K8s高可用集群环境搭建

目录

作者：杨遥

[1. 简介](#_Toc12774_WPSOffice_Level1) [1](#_Toc12774_WPSOffice_Level1)

[2. 整体架构信息](#_Toc1604_WPSOffice_Level1) [1](#_Toc1604_WPSOffice_Level1)

[3. 环境初始化](#_Toc23019_WPSOffice_Level1) [1](#_Toc23019_WPSOffice_Level1)

[3.1配置hosts文件](#_Toc1604_WPSOffice_Level2) [1](#_Toc1604_WPSOffice_Level2)

[3.2关闭selinux和防火墙](#_Toc23019_WPSOffice_Level2) [2](#_Toc23019_WPSOffice_Level2)

[3.3关闭swap](#_Toc20946_WPSOffice_Level2) [2](#_Toc20946_WPSOffice_Level2)

[3.4创建秘钥](#_Toc8439_WPSOffice_Level2) [2](#_Toc8439_WPSOffice_Level2)

[3.5配置内核转发参数](#_Toc8366_WPSOffice_Level2) [2](#_Toc8366_WPSOffice_Level2)

[3.6加载ipvs模块](#_Toc25861_WPSOffice_Level2) [3](#_Toc25861_WPSOffice_Level2)

[4. 安装keepalived](#_Toc20946_WPSOffice_Level1) [3](#_Toc20946_WPSOffice_Level1)

[5. 安装haproxy](#_Toc8439_WPSOffice_Level1) [5](#_Toc8439_WPSOffice_Level1)

[6.安装docker-ce](#_Toc8366_WPSOffice_Level1) [6](#_Toc8366_WPSOffice_Level1)

[6.1安装一些必要的组件信息](#_Toc27831_WPSOffice_Level2) [6](#_Toc27831_WPSOffice_Level2)

[6.2添加docker的yum源](#_Toc14831_WPSOffice_Level2) [6](#_Toc14831_WPSOffice_Level2)

[6.3更新缓存](#_Toc15883_WPSOffice_Level2) [6](#_Toc15883_WPSOffice_Level2)

[6.4安装](#_Toc25690_WPSOffice_Level2) [6](#_Toc25690_WPSOffice_Level2)

[6.5启动并设置成开机启动](#_Toc14265_WPSOffice_Level2) [7](#_Toc14265_WPSOffice_Level2)

[7.安装kubernetes组件](#_Toc25861_WPSOffice_Level1) [7](#_Toc25861_WPSOffice_Level1)

[7.1添加kubernetes的yum源](#_Toc4139_WPSOffice_Level2) [7](#_Toc4139_WPSOffice_Level2)

[7.2更新缓存](#_Toc24275_WPSOffice_Level2) [7](#_Toc24275_WPSOffice_Level2)

[7.3安装组件](#_Toc15310_WPSOffice_Level2) [7](#_Toc15310_WPSOffice_Level2)

[7.4设置kubelet开机自启动](#_Toc5945_WPSOffice_Level2) [7](#_Toc5945_WPSOffice_Level2)

[8.初始化集群](#_Toc27831_WPSOffice_Level1) [7](#_Toc27831_WPSOffice_Level1)

[8.1获取默认配置文件信息](#_Toc1622_WPSOffice_Level2) [7](#_Toc1622_WPSOffice_Level2)

[8.2修改初始化文件](#_Toc27541_WPSOffice_Level2) [7](#_Toc27541_WPSOffice_Level2)

[8.3初始化集群](#_Toc16388_WPSOffice_Level2) [9](#_Toc16388_WPSOffice_Level2)

[8.4初始化其他master节点](#_Toc8801_WPSOffice_Level2) [10](#_Toc8801_WPSOffice_Level2)

[8.5验证集群环境](#_Toc22464_WPSOffice_Level2) [10](#_Toc22464_WPSOffice_Level2)

[1.下载yaml](#_Toc12352_WPSOffice_Level3) [11](#_Toc12352_WPSOffice_Level3)

[2.修改网络地址跟pod网段保持一致](#_Toc8115_WPSOffice_Level3) [11](#_Toc8115_WPSOffice_Level3)

[3.创建calico网络pod](#_Toc12241_WPSOffice_Level3) [11](#_Toc12241_WPSOffice_Level3)

[8.6加入node节点](#_Toc12352_WPSOffice_Level2) [12](#_Toc12352_WPSOffice_Level2)

[9.注意事项](#_Toc12352_WPSOffice_Level1) [12](#_Toc12352_WPSOffice_Level1)

[10.外部如何访问](#_Toc8115_WPSOffice_Level1) [13](#_Toc8115_WPSOffice_Level1)

# 

# 简介

本章节使用kubeadm搭建一套三主三从的高可用kubernetes集群,相比单master而言，多master更加有利于我们生产环境的稳定性。就算master节点宕机一台，我们依然有其他master节点保证我们的集群环境。

多master节点需要依赖于keepalive还生成一个vip，也就是虚拟ip，做过nginx+keepalive的小伙伴们都知道，还需要一个haproxy来实现负载策略。

