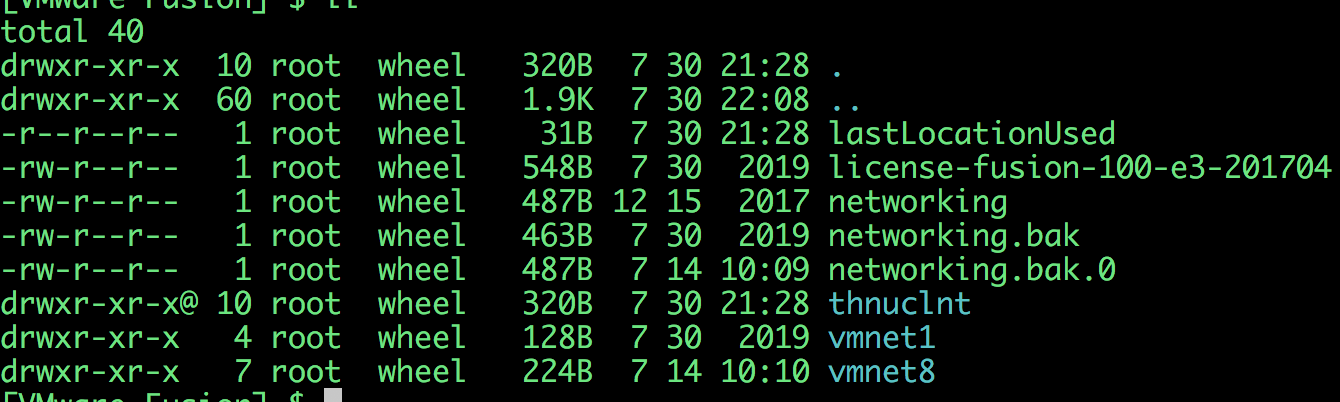
centos7虚拟机的密码：root/123qwe

## 1.1 网络模式设置【vmware fushion mac版】：

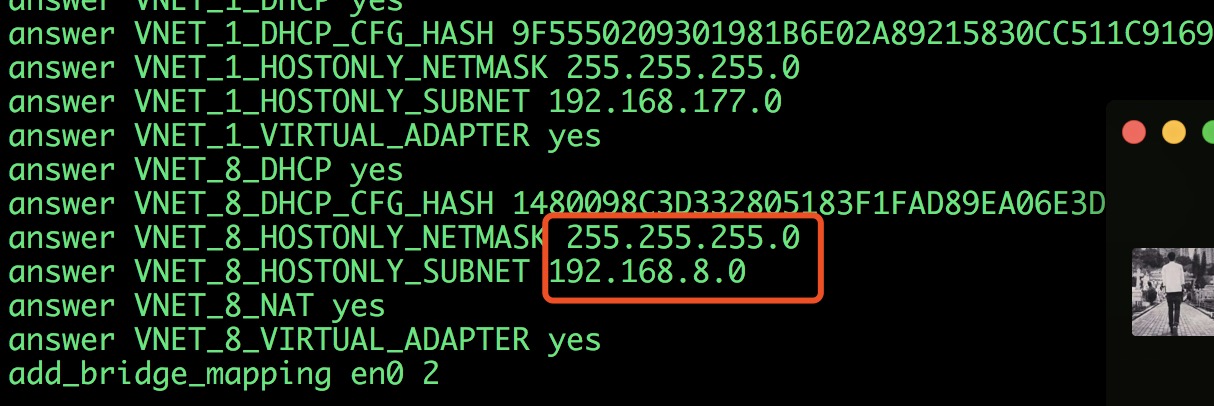
### vmware fusion的网络NAT模式[MAC OS]

