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：

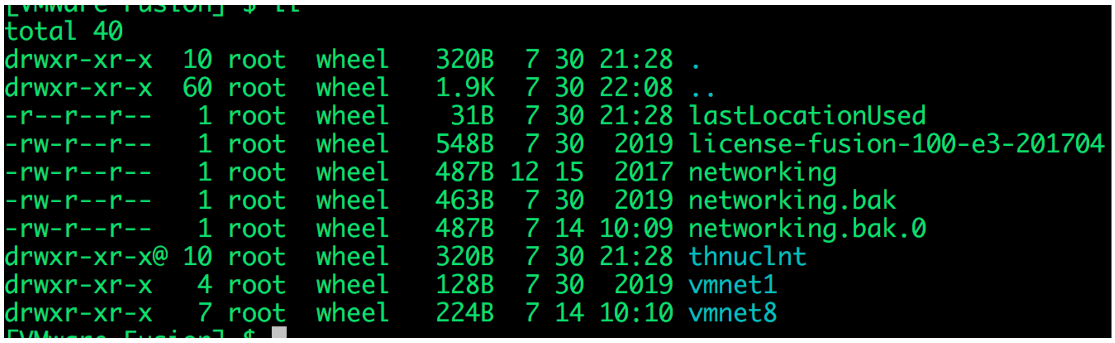
centos7 虚拟机的密码：root/123qwe

1.1 网络模式设置【vmware fushion mac 版】：

vmware fusion 的网络 NAT 模式[MAC OS]

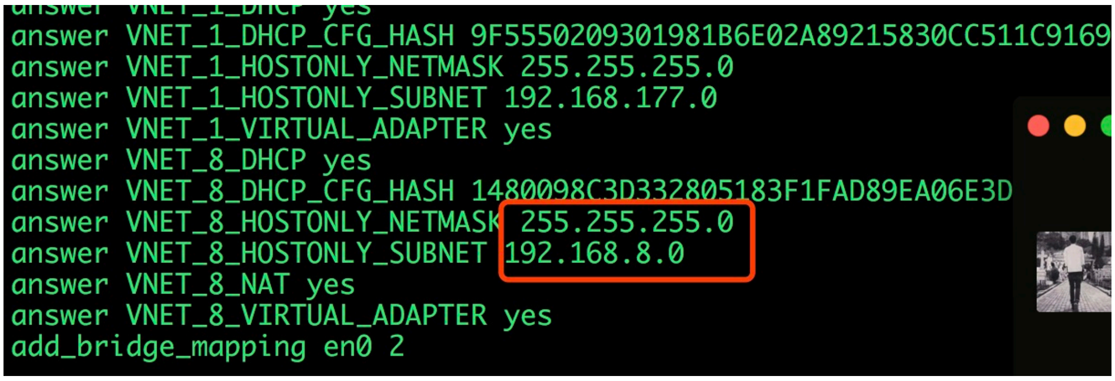
进去以下目录：

```
cd /Library/Preferences/VMware Fusion
```