# 整体架构信息

Linux版本CentOS7,内核版本3.10.0-693.el7.x86\_64

Kubernetes 默认安装最新，目前1.15.0

网段配置：pod网段10.244.0.0/16 service网段10.96.0.0/12

|  |  |  |  |
| --- | --- | --- | --- |
| 节点名称 | 角色 | IP地址 | 安装组件 |
| 负载Vip | Vip | 192.168.3.100 |  |
| master1 | master | 192.168.3.18 | kubeadm、kubelet、kubectl、docker、haproxy、keepalived |
| master2 | master | 192.168.3.19 | kubeadm、kubelet、kubectl、docker、haproxy、keepalived |
| master3 | master | 192.168.3.20 | kubeadm、kubelet、kubectl、docker、haproxy、keepalived |
| node1 | node | 192.168.3.21 | kubeadm、kubelet、kubectl、docker |
| node2 | node | 192.168.3.22 | kubeadm、kubelet、kubectl、docker |
| node3 | node | 192.168.3.23 | kubeadm、kubelet、kubectl、docker |

# 环境初始化

## 3.1配置hosts文件

在每个主机上配置

cat >>/etc/hosts<<EOF

192.168.3.18 master1

192.168.3.19 master2

192.168.3.20 master3

192.168.3.21 node1

192.168.3.22 node2

192.168.3.23 node3

EOF

## 3.2关闭selinux和防火墙

sed -ri 's#(SELINUX=).\*#\1disabled#' /etc/selinux/config

setenforce 0

systemctl disable firewalld

systemctl stop firewalld

## 3.3关闭swap

swapoff -a

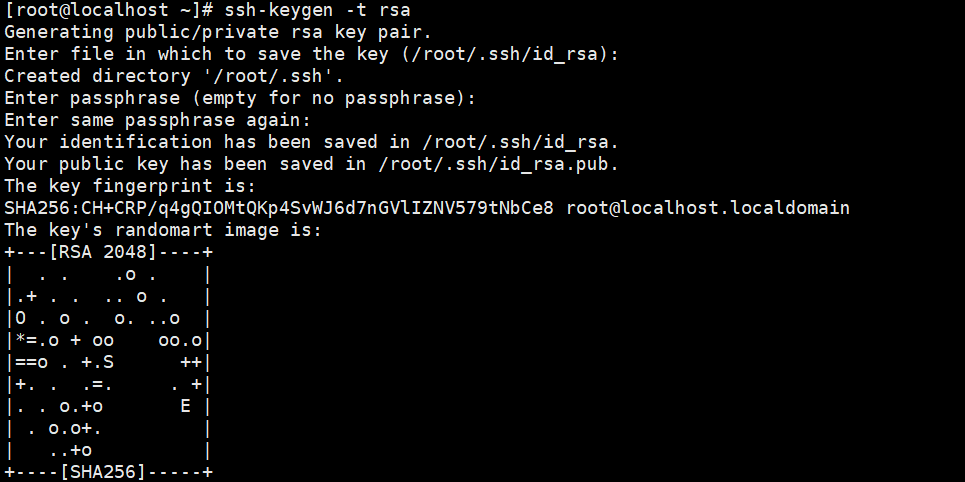
sed -i 's/.\*swap.\*/#&/' /etc/fstab

## 3.4创建秘钥

在master1节点上创建秘钥，设置免密登录其他服务器，主要是为了后面的远程拷贝证书

ssh-keygen -t rsa

一路回车即可。



for n in `seq -w 2 3`;do ssh-copy-id master$n;done

## 3.5配置内核转发参数

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

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

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

net.ipv4.ip\_nonlocal\_bind = 1

net.ipv4.ip\_forward = 1

vm.swappiness=0

EOF

sysctl --system

## 3.6加载ipvs模块

cat > /etc/sysconfig/modules/ipvs.modules <<EOF

#!/bin/bash

modprobe -- ip\_vs

modprobe -- ip\_vs\_rr

modprobe -- ip\_vs\_wrr

modprobe -- ip\_vs\_sh

modprobe -- nf\_conntrack\_ipv4

EOF

chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep -e ip\_vs -e nf\_conntrack\_ipv4