**进去以下目录：**

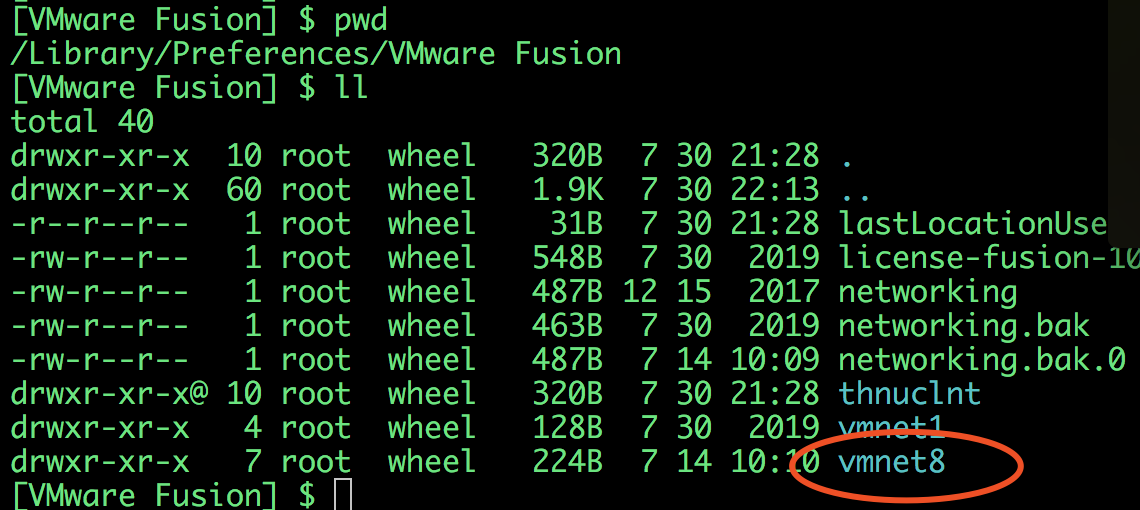
|  |
| --- |
| cd /Library/Preferences/VMware Fusion |



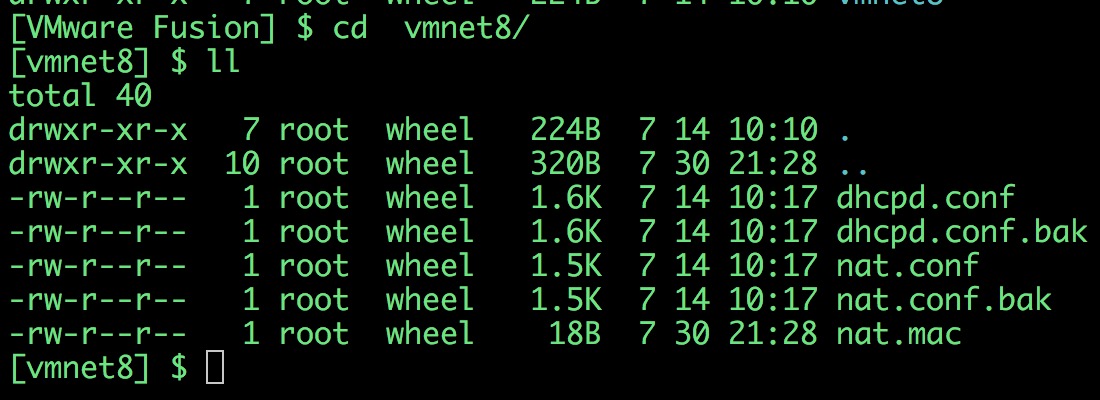
**将vmnet8的配置改为如下,编辑networking：如果没有vmnet8则新增一个：按照以下配置：**



**回到当前目录，进入vmnet8子目录：**



**该目录下有以下文件：**



**首先，修改dhcpd.conf文件内容如下：**

|  |
| --- |
| # Written at: 07/14/2020 10:17:51  allow unknown-clients;  default-lease-time 1800; # default is 30 minutes  max-lease-time 7200; # default is 2 hours  subnet 192.168.8.0 netmask 255.255.255.0 {  range 192.168.8.128 192.168.8.254;  option broadcast-address 192.168.8.255;  option domain-name-servers 192.168.8.2;  option domain-name localdomain;  default-lease-time 1800; # default is 30 minutes  max-lease-time 7200; # default is 2 hours  option netbios-name-servers 192.168.8.2;  option routers 192.168.8.2;  }  host vmnet8 {  hardware ethernet 00:50:56:C0:00:08;  fixed-address 192.168.8.1;  option domain-name-servers 0.0.0.0;  option domain-name "";  option routers 0.0.0.0;  } |

