## Kubernetes练习系统使用手册

#### 操作系统

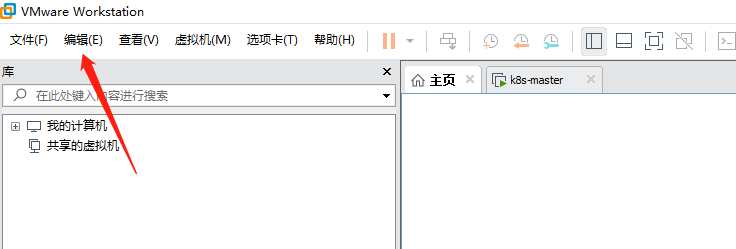
CentOS Linux 7

Red Hat 4.8.5-16

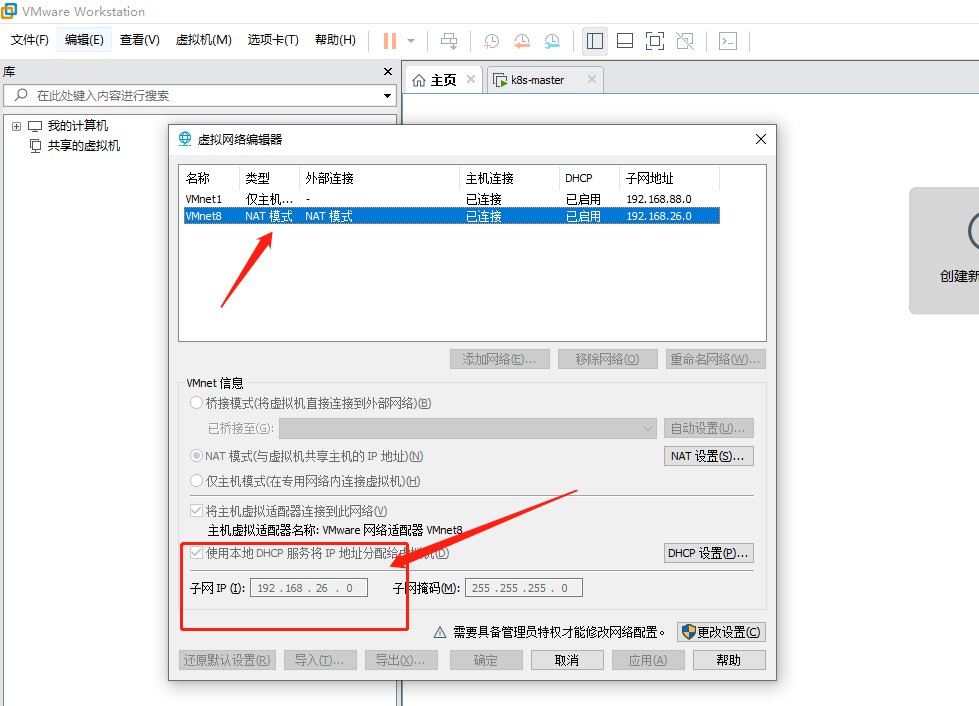
#### 虚拟机设置

###### 1、虚拟机网络配置

在VMware Workstation中选择编辑 -> 虚拟网络编辑器。



进入虚拟网络编辑器后，选择NAT模式的网络VMnet8，修改它的子网IP为192.168.26.0（因为系统内已经写好了一个shell脚本把网段提前写死了，避免出现其他意外报错，建议统一使用192.168.26网段，\*需要修改网段的同学可以进入虚拟机后自行修改根目录下的set.sh脚本内容）修改完成后，点击应用就可以了，不要再修改任何主机内网络相关的配置。



###### 2、VM兼容性问题

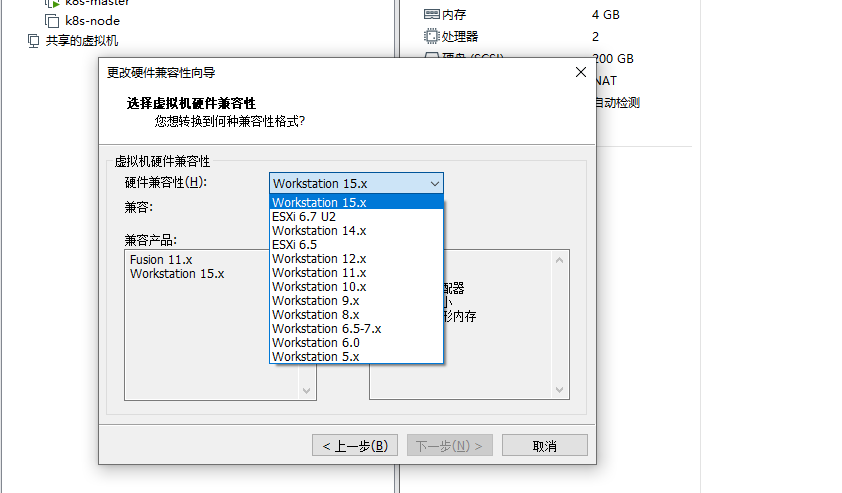
K8s-vms节点主机是采用虚拟机镜像的形式提供给大家使用，该虚拟机镜像在VMware Workstation 15.x打包的，所以考虑到VMware Workstation的版本兼容性，在需要克隆虚拟机时需要提前配置一下VMware Workstation版本兼容性。

操作：右键一台虚拟机 -> 管理 -> 更改硬件兼容性（H）

比如需要克隆一个node节点主机

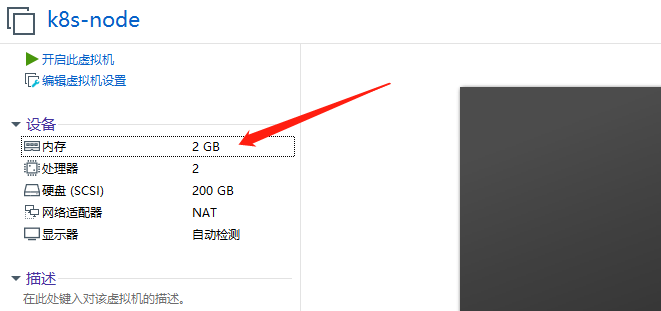


点击下一步，选择自己安装的VMware Workstation版本即可，如果已经是15.x的版本了，那就不需要设置版本兼容性，直接克隆使用即可



###### 3、虚拟机内存配置

考虑到每个人的应用情况不同，以及PC的内存容量也大小各异。所以推荐在运行虚拟机之前统一把虚拟机内存设置成2G，在所有节点主机内存都设置成2G的情况下，同时运行三个节点应该是没问题的（正常情况运行一台master\两台work node就可以满足测试使用，也不会影响PC的其他应用正常使用）