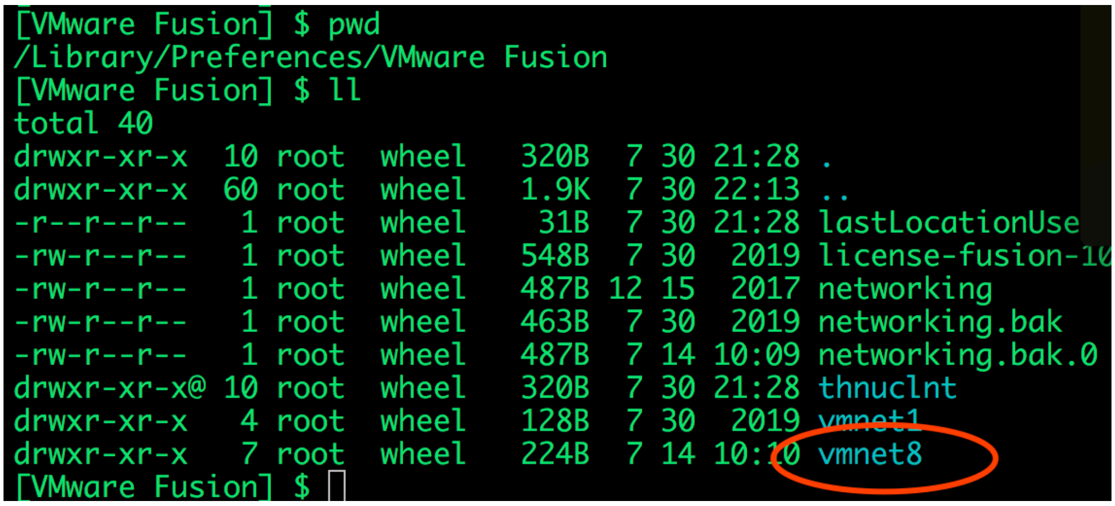
```
[VMware Fusion] $ ll
total 40
drwxr-xr-x  10 root  wheel   320B  7 30 21:28 .
drwxr-xr-x  60 root  wheel   1.9K  7 30 22:08 ..
-r--r--r--   1 root  wheel    31B  7 30 21:28 lastLocationUsed
-rw-r--r--   1 root  wheel   548B  7 30  2019 license-fusion-100-e3-201704
-rw-r--r--   1 root  wheel   487B 12 15  2017 networking
-rw-r--r--   1 root  wheel   463B  7 30  2019 networking.bak
-rw-r--r--   1 root  wheel   487B  7 14 10:09 networking.bak.0
drwxr-xr-x@ 10 root  wheel   320B  7 30 21:28 thnuclnt
drwxr-xr-x   4 root  wheel   128B  7 30  2019 vmnet1
drwxr-xr-x   7 root  wheel   224B  7 14 10:10 vmnet8
[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：



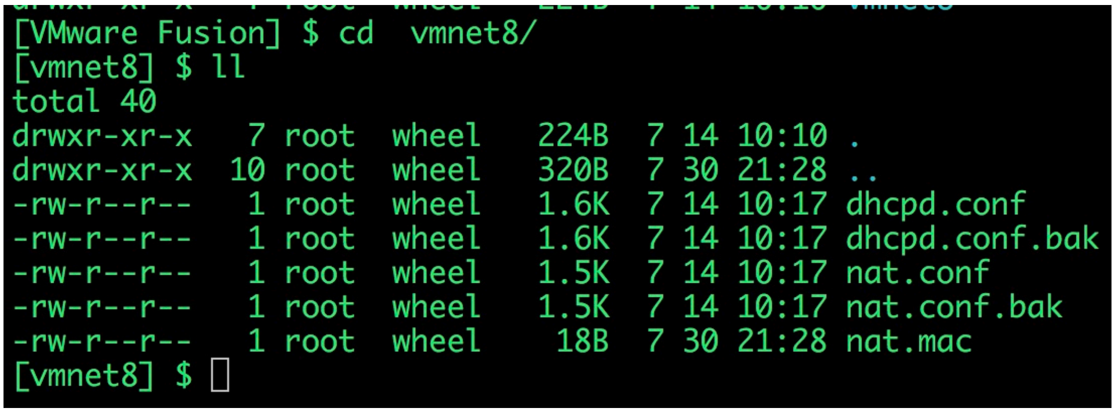
```
answer VNET_1_DHCP yes
answer VNET_1_DHCP_CFG_HASH 9F5550209301981B6E02A89215830CC511C9169
answer VNET_1_HOSTONLY_NETMASK 255.255.255.0
answer VNET_1_HOSTONLY_SUBNET 192.168.177.0
answer VNET_1_VIRTUAL_ADAPTER yes
answer VNET_8_DHCP yes
answer VNET_8_DHCP_CFG_HASH 1480098C3D332805183F1FAD89EA06E3D
answer VNET_8_HOSTONLY_NETMASK 255.255.255.0
answer VNET_8_HOSTONLY_SUBNET 192.168.8.0
answer VNET_8_NAT yes
answer VNET_8_VIRTUAL_ADAPTER yes
add_bridge_mapping en0 2
```

回到当前目录，进入 vmnet8 子目录：