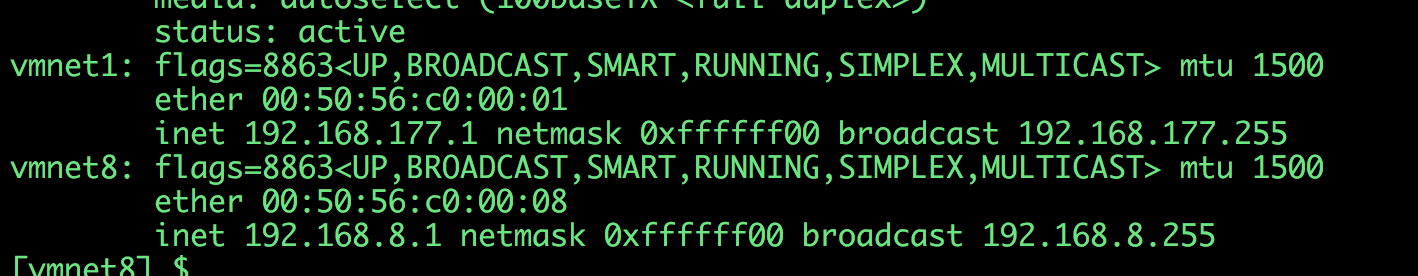
**修改nat.conf如下：**

|  |
| --- |
| # VMware NAT configuration file  # Manual editing of this file is not recommended. Using UI is preferred.  [host]  # NAT gateway address  ip = 192.168.8.2  netmask = 255.255.255.0  # VMnet device if not specified on command line  device = vmnet8  # Allow PORT/EPRT FTP commands (they need incoming TCP stream ...)  activeFTP = 1  # Allows the source to have any OUI. Turn this on if you change the OUI  # in the MAC address of your virtual machines.  allowAnyOUI = 1  # Controls if (TCP) connections should be reset when the adapter they are  # bound to goes down  resetConnectionOnLinkDown = 1  # Controls if (TCP) connection should be reset when guest packet's destination  # is NAT's IP address  resetConnectionOnDestLocalHost = 1  # Controls if enable nat ipv6  natIp6Enable = 0  # Controls if enable nat ipv6  natIp6Prefix = fd15:4ba5:5a2b:1008::/64  [tcp]  # Value of timeout in TCP TIME\_WAIT state, in seconds  timeWaitTimeout = 30  [udp]  # Timeout in seconds. Dynamically-created UDP mappings will purged if  # idle for this duration of time 0 = no timeout, default = 60; real  # value might be up to 100% longer  timeout = 60  [netbios]  # Timeout for NBNS queries.  nbnsTimeout = 2  # Number of retries for each NBNS query.  nbnsRetries = 3  # Timeout for NBDS queries.  nbnsTimeout = 2  # Number of retries for each NBNS query.  nbnsRetries = 3  # Timeout for NBDS queries.  nbdsTimeout = 3  [incomingtcp]  # Use these with care - anyone can enter into your VM through these...  # The format and example are as follows:  #<external port number> = <VM's IP address>:<VM's port number>  #8080 = 172.16.3.128:80  [incomingudp]  # UDP port forwarding example  #6000 = 172.16.3.0:6001 |

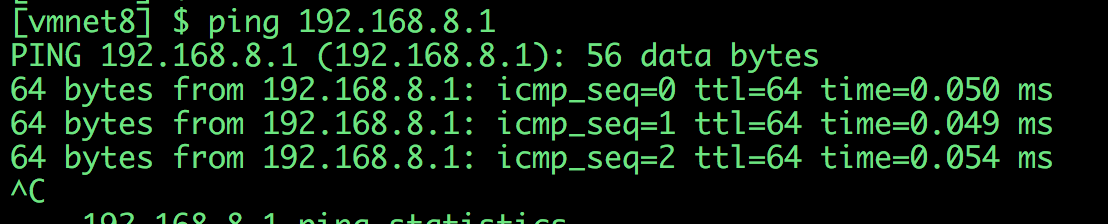
**然后重启虚拟网络：**

|  |
| --- |
| sudo /Applications/VMware\ Fusion.app/Contents/Library/vmnet-cli --stop  sudo /Applications/VMware\ Fusion.app/Contents/Library/vmnet-cli --start |

**查看一下ifconfig：**



**查看vmbet8是否有ip且未192.168.8.1，且能ping通说明配置成功：**



然后在虚拟机菜单中将网络设置为NAT网络即可：



# 2 使用现有集群镜像

使用构建好的k8s集群

|  |
| --- |
| 链接:https://pan.baidu.com/s/1VIAfbpVGPycDF4rKX88CpA 密码:kktl |

## 2.1 步骤1：配置网络

确保步骤1操作正确，能连上网，在导入并且启动成功的虚拟机里面ping网关192.168.8.1和[www.baidu.com](http://www.baidu.com)，如果分别能连通说明网络配置ok。