# 安装keepalived

3个master节点上安装keepalived

yum install -y keepalived

修改keepalived的配置文件，保证priority不一样即可，这里我们分别采用100,90,80

配置文件路径/etc/keepalived/keepalived.conf

配置文件内容：

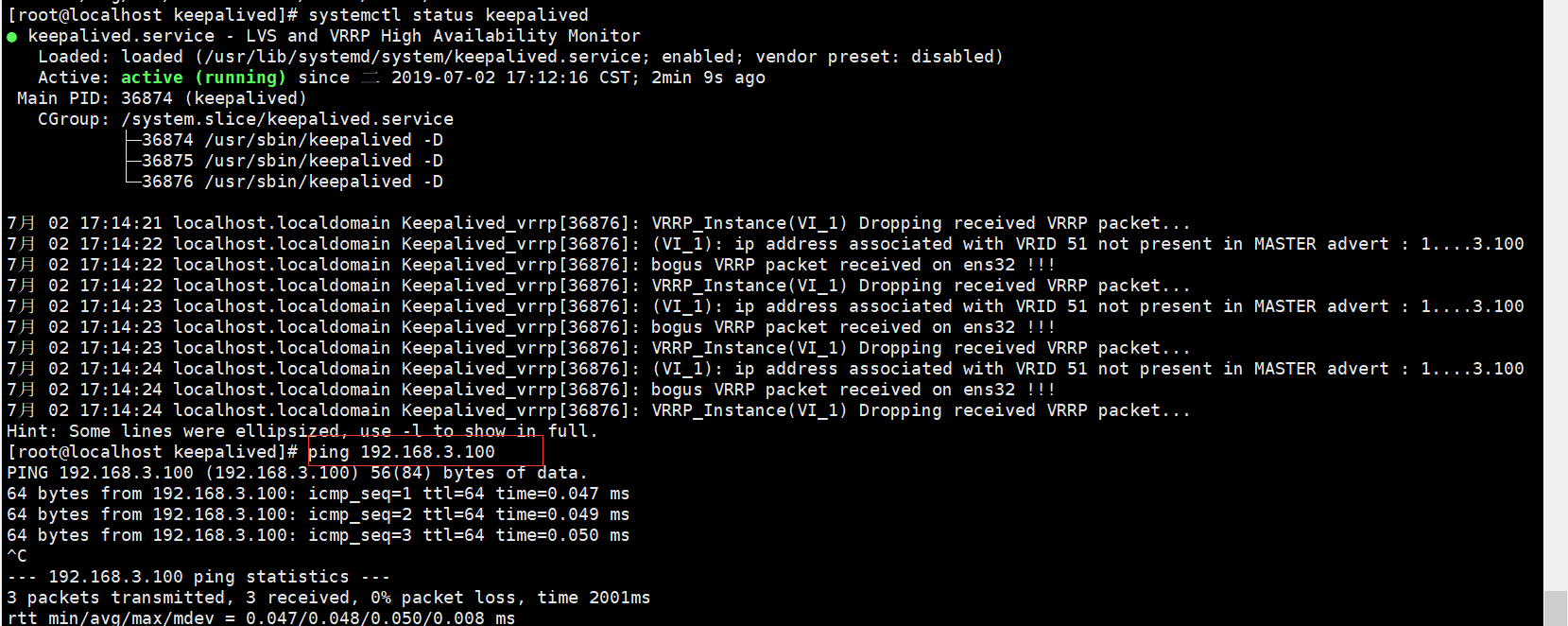
|  |
| --- |
| ! Configuration File for keepalived  global\_defs {  notification\_email {  acassen@firewall.loc  failover@firewall.loc  sysadmin@firewall.loc  }  notification\_email\_from Alexandre.Cassen@firewall.loc  smtp\_server 192.168.200.1  smtp\_connect\_timeout 30  router\_id LVS\_DEVEL  vrrp\_skip\_check\_adv\_addr  #vrrp\_strict  vrrp\_garp\_interval 0  vrrp\_gna\_interval 0  }  vrrp\_instance VI\_1 {  state MASTER  interface ens32  virtual\_router\_id 51  priority 100  advert\_int 1  authentication {  auth\_type PASS  auth\_pass 1111  }  virtual\_ipaddress {  192.168.3.100  }  } |

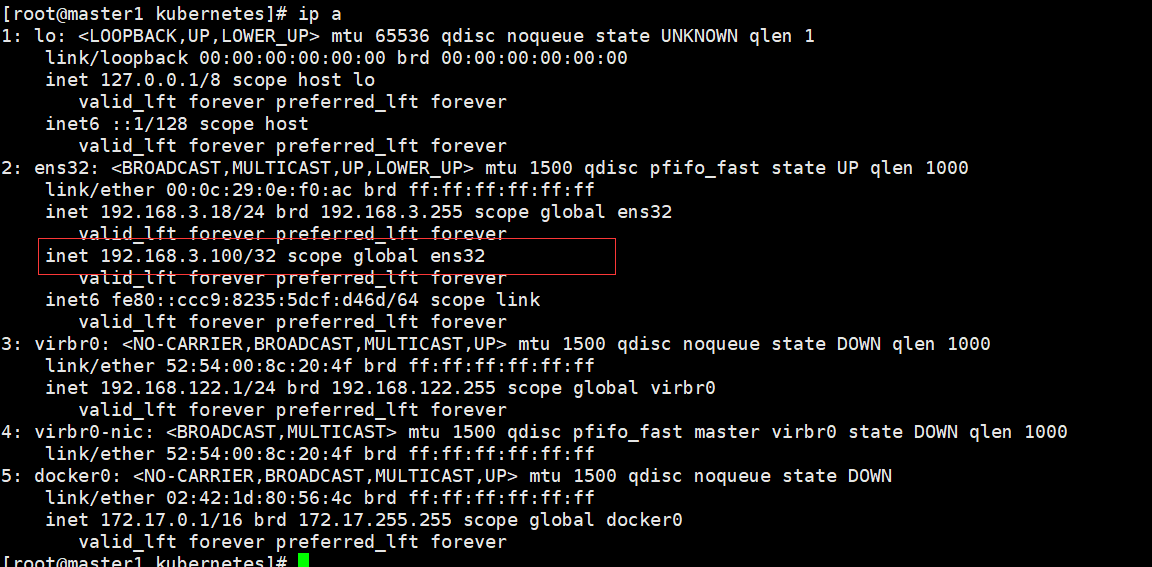
启动keepalive并设置成开启启动

systemctl start keepalived && systemctl enable keepalived

检查虚拟vip是否正常工作

ifconfig查看vip是否绑定成功，如果能ping 192.168.3.100则说明vip绑定成功，也可以使用ip a看虚拟vip是否绑定到当前指定网卡上





# 安装haproxy

yum install -y haproxy

修改haproxy的配置文件，因为我们的haproxy安装在3个master节点上面，haproxy实现api-server的负载均衡，所有不能用api-server的端口，我们设置端口信息为8443，默认是api-server是8443