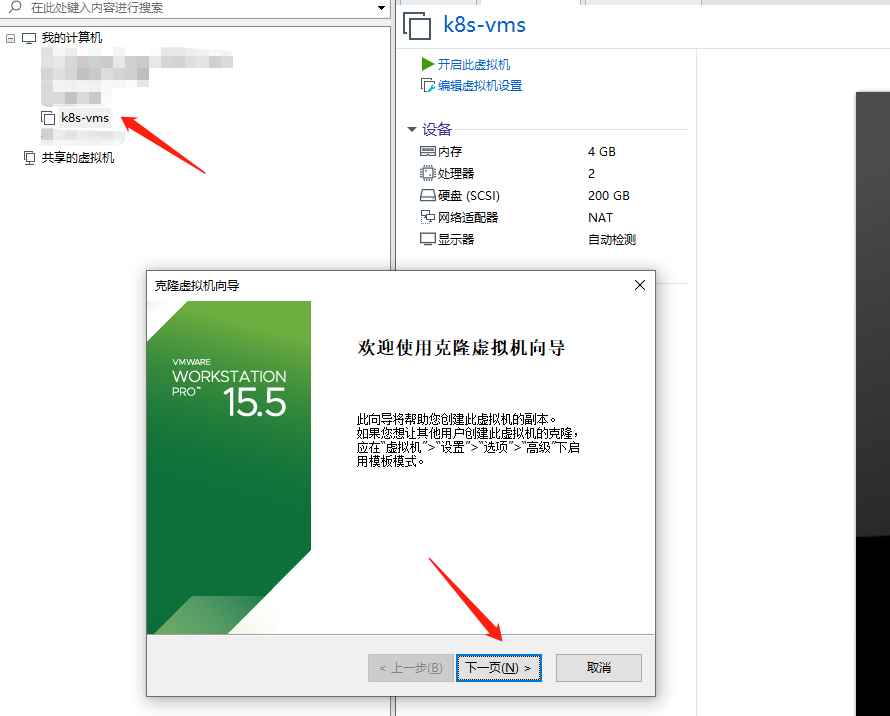
#### kubernetes部署

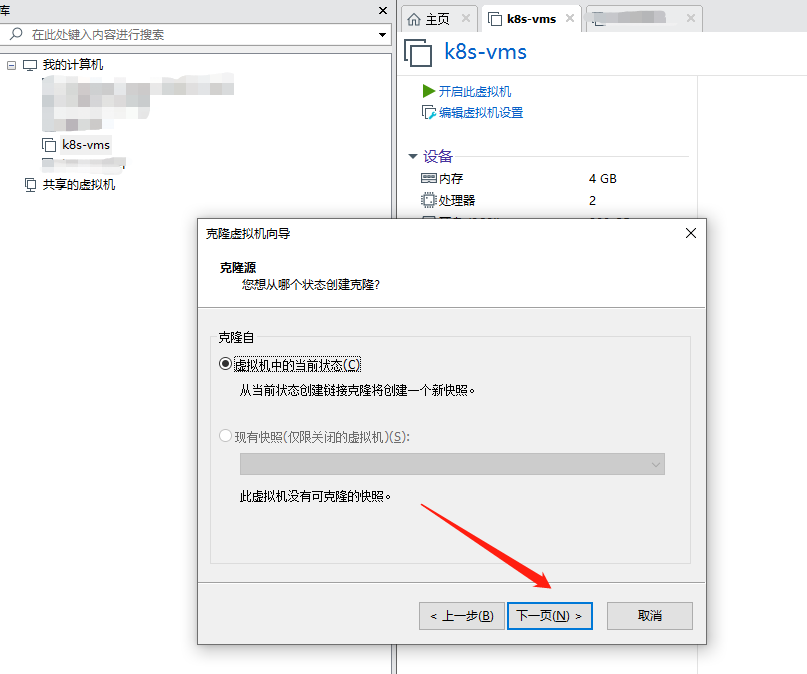
###### 节点主机准备

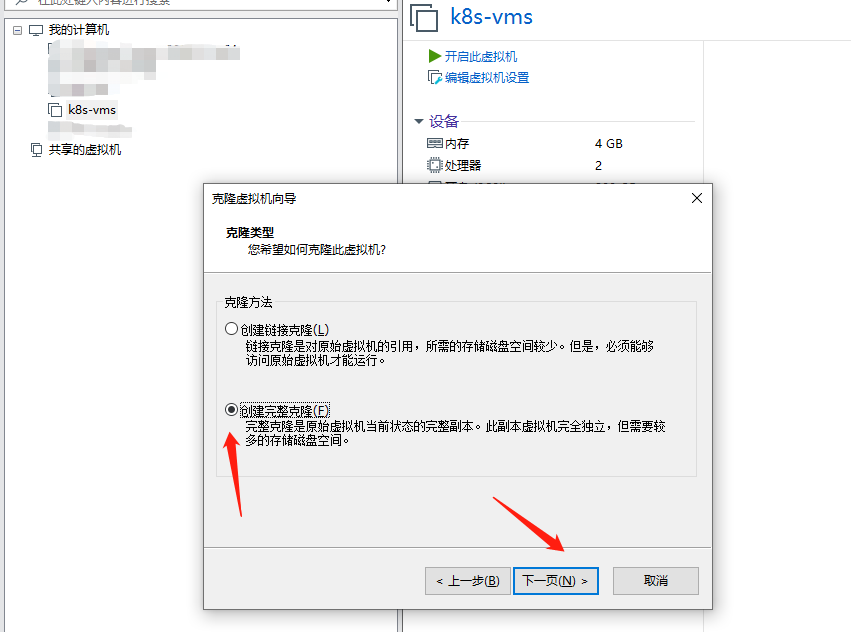
1.1克隆节点主机

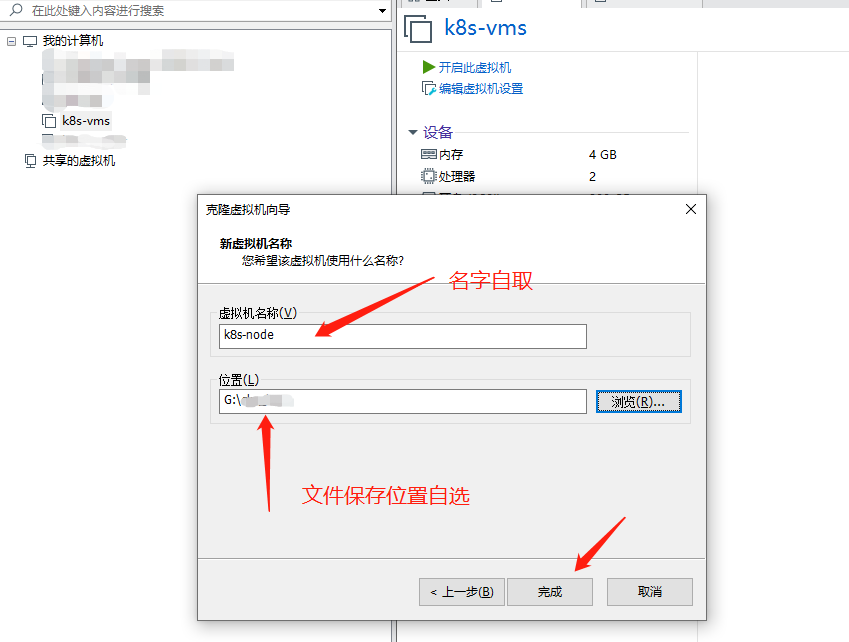
以k8s-vms为主体克隆出一个matser节点和一个node节点，在这一步操作中不需要运行k8s-vms虚拟机。