## 2.2 步骤2：重启虚拟机

倒入成功后，将全部虚拟机重启一遍。

重启后，

执行以下命令确保kubelet和docker启动成功：

|  |
| --- |
| system restart docker  systemctl start kubelet |

执行kubectl get node，确保node都处于ready状态，如果没有处于ready状态，则：

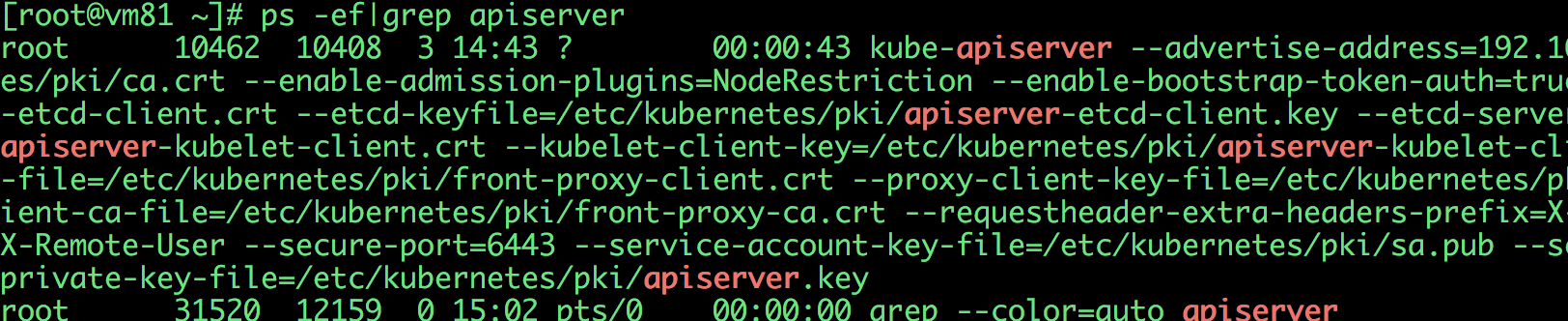
尝试：

1. 重启kubelet和docker
   1. systemctl restart kubelet
   2. system restart docker
2. 重启虚拟机

## 2.3 检查方式

### 确保apiserver启动：

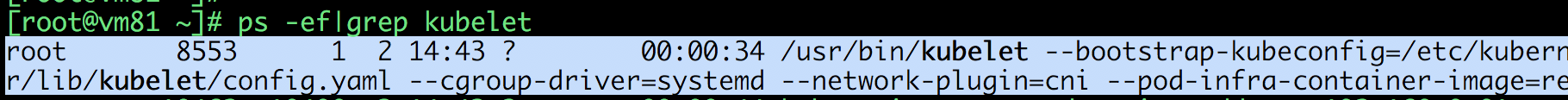
apiserver作为总线，所有的pod和kubelet都要和其打交道，需要确保apiserver启动：