配置文件路径/etc/haproxy/haproxy.conf

配置文件内容：

|  |
| --- |
| global  chroot /var/lib/haproxy  daemon  group haproxy  user haproxy  log 127.0.0.1:514 local0 warning  pidfile /var/lib/haproxy.pid  maxconn 20000  spread-checks 3  nbproc 8  defaults  log global  mode tcp  retries 3  option redispatch  listen https-apiserver  bind 192.168.3.100:8443  mode tcp  balance roundrobin  timeout server 900s  timeout connect 15s  server apiserver01 192.168.3.18:6443 check port 6443 inter 5000 fall 5  server apiserver02 192.168.3.19:6443 check port 6443 inter 5000 fall 5  server apiserver03 192.168.3.20:6443 check port 6443 inter 5000 fall 5 |

启动并设置为开机启动

systemctl start haproxy && systemctl enable haproxy

# 6.安装docker-ce

## 6.1安装一些必要的组件信息

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

## 6.2添加docker的yum源

sudo yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

6.3更新缓存

sudo yum makecache fast

6.4安装

yum install -y docker-ce

6.5启动并设置成开机启动

systemctl start docker && systemctl enable docker

# 7.安装kubernetes组件

## 7.1添加kubernetes的yum源

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

[kubernetes]

name=Kubernetes

baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/

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/rpm-package-key.gpg

EOF

## 7.2更新缓存

yum makecache fast

## 7.3安装组件

yum install -y kubeadm kubelet kubectl

## 7.4设置kubelet开机自启动

systemctl enable kubelet

# 8.初始化集群

## 8.1获取默认配置文件信息

kubeadm config print init-defaults > kubeadm-init.yaml

## 8.2修改初始化文件

修改之后的配置文件如下，注意以下几点

advertiseAddress，imageRepository，kubernetesVersion，podSubnet，controlPlaneEndpoint

|  |
| --- |
| apiVersion: kubeadm.k8s.io/v1beta2  bootstrapTokens:  - groups:  - system:bootstrappers:kubeadm:default-node-token  token: abcdef.0123456789abcdef  ttl: 24h0m0s  usages:  - signing  - authentication  kind: InitConfiguration  localAPIEndpoint:  advertiseAddress: 192.168.3.18  bindPort: 6443  nodeRegistration:  criSocket: /var/run/dockershim.sock  name: master1  taints:  - effect: NoSchedule  key: node-role.kubernetes.io/master  ---  apiServer:  timeoutForControlPlane: 4m0s  apiVersion: kubeadm.k8s.io/v1beta2  certificatesDir: /etc/kubernetes/pki  clusterName: kubernetes  controlPlaneEndpoint: 192.168.3.100:8443  controllerManager: {}  dns:  type: CoreDNS  etcd:  local:  dataDir: /var/lib/etcd  imageRepository: registry.cn-hangzhou.aliyuncs.com/google\_containers  kind: ClusterConfiguration  kubernetesVersion: v1.15.0  networking:  dnsDomain: cluster.local  serviceSubnet: 10.96.0.0/12  podSubnet: 10.244.0.0/16  scheduler: {} |