```
[VMware Fusion] $ pwd
/Library/Preferences/VMware Fusion
[VMware Fusion] $ ll
total 40
drwxr-xr-x  10 root  wheel   320B  7 30 21:28 .
drwxr-xr-x  60 root  wheel   1.9K  7 30 22:13 ..
-r--r--r--   1 root  wheel    31B  7 30 21:28 lastLocationUse
-rw-r--r--   1 root  wheel   548B  7 30  2019 license-fusion-10
-rw-r--r--   1 root  wheel   487B 12 15  2017 networking
-rw-r--r--   1 root  wheel   463B  7 30  2019 networking.bak
-rw-r--r--   1 root  wheel   487B  7 14 10:09 networking.bak.0
drwxr-xr-x@ 10 root  wheel   320B  7 30 21:28 thnuclnt
drwxr-xr-x   4 root  wheel   128B  7 30  2019 vmnet1
drwxr-xr-x   7 root  wheel   224B  7 14 10:10 vmnet8
[VMware Fusion] $ 
```

该目录下有以下文件：



```
drwxr-xr-x   7 root  wheel   224B  7 14 10:10 vmnet8
[VMware Fusion] $ cd  vmnet8/
[vmnet8] $ ll
total 40
drwxr-xr-x   7 root  wheel   224B  7 14 10:10 .
drwxr-xr-x  10 root  wheel   320B  7 30 21:28 ..
-rw-r--r--   1 root  wheel   1.6K  7 14 10:17 dhcpd.conf
-rw-r--r--   1 root  wheel   1.6K  7 14 10:17 dhcpd.conf.bak
-rw-r--r--   1 root  wheel   1.5K  7 14 10:17 nat.conf
-rw-r--r--   1 root  wheel   1.5K  7 14 10:17 nat.conf.bak
-rw-r--r--   1 root  wheel    18B  7 30 21:28 nat.mac
[vmnet8] $ 
```

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;
}
```