apiserver是部署在k8s pod中的，如果apiserver没有启动，原因可能有：

1. kubelet没有启动
2. kubelet启动了，但是pod没有启动

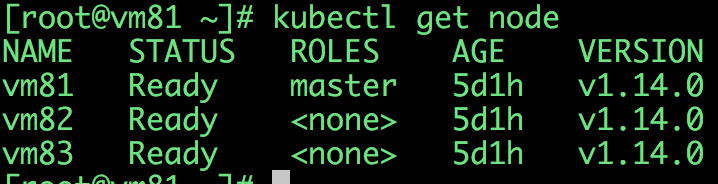
### 确保kubelet启动：



如果没有启动：执行

**systemctl restart kubelet**

确保k8s的node处于ready状态：



如果node没有处于ready状态，检查pod：



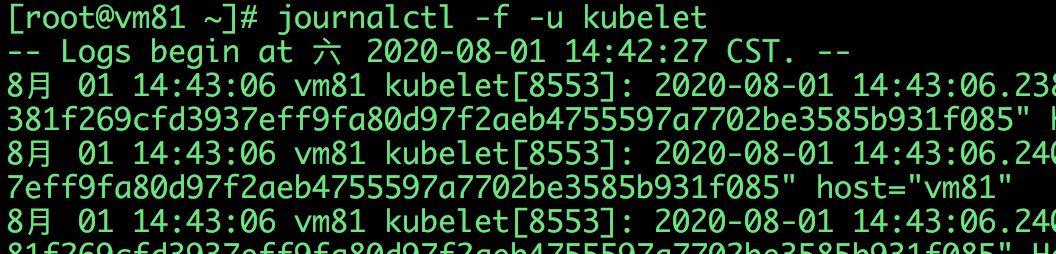
### 检查pod

如果有的pod没有启动，检查pod的状态：

kubectl describe pod pod-XXXXXXX

### 查看日志

如果依然有问题，查看kubelet的日志，看问题针对性的解决：



# 3 现有镜像重新安装

如果依然没有搞定，集群还是无法启动，则可以考虑重新安装，重新安装的话从kubeadm reset开始。

## 3.1 步骤1：reset

在**集群的每个节点上**执行一次：

|  |
| --- |
| kubeadm reset |

注意清理iptables：

|  |
| --- |
| iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X |

## 3.2 步骤2：init

参考4.5-4.7节的步骤

# 4 自己安装k8s【如果想自己安装的话】

使用centos的干净镜像，镜像及配置文件下载地址：

|  |
| --- |
| 链接:https://pan.baidu.com/s/14dCRt15Ozg9K1-DzHoKaxw 密码:o73e |

将镜像复制三份，分别倒入vmware，配置为不同的ip，ip配置方法参考步骤1.

## 4.1 hostname配置

比如当前我们的三台虚拟机的配置信息如下

|  |  |
| --- | --- |
| Hostname | ip |
| vm81 | 192.168.8.81 |
| vm82 | 192.168.8.82 |
| vm83 | 192.168.8.83 |

1. 为三台虚拟机设置ip hostname：以master为例：（master&worker）

|  |
| --- |
| *#修改当前的主机名，比如master/slave1/slave2*  hostnamectl set-hostname vm81  *#修改host文件*  echo **192.168.8.81** vm81 >>/etc/hosts  echo **192.168.8.82** vm82 >>/etc/hosts  echo **192.168.8.83** vm83 >>/etc/hosts |

## 4.2 更新系统配置（master&worker）

|  |
| --- |
| #安装依赖  yum -y remove kube\*  yum -y update  yum install -y conntrack ipvsadm ipset jq sysstat curl iptables libseccomp  #关闭防火墙  systemctl stop firewalld && systemctl disable firewalld  #重置iptables  iptables -F && iptables -X && iptables -F -t nat && iptables -X -t nat && iptables -P FORWARD ACCEPT  #关闭swap  swapoff -a  sed -i '/swap/s/^\(.\*\)$/#\1/g' /etc/fstab  #关闭selinux  setenforce 0  #关闭dnsmasq  service dnsmasq stop && systemctl disable dnsmasq  #配置文件  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  vm.swappiness=0  vm.overcommit\_memory=1  vm.panic\_on\_oom=0  fs.inotify.max\_user\_watches=89100  EOF  chmod 755 /etc/sysctl.d/kubernetes.conf  modprobe br\_netfilter  #加载  sysctl -p /etc/sysctl.d/kubernetes.conf |

## 4.3 安装docker（master&worker）

|  |
| --- |
| sudo yum install -y yum-utils wgt  sudo yum-config-manager \  --add-repo \  https://download.docker.com/linux/centos/docker-ce.repo  #安装docker  sudo yum install docker-ce docker-ce-cli containerd.io  #修改cgroup  cat >>/etc/docker/daemon.json<<EOF  {  "exec-opts": ["native.cgroupdriver=systemd"]  }  EOF  #启动docker  sudo systemctl enable docker.service&&systemctl start docker  **#修改yum源(可选)：yum报404时**  mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak&&  wget -O CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo&& yum clean all&& yum makecache |

## 4.4 安装kubernetes（master&worker）

|  |
| --- |
| 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  yum install -y kubeadm-1.14.0-0 kubelet-1.14.0-0 kubectl-1.14.0-0 kubernetes-cni-0.7.5-0.x86\_64 --disableexcludes=kubernetes  #启动kubelet  systemctl enable kubelet && systemctl start kubelet |