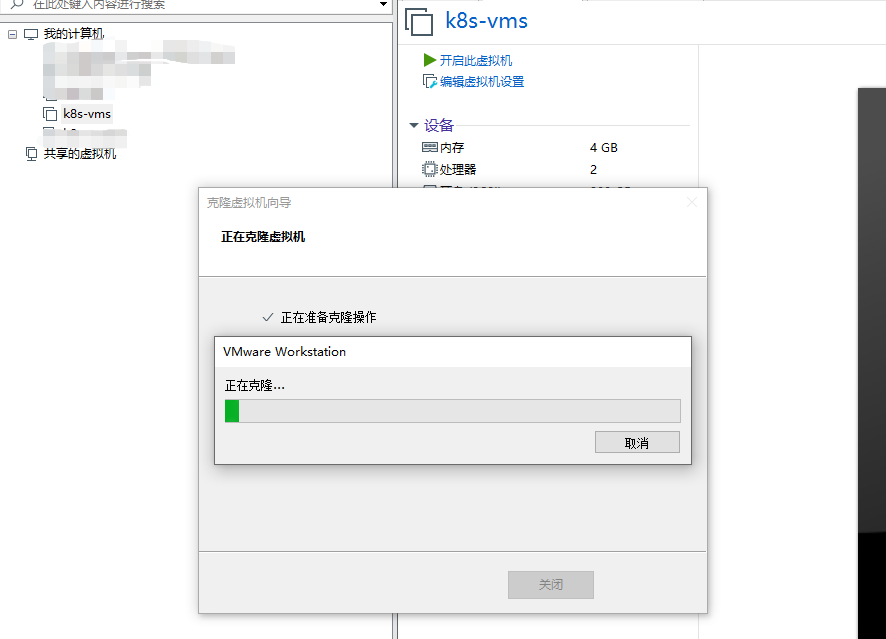
操作：右键k8s-vms -> 管理 -> 克隆





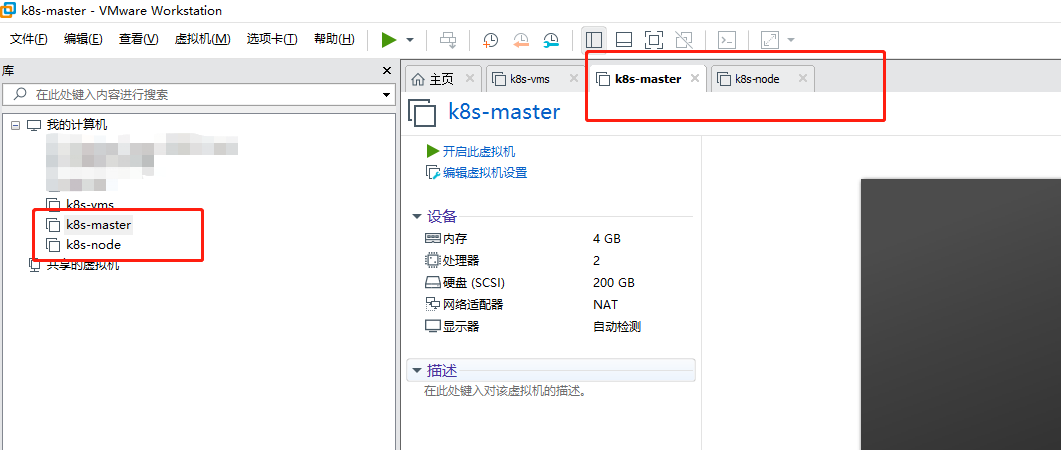




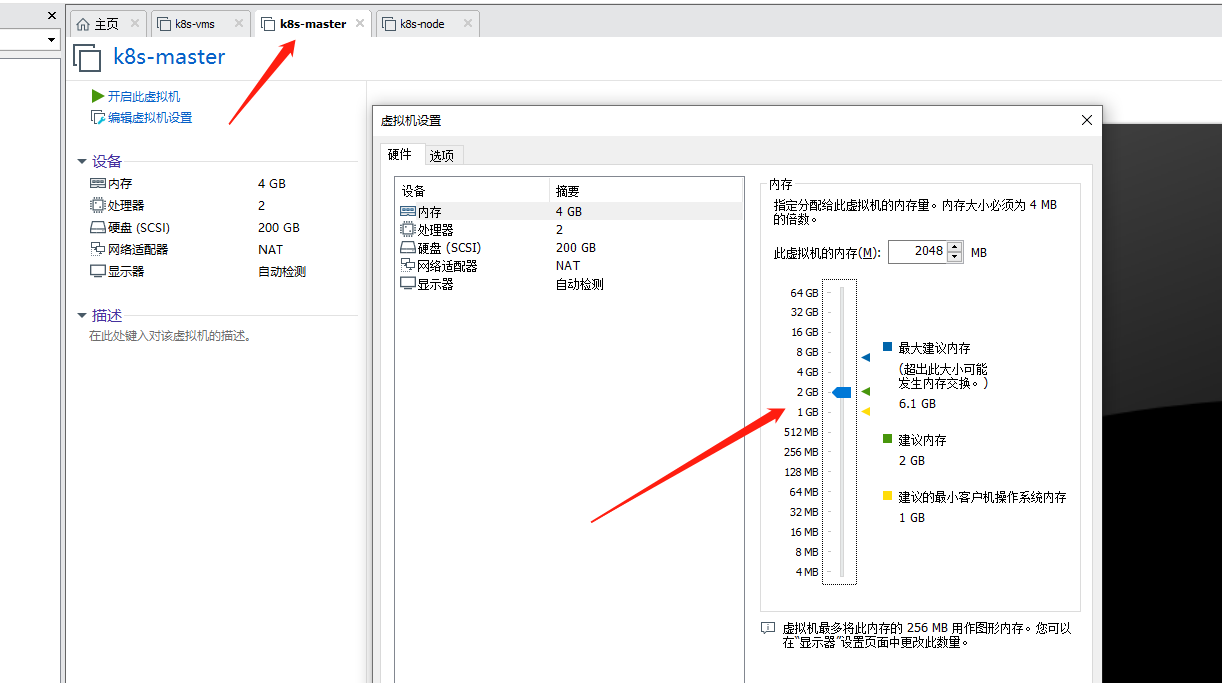


1.2主机内存设置

克隆出了两台节点主机后，分别修改内存大小为2G

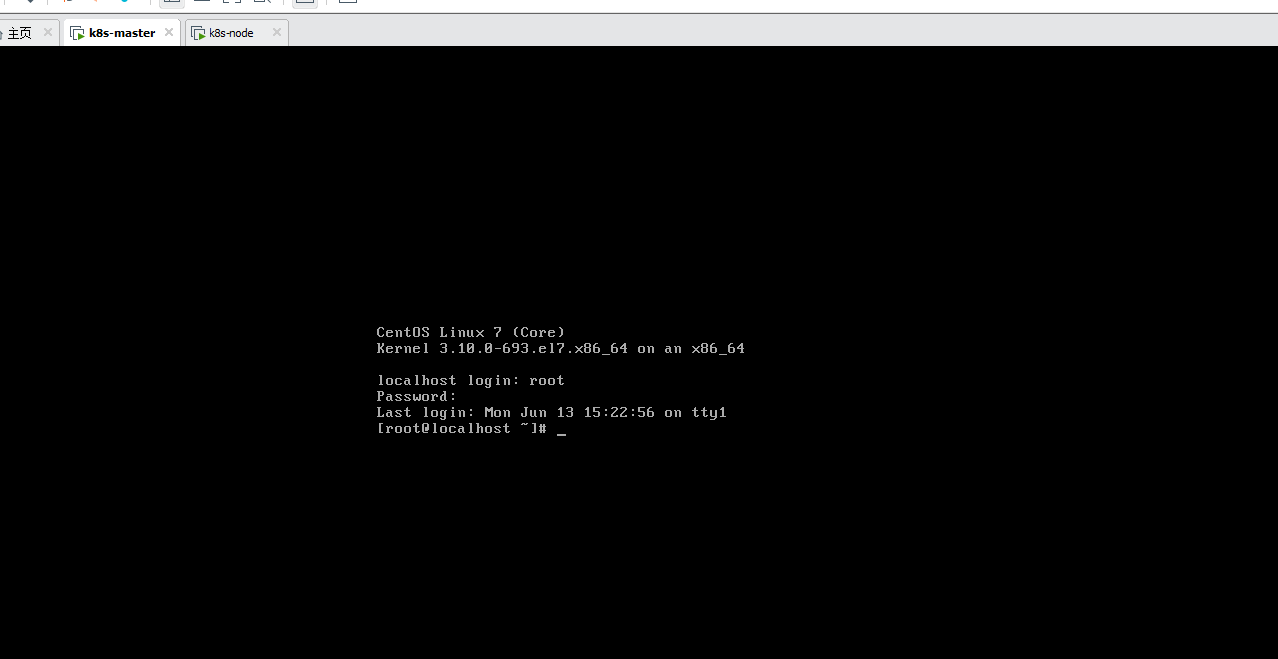


操作：右键节点主机 -> 设置 -> 内存 -> 拉取到2G -> 确定



1.3主机网络设置

分别运行两台节点主机，进入登入界面，主机账号密码：root / redhat

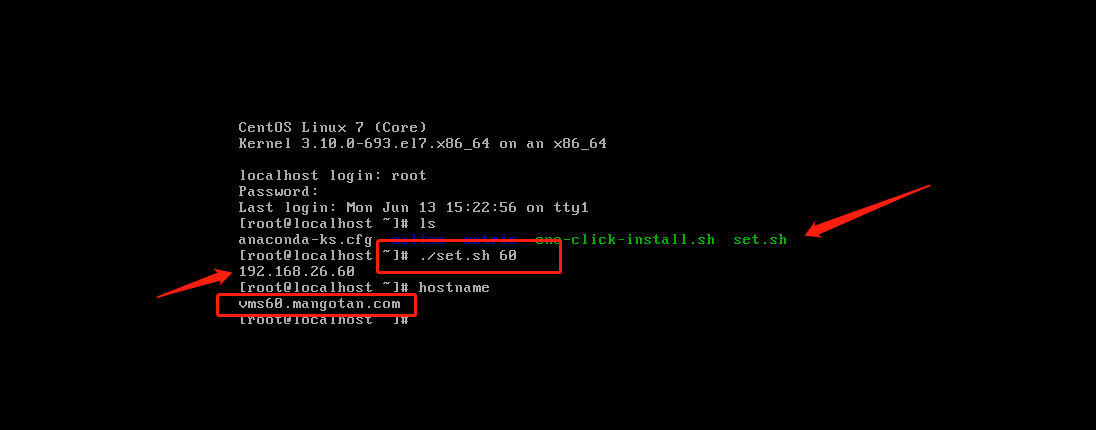


分别进入到系统后，设置网络，根目录下有一个shell脚本set.sh

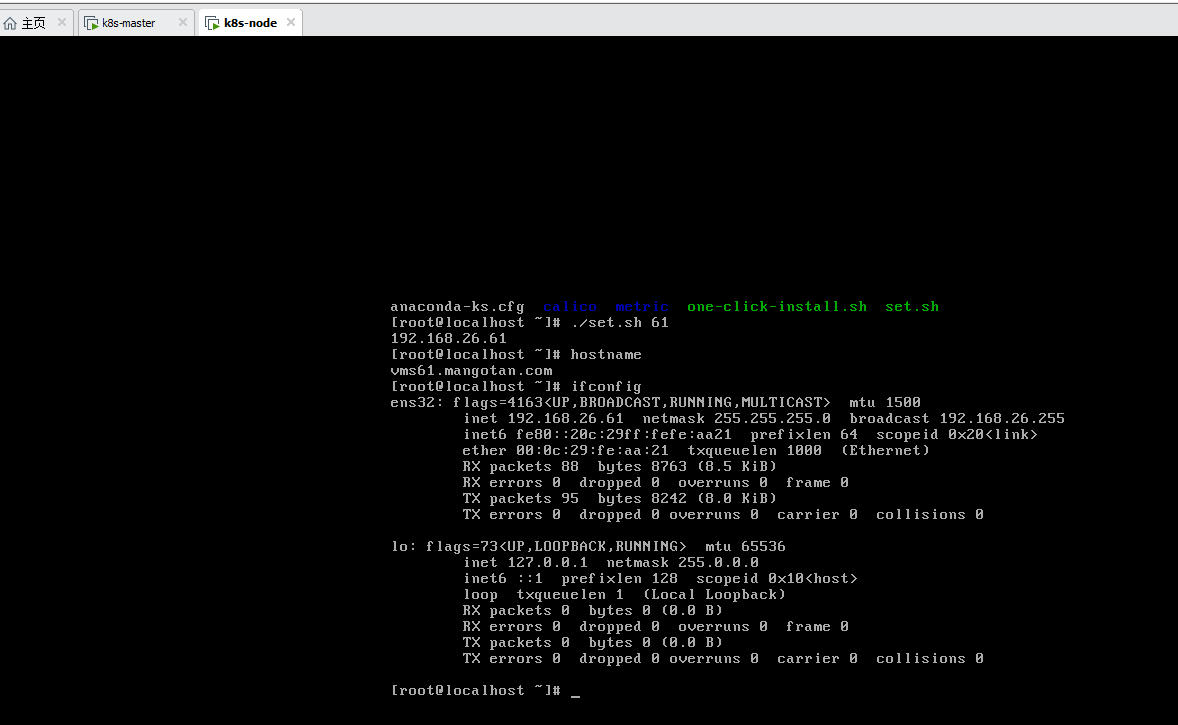