## 8.3初始化集群

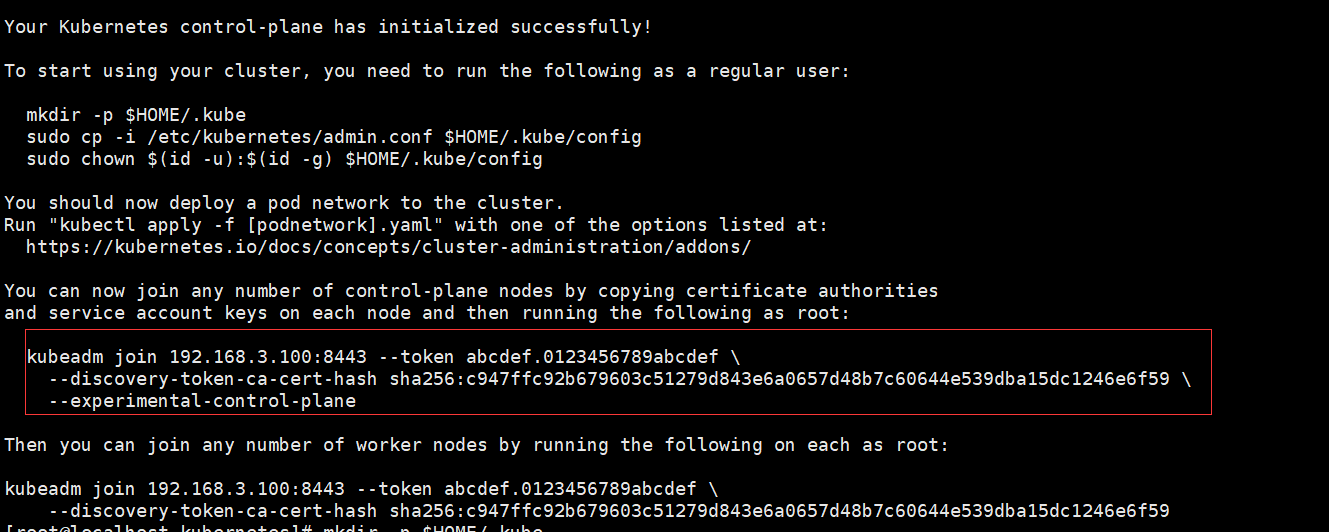
kubeadm init --config kubeadm-init.yaml

集群初始化完成后执行一下三步：

mkdir -p $HOME/.kube

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

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



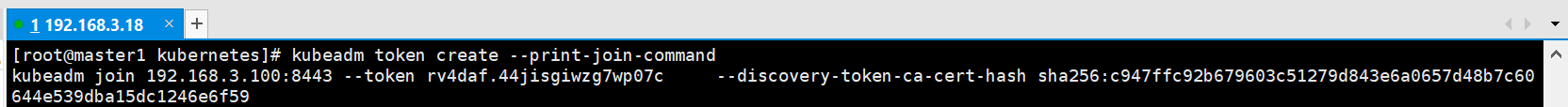
画红线部分主要是针对其他，master节点加入集群，node节点加入集群的话则不需要--experimental-control-plane。

注意这里token默认24小时过期，查看token的命令

kubeadm token list

token过期后，如果我们需要重新生成token，可以使用以下命令，生成命令默认是node节点加入的命令，如需要角色是master则需要在最后面加上--experimental-control-plane。

kubeadm token create --print-join-command



接着我们将证书拷贝到其他master节点

|  |
| --- |
| USER=root  CONTROL\_PLANE\_IPS="master2 master3"  for host in ${CONTROL\_PLANE\_IPS}; do  ssh "${USER}"@$host "mkdir -p /etc/kubernetes/pki/etcd"  scp /etc/kubernetes/pki/ca.\* "${USER}"@$host:/etc/kubernetes/pki/  scp /etc/kubernetes/pki/sa.\* "${USER}"@$host:/etc/kubernetes/pki/  scp /etc/kubernetes/pki/front-proxy-ca.\* "${USER}"@$host:/etc/kubernetes/pki/  scp /etc/kubernetes/pki/etcd/ca.\* "${USER}"@$host:/etc/kubernetes/pki/etcd/  scp /etc/kubernetes/admin.conf "${USER}"@$host:/etc/kubernetes/  done |

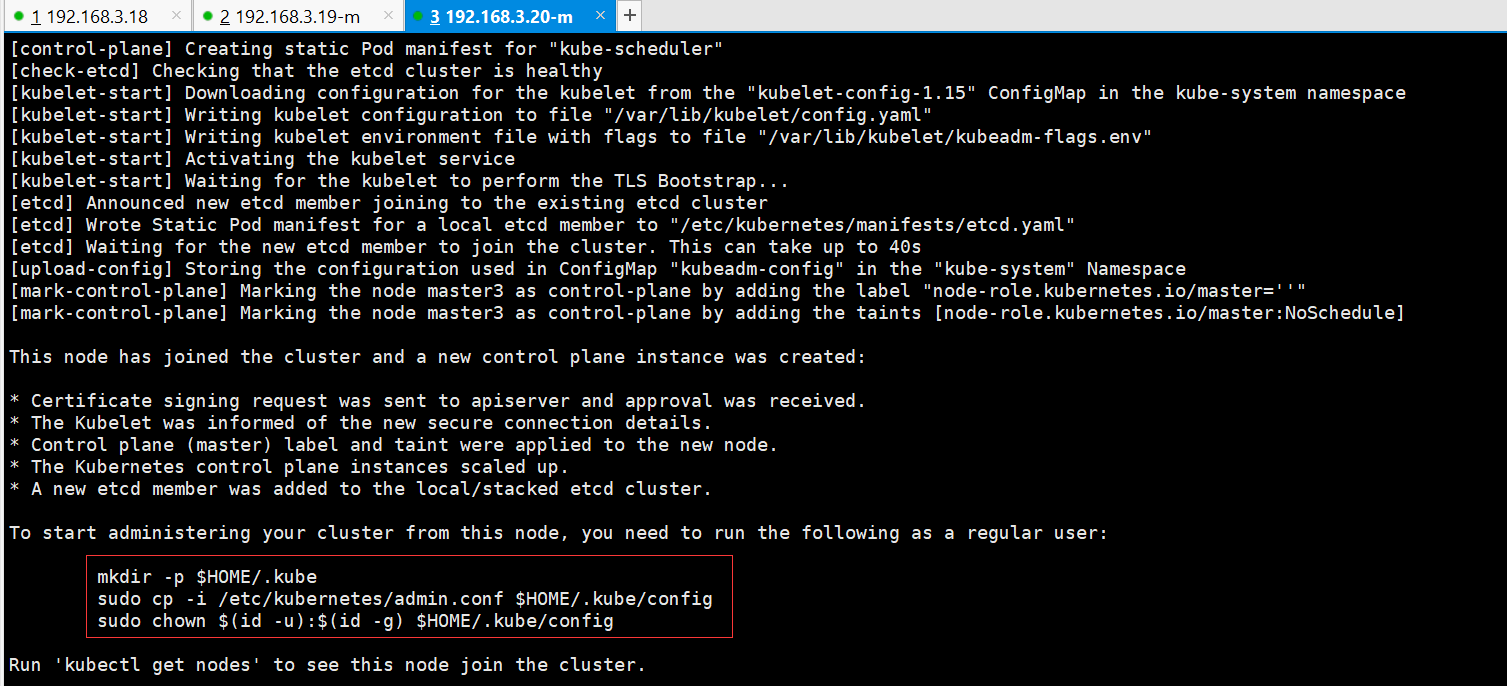
## 8.4初始化其他master节点

使用命令

kubeadm join 192.168.3.100:8443 --token rv4daf.44jisgiwzg7wp07c --discovery-token-ca-cert-hash sha256:c947ffc92b679603c51279d843e6a0657d48b7c60644e539dba15dc1246e6f59 --experimental-control-plane