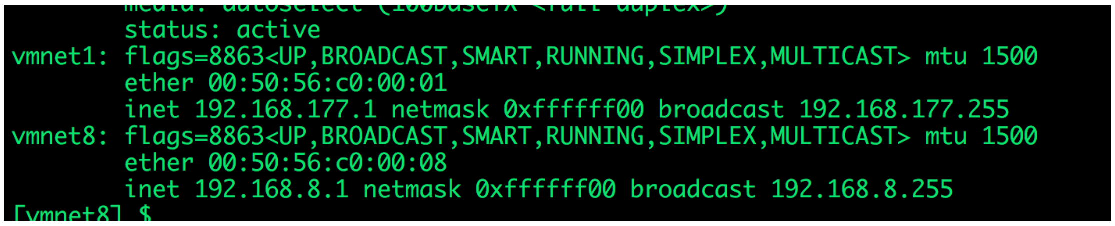
修改 nat.conf 如下：

```
# 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：

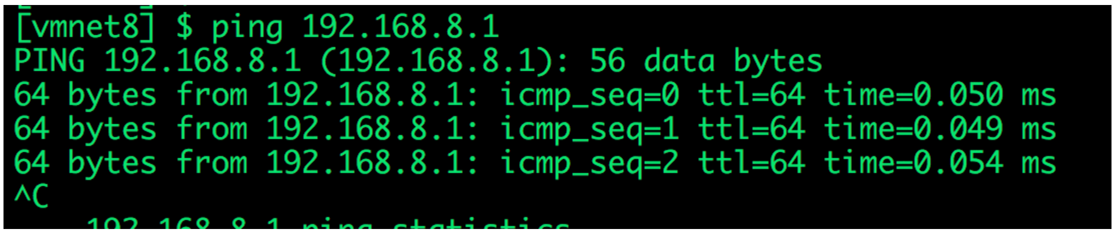
```
sudo /Applications/VMware\ Fusion.app/Contents/Library/vmnet-cli --stop
sudo /Applications/VMware\ Fusion.app/Contents/Library/vmnet-cli --start
```

查看一下 ifconfig：



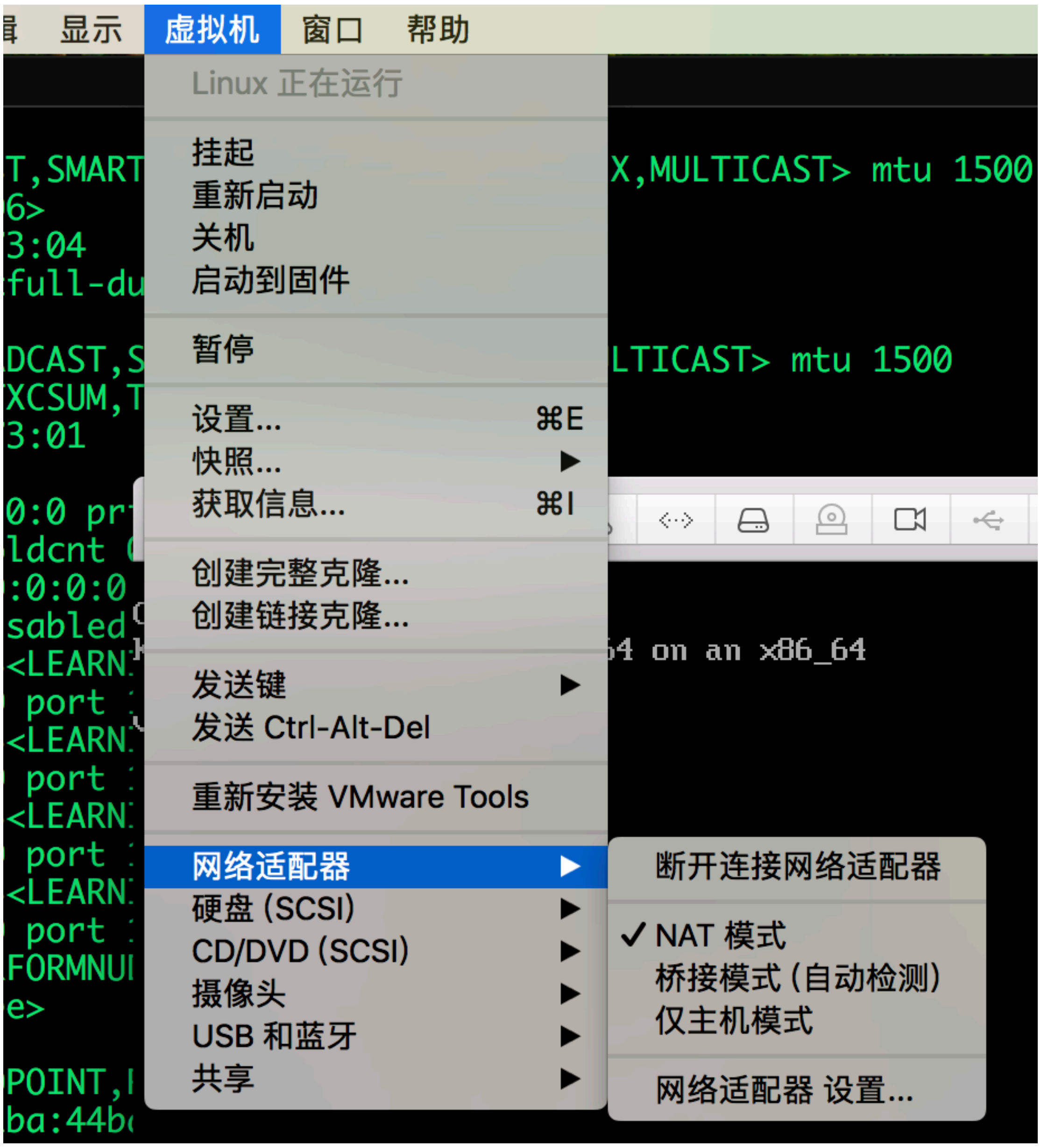
```
        status: active
vmnet1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:50:56:c0:00:01
        inet 192.168.177.1 netmask 0xffffff00 broadcast 192.168.177.255
vmnet8: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:50:56:c0:00:08
        inet 192.168.8.1 netmask 0xffffff00 broadcast 192.168.8.255
[vmnet8] $
```

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



```
[vmnet8] $ ping 192.168.8.1
PING 192.168.8.1 (192.168.8.1): 56 data bytes
64 bytes from 192.168.8.1: icmp_seq=0 ttl=64 time=0.050 ms
64 bytes from 192.168.8.1: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 192.168.8.1: icmp_seq=2 ttl=64 time=0.054 ms
^C
```

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，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
```