在master节点主机设置网络只需要执行.sh脚本，输入自己想设置的C段数值即可，比如设置网络IP为192.168.26.60的操作如下所示

执行命令：./set.sh 60

回车，等待一会儿就设置好了当前主机的IP地址，以及生成一个hostname



同理，设置node节点



做好以上工作后，节点主机就已经准备好了，可以利用远程连接工具开始进行kubernetes的部署。

示例环境

Master：192.168.26.60

Node：192.168.26.61

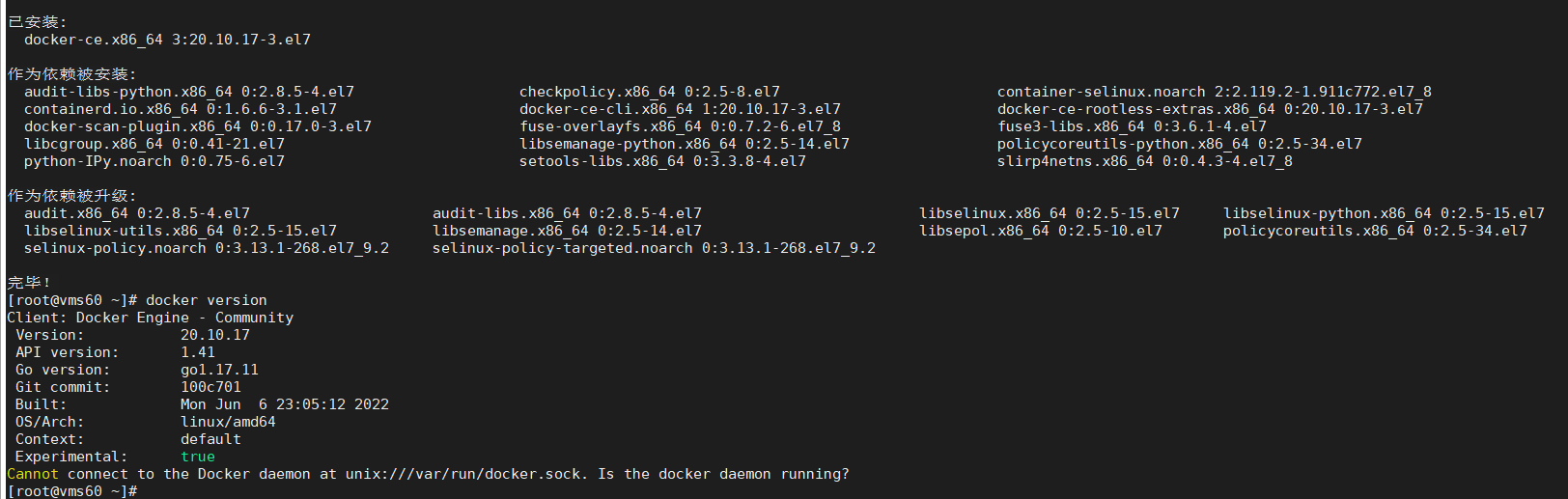
###### Kubernetes部署

2.1安装docker

以下操作按顺序分别在master和node节点主机上进行

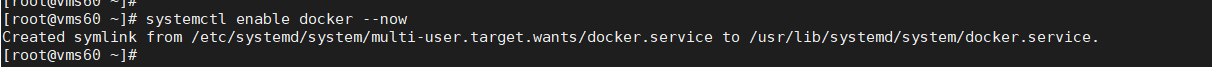
# 安装docker

yum install docker-ce -y



# 启动docker，自动加载一些配置文件

systemctl enable docker --now



# 修改参数，如果 exec-opts 不修改默认使用cgroupfs

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

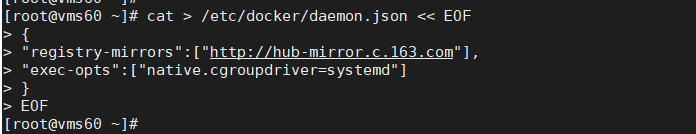
{

"registry-mirrors":["[http://hub-mirror.c.163.com](http://hub-mirror.c.163.com" \t "_blank)"],

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

}

EOF



#再重启一下docker

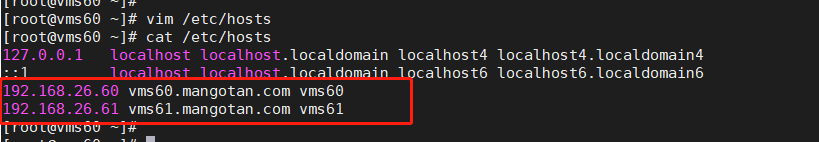
systemctl daemon-reload ; systemctl restart docker