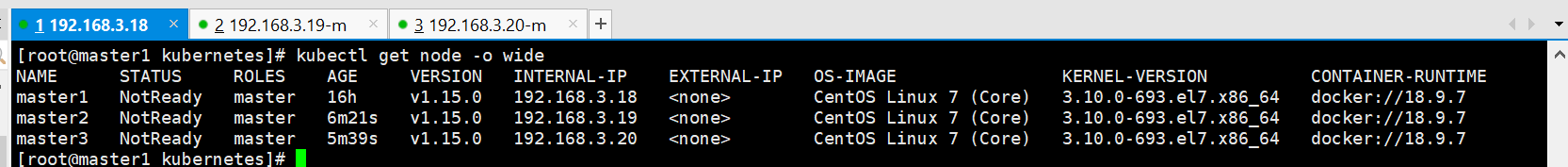
接着等待一段时间，初始化的时候需要从网上拉取镜像，直到我们看见

This node has joined the cluster ...则证明我们其他master节点成功加入到集群，加入集群后我们依然需要执行图中画红线的部分，执行后才可以使用kubectl直接和api-server交互。



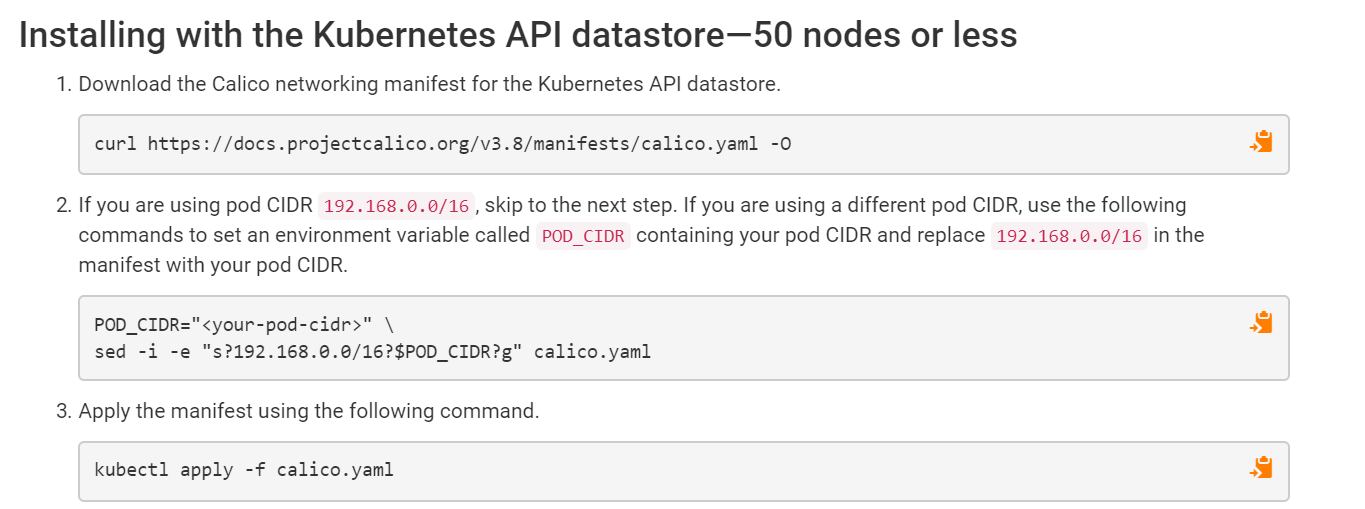
## 8.5验证集群环境

我们使用kubectl get node [-o wide]查看集群状态，可以看到当前已经有3个节点了，都是master角色，但是状态都是NotReady状态，别着急，我们目前为止还没有安装网络组件。



安装calico网络组件，生产集群大的时候推荐calico,当然你可以使用其他的flannel canal等其他的网络组件。官方介绍如下：

详情可以根据情况查看官网介绍，连接地址：<https://docs.projectcalico.org/v3.8/getting-started/kubernetes/installation/calico>