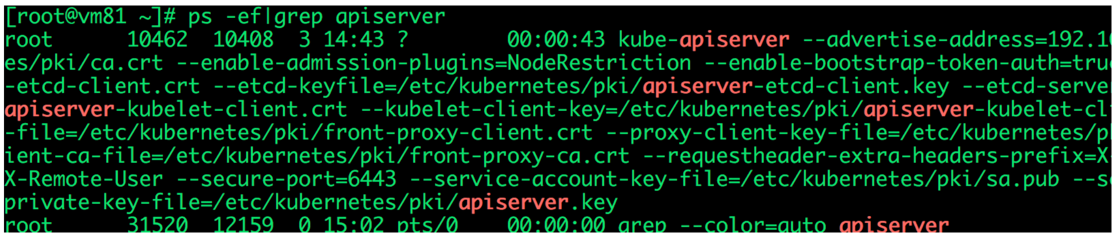
执行 kubectl get node，确保 node 都处于 ready 状态，如果没有处于 ready 状态，则：
尝试：

1、 重启 kubelet 和 docker
    a) systemctl restart kubelet
    b) 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：

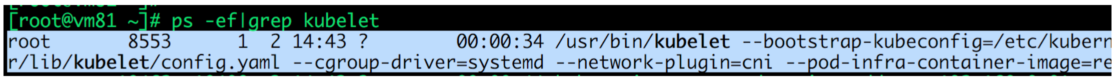


```
[root@vm81 ~]# ps -ef|grep apiserver
root      10462  10408  3 14:43 ?        00:00:43 kube-apiserver --advertise-address=192.1
es/pki/ca.crt --enable-admission-plugins=NodeRestriction --enable-bootstrap-token-auth=tru
-etcd-client.crt --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key --etcd-serve
apiserver-kubelet-client.crt --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-cl
-file=/etc/kubernetes/pki/front-proxy-client.crt --proxy-client-key-file=/etc/kubernetes/p
ient-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt --requestheader-extra-headers-prefix=X
X-Remote-User --secure-port=6443 --service-account-key-file=/etc/kubernetes/pki/sa.pub --s
private-key-file=/etc/kubernetes/pki/apiserver.key
root      31520  12159  0 15:02 pts/0    00:00:00 grep --color=auto apiserver
```

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：

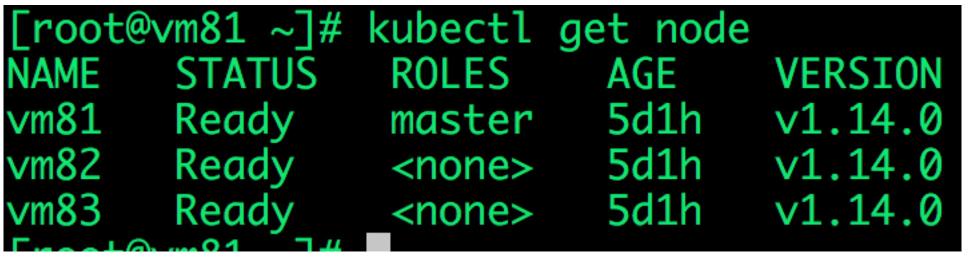


```
[root@vm81 ~]# ps -ef|grep kubelet
root      8553     1  2 14:43 ?        00:00:34 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubern
r/lib/kubelet/config.yaml --cgroup-driver=systemd --network-plugin=cni --pod-infra-container-image=re
```