2.2安装kubectl/kubeadm/kubelet

以下操作按顺序分别在master和node节点主机上进行

# 添加节点host，保证两台主机的hosts文件都是一样的内容，如果还有其他node节点存在，需要自行在末尾添加该node节点的host

vim /etc/hosts



# 关闭防火墙：

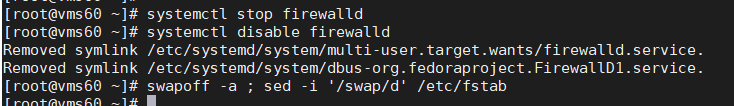
systemctl stop firewalld

# 禁止防火墙开机启动

systemctl disable firewalld

# 关闭swap

swapoff -a ; sed -i '/swap/d' /etc/fstab



# 设置k8s相关属性

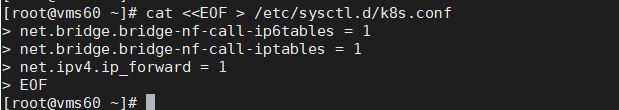
cat <<EOF > /etc/sysctl.d/k8s.conf

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

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

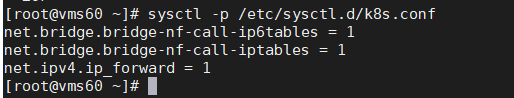
net.ipv4.ip\_forward = 1

EOF



# 再执行以下命令查看属性是否配置成功

sysctl -p /etc/sysctl.d/k8s.conf



# 配置阿里源安装Kubernetes

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

[kubernetes]

name=Kubernetes

baseurl=[https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/](https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/" \t "_blank)

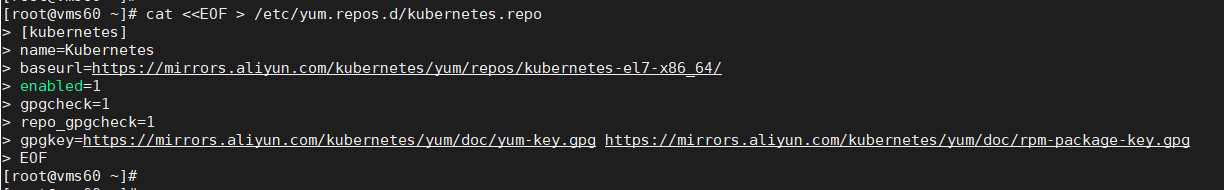
enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=[https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg](https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg" \t "_blank) [https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg](https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg" \t "_blank)

EOF



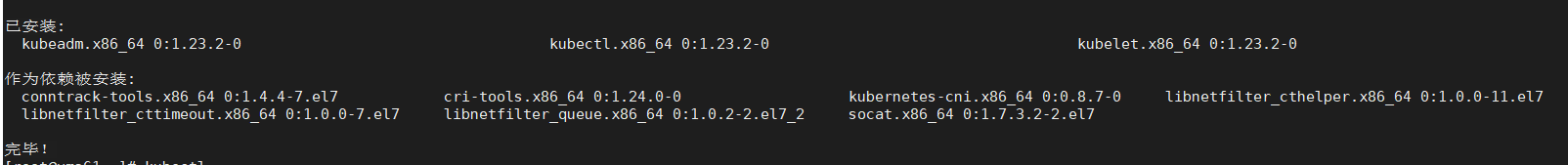
# 关闭 SELinux

setenforce 0



# 安装kubelet/kubeadm/kubectl，由于官网未开放同步方式, 可能会有索引gpg检查失败的情况，用如下命令安装(这里以安装kubernetes版本1.23.2为例，需要安装其他版本的可以自行修改)

yum install -y --nogpgcheck kubelet-1.23.2-0 kubeadm-1.23.2-0 kubectl-1.23.2-0



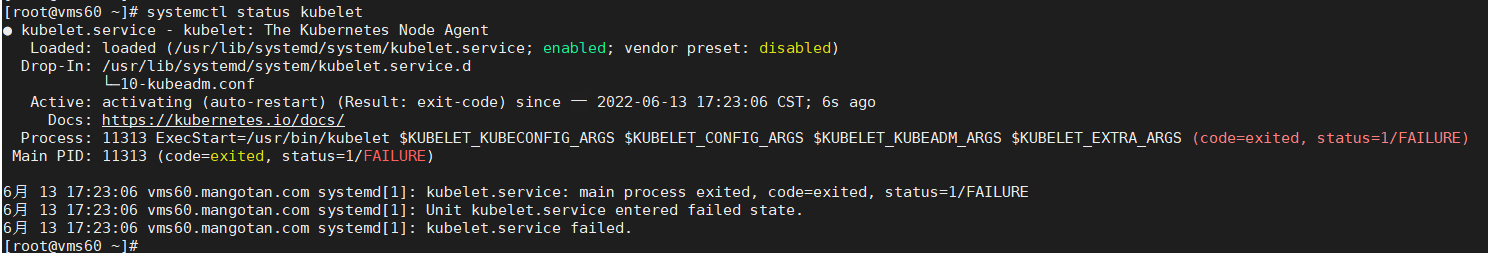
# 启动服务

systemctl enable kubelet && systemctl start kubelet



# 启动后可以查看一下kubernetes的状态

systemctl status kubelet



2.3初始化集群

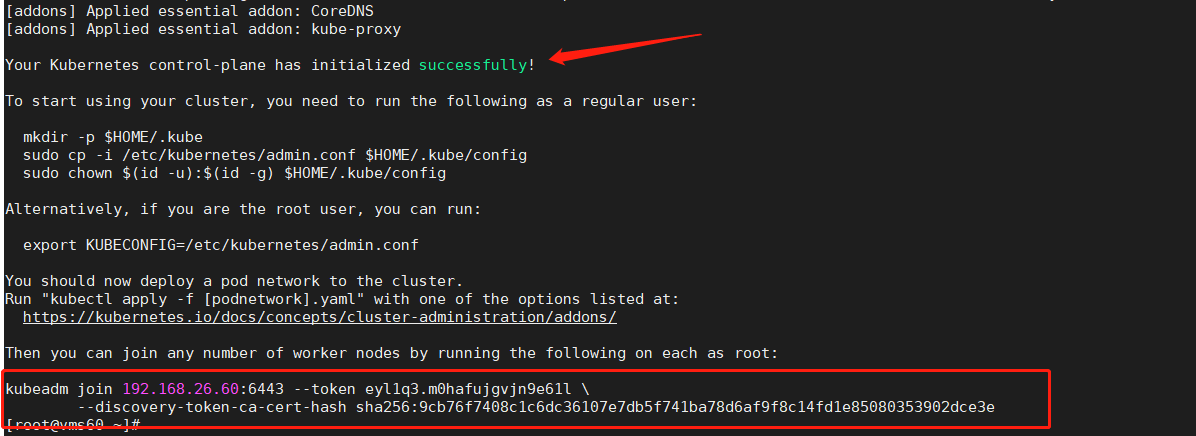
以下操作按顺序只在master节点主机上进行

# 初始化集群（如果是安装的其他kubernetes版本以及修改过master节点的IP地址，那么只需要修改版本号以及两个IP地址和主机名就可以了）

kubeadm init --kubernetes-version=v1.23.2 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=192.168.26.60 --apiserver-cert-extra-sans=192.168.26.60,vms60 --image-repository registry.aliyuncs.com/google\_containers