## 4.5 在master上执行初始化（仅master）

|  |
| --- |
| #重置一下  kubeadm reset  #自定义config安装kube  cat <<EOF>kubeadm-config.yaml  apiVersion: kubeadm.k8s.io/v1beta1  kind: ClusterConfiguration  kubernetesVersion: v1.14.0  #第一个master节点的ip  controlPlaneEndpoint: "**192.168.8.81:6443**"  networking:  podSubnet: "172.16.0.0/16"  imageRepository: registry.cn-beijing.aliyuncs.com/xianshuangzhang  EOF  #执行init命令  kubeadm init --config=kubeadm-config.yaml --experimental-upload-certs  #观察打印出的命令  #如果有问题，kubeadm reset  一下，将iptables和ipvs重置一下  *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 each as root:*  *kubeadm join 192.168.1.201:6443 --token zlfmmm.a41tyorwikg336fx \*  *--discovery-token-ca-cert-hash sha256:44f5622e441e88e172a103f084ea150e62b2a5cdd11cb6fb65f314a0ac92fb9a \*  *--experimental-control-plane --certificate-key bb7b737d193d043102123af2d50ef7ffdbdc74b76fa4a9390853c2a54c019add*  *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 --experimental-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 192.168.1.201:6443 --token zlfmmm.a41tyorwikg336fx \*  *--discovery-token-ca-cert-hash sha256:44f5622e441e88e172a103f084ea150e62b2a5cdd11cb6fb65f314a0ac92fb9a*  #master安装完毕 |

## 4.6 在worker上执行kubeadm的join指令（仅worker）

|  |
| --- |
| kubeadm join 192.168.1.201:6443 --token u4amfg.abg0ljzx4oauygvi \  --discovery-token-ca-cert-hash sha256:493ee8da1180e7e1b770d510f9b25162a39b90e3792c9e94c2fe00ee37954efa |

如果想增加多个master，则执行上面的join命令：

|  |
| --- |
| *kubeadm join 192.168.1.201:6443 --token zlfmmm.a41tyorwikg336fx \*  *--discovery-token-ca-cert-hash sha256:44f5622e441e88e172a103f084ea150e62b2a5cdd11cb6fb65f314a0ac92fb9a \*  *--experimental-control-plane --certificate-key bb7b737d193d043102123af2d50ef7ffdbdc74b76fa4a9390853c2a54c019add* |

请注意保存join命令，未来如果集群需要扩容，则需要该命令。

## 4.7 安装addons插件（安装目录下的三个yaml文件）

|  |
| --- |
| kubectl apply -f calico-rbac-kdd.yaml  kubectl apply -f calico.yaml  kubectl apply -f dashboard-all.yaml |

kubectl get node -o wide：

|  |
| --- |
| NAMESPACE NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES  kube-system calico-node-2nq5h 2/2 Running 0 9m27s 192.168.1.212 slave2 <none> <none>  kube-system calico-node-t77jj 2/2 Running 0 9m27s 192.168.1.201 master <none> <none>  kube-system calico-typha-666749994b-jzfl9 1/1 Running 0 9m27s 192.168.1.212 slave2 <none> <none>  kube-system coredns-78498d8ff6-4nq6x 1/1 Running 0 21m 172.16.0.2 master <none> <none>  kube-system coredns-78498d8ff6-gc4gw 1/1 Running 0 21m 172.16.0.3 master <none> <none>  kube-system etcd-master 1/1 Running 0 20m 192.168.1.201 master <none> <none>  kube-system kube-apiserver-master 1/1 Running 0 20m 192.168.1.201 master <none> <none>  kube-system kube-controller-manager-master 1/1 Running 0 20m 192.168.1.201 master <none> <none>  kube-system kube-proxy-9hb6s 1/1 Running 0 18m 192.168.1.212 slave2 <none> <none>  kube-system kube-proxy-qvd48 1/1 Running 0 21m 192.168.1.201 master <none> <none>  kube-system kube-scheduler-master 1/1 Running 0 20m 192.168.1.201 master <none> <none> |

安装完毕，如果init时有问题，则重置一下kubeadm重新init：

kubeadm reset

注意执行以下打印出的命令：

|  |
| --- |
| 1. **rm -rf ~/.kube systemctl stop kubelet** 2. **systemctl stop docker** 3. **iptables --flush** 4. **iptables -tnat --flush** 5. **systemctl start kubelet** 6. **systemctl start docker** |