如果没有启动：执行

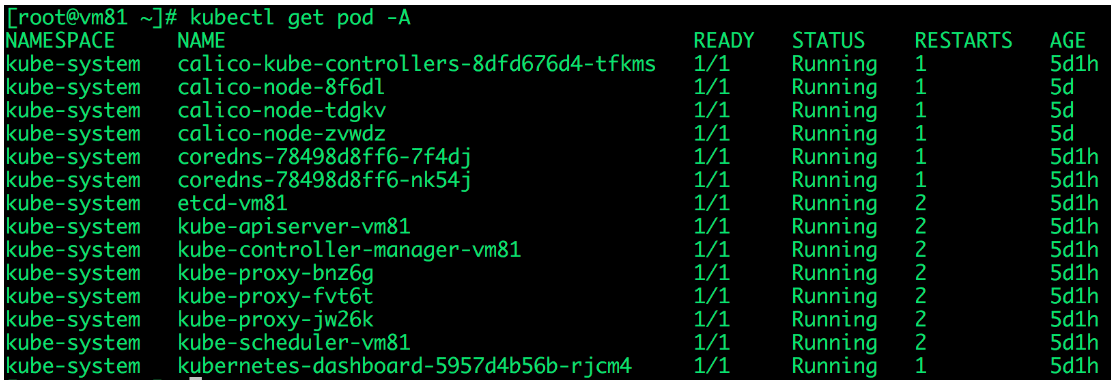
systemctl restart kubelet

确保 k8s 的 node 处于 ready 状态：



```
[root@vm81 ~]# kubectl get node
NAME    STATUS   ROLES    AGE    VERSION
vm81    Ready    master   5d1h   v1.14.0
vm82    Ready    <none>   5d1h   v1.14.0
vm83    Ready    <none>   5d1h   v1.14.0
```

如果 node 没有处于 ready 状态，检查 pod：



```
[root@vm81 ~]# kubectl get pod -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-8dfd676d4-tfkms    1/1     Running   1          5d1h
kube-system   calico-node-8f6dl                          1/1     Running   1          5d
kube-system   calico-node-tdgkv                          1/1     Running   1          5d
kube-system   calico-node-zvwdz                          1/1     Running   1          5d
kube-system   coredns-78498d8ff6-7f4dj                   1/1     Running   1          5d1h
kube-system   coredns-78498d8ff6-nk54j                   1/1     Running   1          5d1h
kube-system   etcd-vm81                                  1/1     Running   2          5d1h
kube-system   kube-apiserver-vm81                        1/1     Running   2          5d1h
kube-system   kube-controller-manager-vm81               1/1     Running   2          5d1h
kube-system   kube-proxy-bnz6g                           1/1     Running   2          5d1h
kube-system   kube-proxy-fvt6t                           1/1     Running   2          5d1h
kube-system   kube-proxy-jw26k                           1/1     Running   2          5d1h
kube-system   kube-scheduler-vm81                        1/1     Running   2          5d1h
kube-system   kubernetes-dashboard-5957d4b56b-rjcm4      1/1     Running   1          5d1h
```

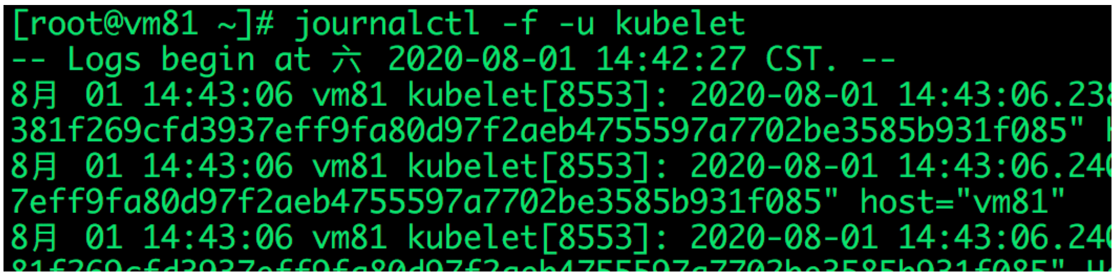
检查 pod

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：



```
[root@vm81 ~]# journalctl -f -u kubelet
-- Logs begin at 六 2020-08-01 14:42:27 CST. --
8月 01 14:43:06 vm81 kubelet[8553]: 2020-08-01 14:43:06.23
381f269cfd3937eff9fa80d97f2aeb4755597a7702be3585b931f085"
8月 01 14:43:06 vm81 kubelet[8553]: 2020-08-01 14:43:06.24
7eff9fa80d97f2aeb4755597a7702be3585b931f085" host="vm81"
8月 01 14:43:06 vm81 kubelet[8553]: 2020-08-01 14:43:06.24
```

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：

```
rm -rf ~/.kube
systemctl stop kubelet
systemctl stop docker
iptables --flush
iptables -tnat --flush
systemctl start kubelet
systemctl start docker
```