# 初始化的过程中会拉取一些必要的镜像，所以等待时间会相对来说较长一些

# 初始化成功的特征，如下图所示

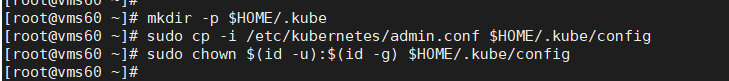


# 配置集群管理员

mkdir -p $HOME/.kube

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

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



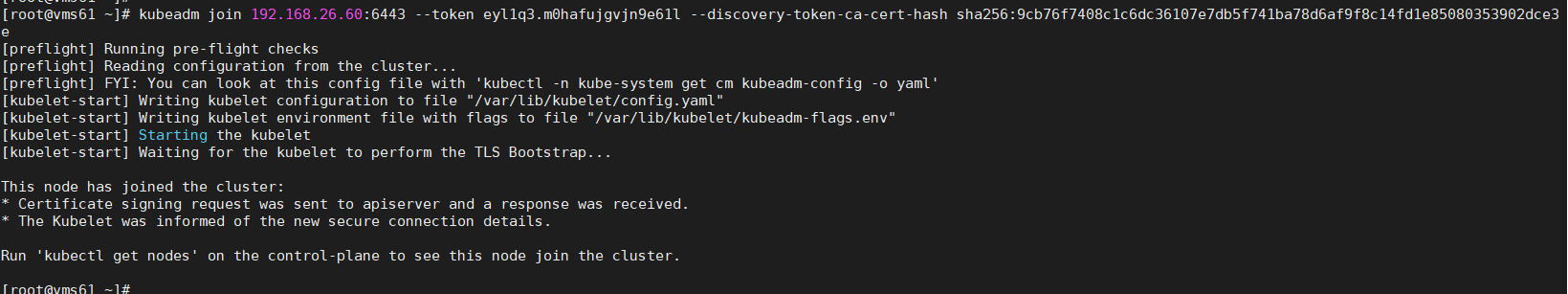
2.4加入node节点

以下操作只在node节点主机上进行

# 集群初始化成功后，会生成一段token，将这段token复制到node节点上执行（这里直接使用上述生成的token）

# 这个tokne可以保存下来，如果node节点掉了还可以重新加入。当然也可以重新在master上生成新的token给node重新加入，后续会给出方法

kubeadm join 192.168.26.60:6443 --token eyl1q3.m0hafujgvjn9e61l --discovery-token-ca-cert-hash sha256:9cb76f7408c1c6dc36107e7db5f741ba78d6af9f8c14fd1e85080353902dce3e