步骤：

1.下载yaml

curl https://docs.projectcalico.org/v3.8/manifests/calico.yaml -O

2.修改网络地址跟pod网段保持一致

我们的第二步骤跟官方不一样，但是是一个意思，都是修改网络地址段

修改calico.yaml的192.168.0.0/24 改为初始化的pod网段地址，我们之前使用的10.244.0.0/16

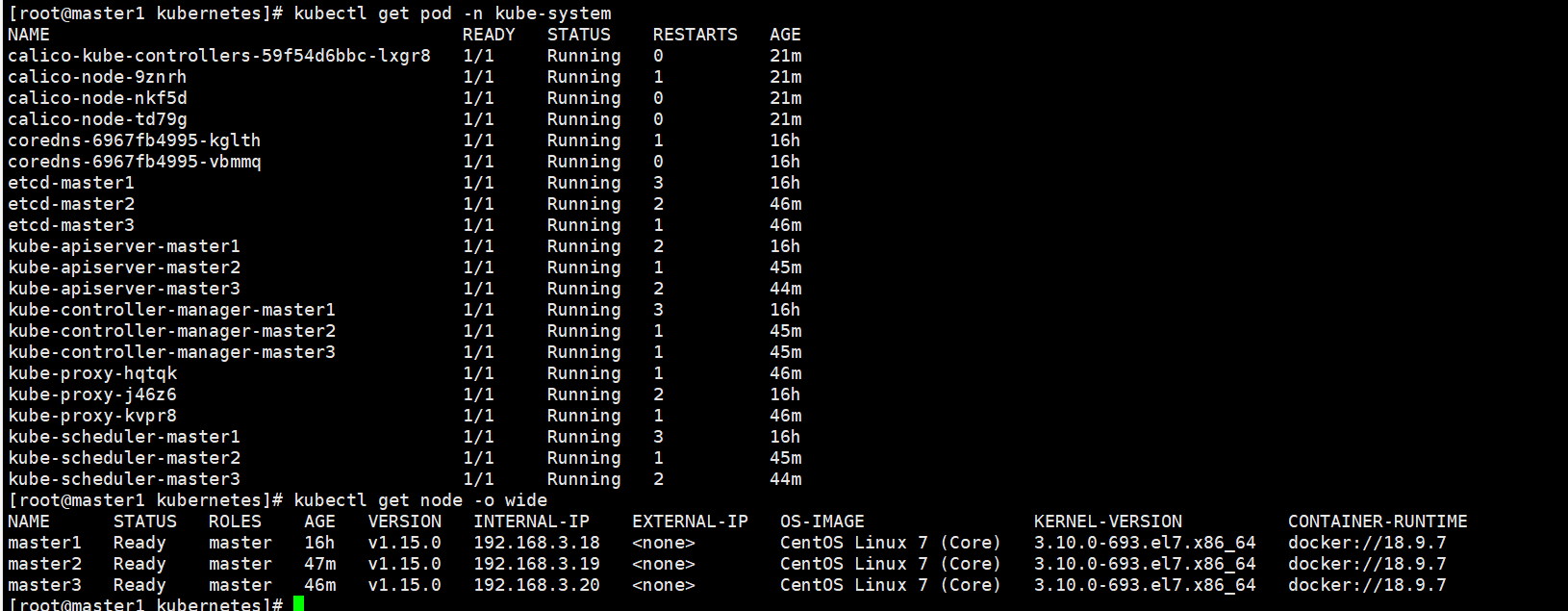
3.创建calico网络pod

kubectl apply -f calico.yaml

创建后等待段时间后，这个根据网络环境等待，下载镜像，启动pod

直到我们使用kubectl get pod -n kube-system的pod全部变成n/n的时候集群就算搭建完成。

等待一段时间后，我们再次使用kubectl get node [-o wide]查看集群状态信息，可以看到所有的pod已经成功启动，3个master节点变成了Ready状态