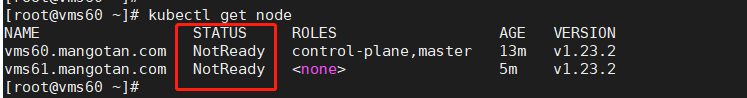


2.5网络插件安装

以下操作看清楚说明，分别在master和node节点主机上进行

# 加入了node节点后，在master节点上查看节点的状态

Kubectl get node



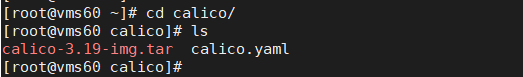
# 安装网络插件（这里推荐使用calico）

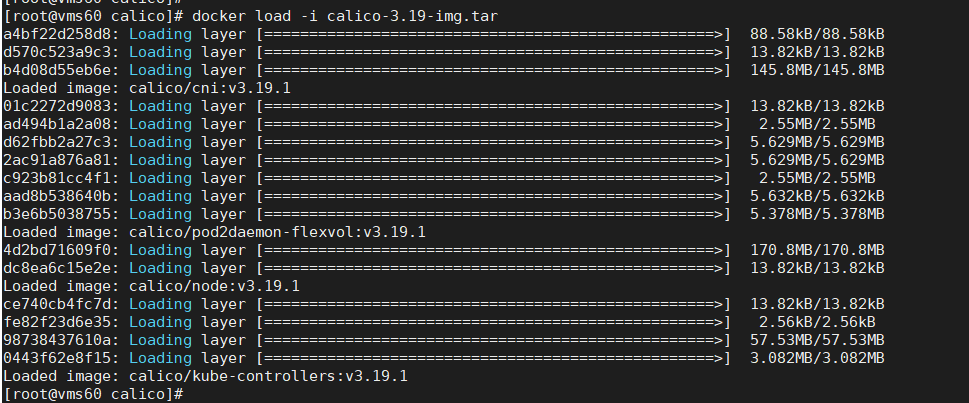
# 在系统里已经预先准备好了calico插件的安装包，所以直接使用就可以了



# 在master和node节点主机上分别cd到calico的目录，执行以下命令拉取需要的镜像

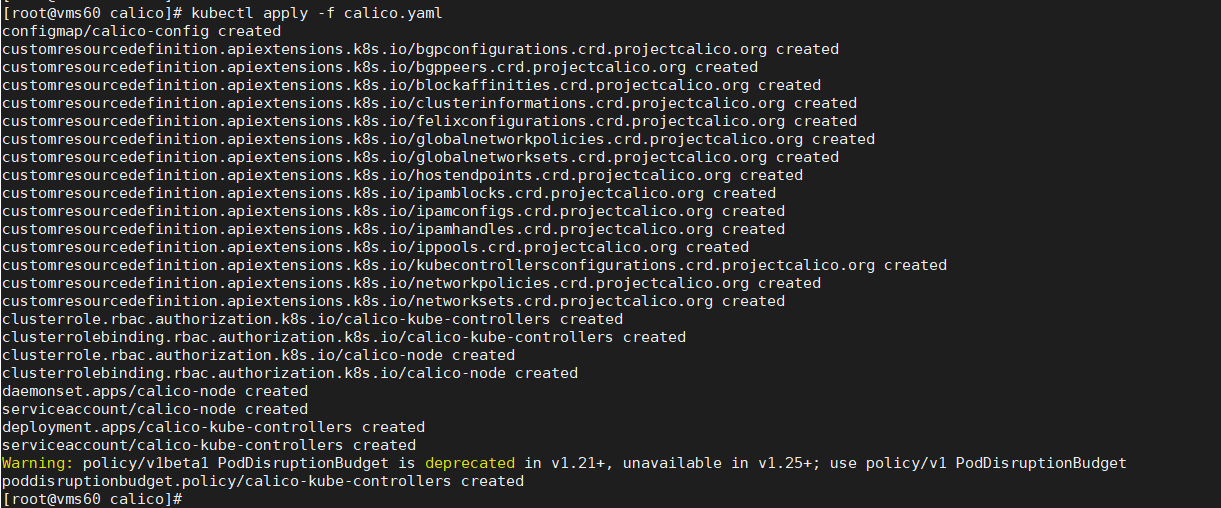
docker load -i calico-3.19-img.tar



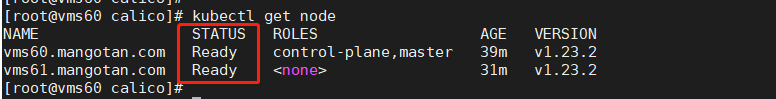


# 等待两个节点的镜像都拉取完成后，只需要在master节点主机上执行以下命令安装calico

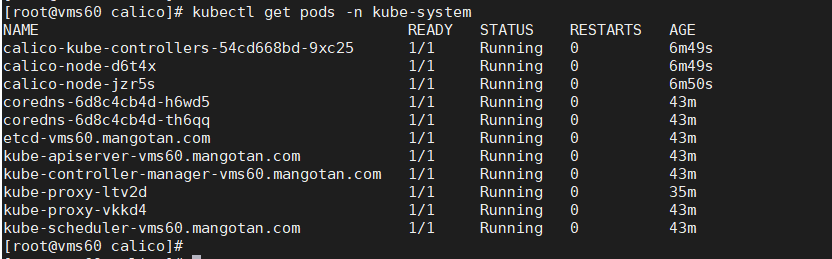
kubectl apply -f calico.yaml



# 安装完成后，再查看node节点的状态，如果变为Ready则表示网络插件运行正常，也可以通过查看kube-system命名空间下的pod运行状态，如果全部为Running，则kubenetes到此就完成了部署



kubectl get pods -n kube-system

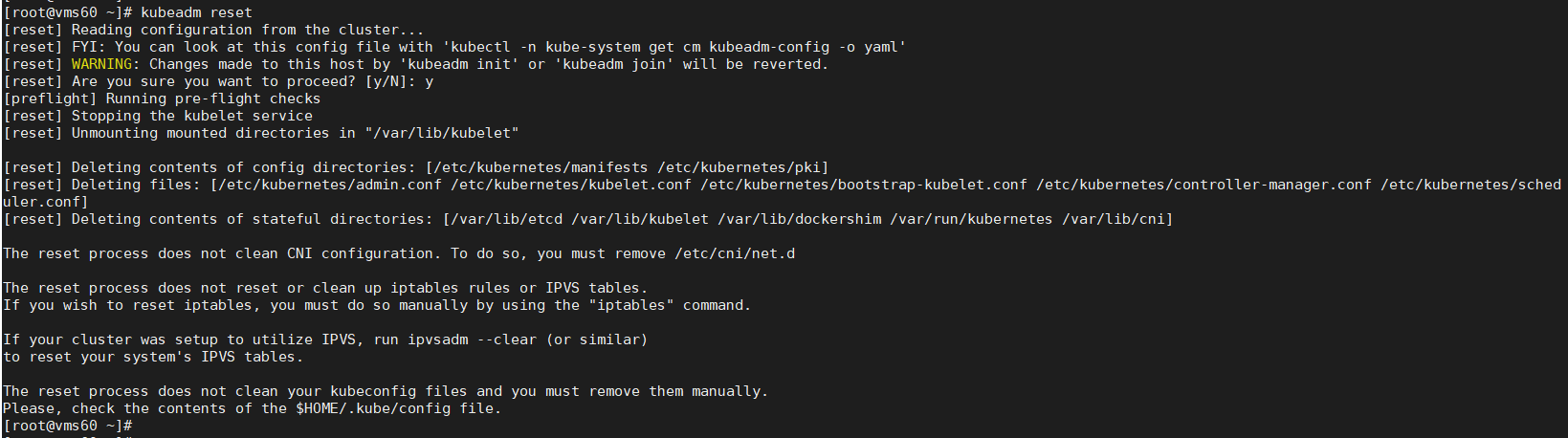


###### Kubernetes重置

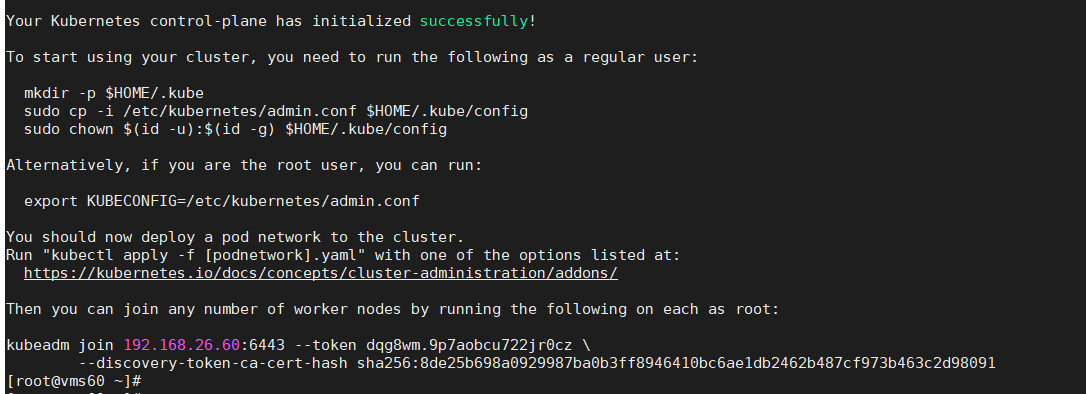
如果集群出现一些临时无法解决的报错异常问题，并且影响了测试，可以采用极端的手段处理，重置集群

# 分别在master和node节点主机上执行重置命令

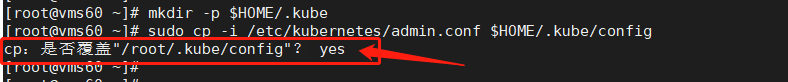
kubeadm reset



# 在master节点主机上重复 2.3 的操作，再次初始化集群，初始化成功后会生成新的token

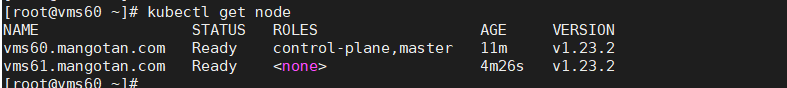


# 再次配置集群管理员时会出现如下请求，此时必须手动的输入yes（如果直接按回车键，默认是执行no操作）



# 完成上述操作后，在node节点主机上重复2.4的操作，将新生成的token复制执行

# 执行重置操作不需要再重新安装网络插件，在节点加入成功后，集群会自动的重新运行原有的网络插件



#### 常见报错异常解决

###### X509报错



解决办法：

export KUBECONFIG=/etc/kubernetes/kubeconfig.conf

###### **refused异常**



解决办法：

export KUBECONFIG=/etc/kubernetes/admin.conf