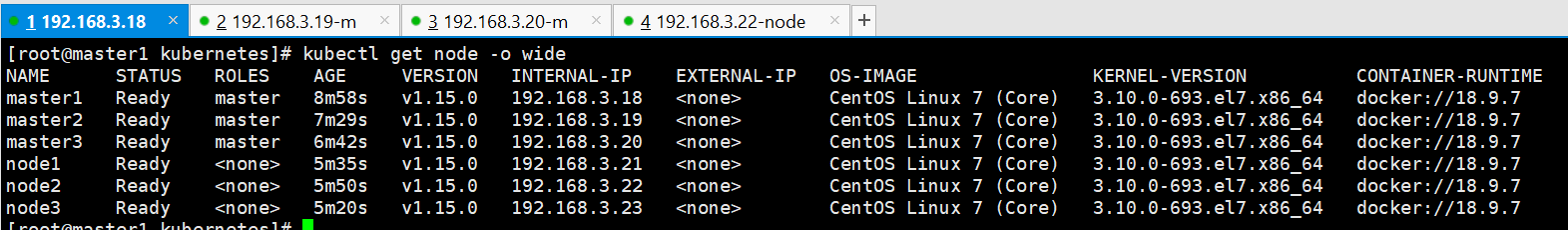


## 8.6加入node节点

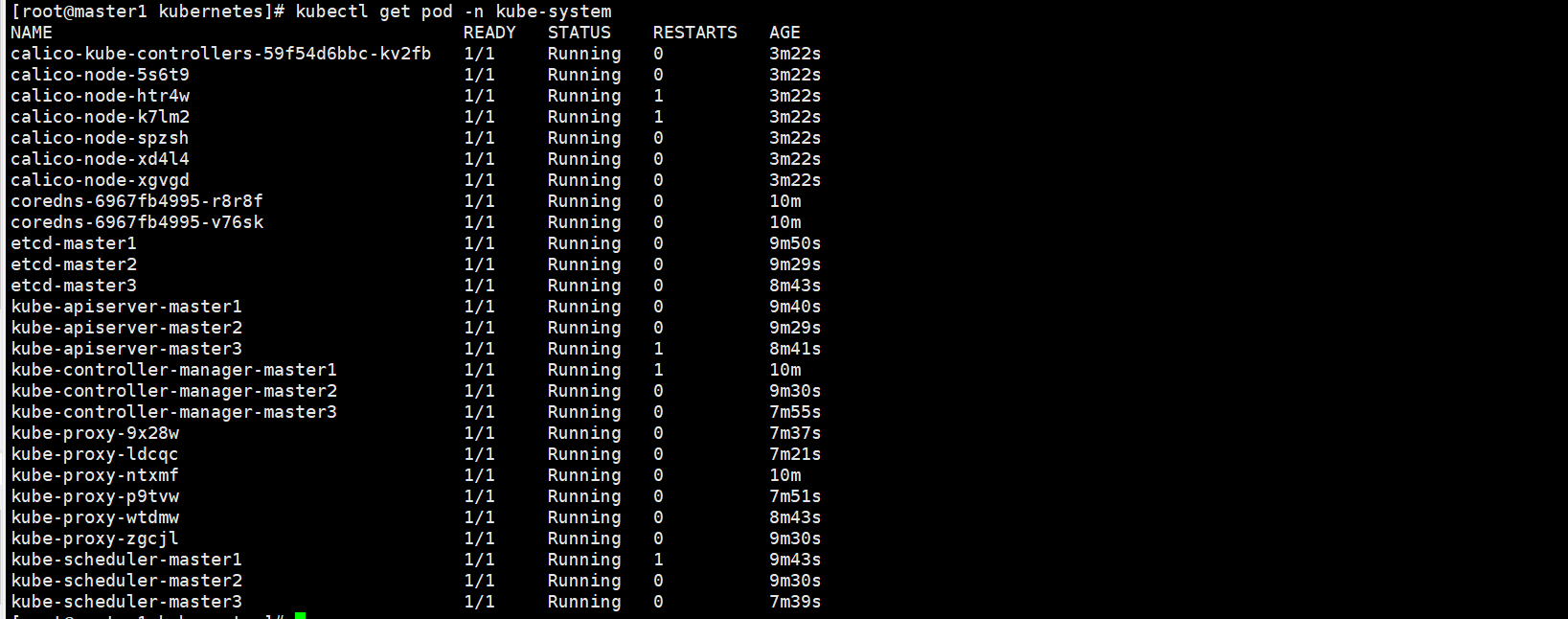
这里我们使用直接加入到集群环境即可，注意：node节点没有后面的--experimental-control-plane参数。

|  |
| --- |
| kubeadm join 192.168.3.100:8443 --token rv4daf.44jisgiwzg7wp07c --discovery-token-ca-cert-hash sha256:c947ffc92b679603c51279d843e6a0657d48b7c60644e539dba15dc1246e6f59 |

三个节点都执行加入集群的命令后，稍等一下即可看到，此时我们的master和node都处于Ready状态。



并且此时我们kube-system名称空间下的所有pod都处于n/n的状态，此时我们的k8s多master高可用环境搭建完成。



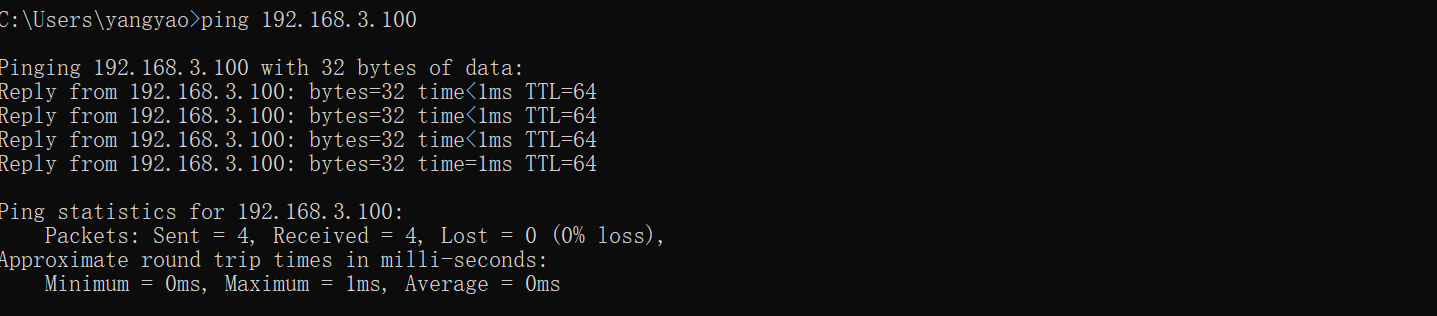
# 9.注意事项

我们的高可用目前环境是三个master,因为etcd集群至少要保证2个节点，所以当我们挂了一个master节点时是对于我们高可用环境是不受影响的，生产环境至少要保证三个master节点以上即可。

# 10.外部如何访问

因为我们集群环境使用的keepalived生成并绑定了vip，这个vip是会飘走的。举个例子，现在vip绑定到192.168.3.18的ens32网卡上面，当我们这个服务器宕机，keepalived的高可用机制可能就把vip飘到其他keepalived上面。

所以，外部访问直接访问我们的vip即可。为了测试，我们在自己电脑上启动一个cmd,去ping我们这个vip看是否有返回结果。



可以看到，我们在集群外部只需要访问我们的vip即可。