# 配置HOSTS(手工修改)

|  |
| --- |
| cat /etc/hosts  #根据自己的机器数量决定，红色为master **192.168.8.81 vm81** 192.168.8.82 vm82 192.168.8.83 vm83 |

# 2.配置DNS(手工修改) (所有机器)

|  |
| --- |
| ##配置dns  [root@vm81 manifests]# cat /etc/resolv.conf  nameserver 114.114.114.114 |

# 3.安装配置yum源(所有机器)

|  |
| --- |
| curl -o /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-7.repo  yum install -y yum-utils device-mapper-persistent-data lvm2  yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo  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  sed -i -e '/mirrors.cloud.aliyuncs.com/d' -e '/mirrors.aliyuncs.com/d' /etc/yum.repos.d/CentOS-Base.repo  yum install wget jq psmisc vim net-tools telnet yum-utils device-mapper-persistent-data lvm2 git -y  systemctl disable --now firewalld  systemctl disable --now dnsmasq  systemctl disable --now NetworkManager  setenforce 0  sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/sysconfig/selinux  sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/selinux/config  swapoff -a && sysctl -w vm.swappiness=0  sed -ri '/^[^#]\*swap/s@^@#@' /etc/fstab  ulimit -SHn 65535  echo '\* soft nofile 65536' >>/etc/security/limits.conf  echo '\* hard nofile 131072' >>/etc/security/limits.conf  echo '\* soft nproc 65535' >>/etc/security/limits.conf  echo '\* hard nproc 655350' >>/etc/security/limits.conf  echo '\* soft memlock unlimited' >>/etc/security/limits.conf  echo '\* hard memlock unlimited' >>/etc/security/limits.conf |

# 4升级系统(所有机器)

|  |
| --- |
| ## 升级系统  yum update -y --exclude=kernel\* && reboot |

## 升级内核(所有机器)

执行完后重启看内核是否已经是4.19版本，如果失败则重新安装一次

|  |
| --- |
| cd /root  wget http://193.49.22.109/elrepo/kernel/el7/x86\_64/RPMS/kernel-ml-devel-4.19.12-1.el7.elrepo.x86\_64.rpm  wget http://193.49.22.109/elrepo/kernel/el7/x86\_64/RPMS/kernel-ml-4.19.12-1.el7.elrepo.x86\_64.rpm  cd /root && yum localinstall -y kernel-ml\*  grub2-set-default 0 && grub2-mkconfig -o /etc/grub2.cfg  grubby --args="user\_namespace.enable=1" --update-kernel="$(grubby --default-kernel)"  reboot |

## 启用IPVS(所有机器)

|  |
| --- |
| ## 启用ipvs  yum install ipvsadm ipset sysstat conntrack libseccomp -y  modprobe -- ip\_vs  modprobe -- ip\_vs\_rr  modprobe -- ip\_vs\_wrr  modprobe -- ip\_vs\_sh  modprobe -- nf\_conntrack    echo 'ip\_vs' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_lc' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_wlc' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_rr' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_wrr' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_lblc' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_lblcr' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_dh' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_sh' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_fo' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_nq' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_sed' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_ftp' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_vs\_sh' >>/etc/modules-load.d/ipvs.conf    echo 'nf\_conntrack' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_tables' >>/etc/modules-load.d/ipvs.conf    echo 'ip\_set' >>/etc/modules-load.d/ipvs.conf    echo 'xt\_set' >>/etc/modules-load.d/ipvs.conf    echo 'ipt\_set' >>/etc/modules-load.d/ipvs.conf    echo 'ipt\_rpfilter' >>/etc/modules-load.d/ipvs.conf    echo 'ipt\_REJECT' >>/etc/modules-load.d/ipvs.conf    echo 'ipip' >>/etc/modules-load.d/ipvs.conf  systemctl enable --now systemd-modules-load.service |

## 配置内核参数(所有机器)

|  |
| --- |
| ## 配置内核参数  cat <<EOF > /etc/sysctl.d/k8s.conf  net.ipv4.ip\_forward = 1  net.bridge.bridge-nf-call-iptables = 1  net.bridge.bridge-nf-call-ip6tables = 1  fs.may\_detach\_mounts = 1  net.ipv4.conf.all.route\_localnet = 1  vm.overcommit\_memory=1  vm.panic\_on\_oom=0  fs.inotify.max\_user\_watches=89100  fs.file-max=52706963  fs.nr\_open=52706963  net.netfilter.nf\_conntrack\_max=2310720  net.ipv4.tcp\_keepalive\_time = 600  net.ipv4.tcp\_keepalive\_probes = 3  net.ipv4.tcp\_keepalive\_intvl =15  net.ipv4.tcp\_max\_tw\_buckets = 36000  net.ipv4.tcp\_tw\_reuse = 1  net.ipv4.tcp\_max\_orphans = 327680  net.ipv4.tcp\_orphan\_retries = 3  net.ipv4.tcp\_syncookies = 1  net.ipv4.tcp\_max\_syn\_backlog = 16384  net.ipv4.ip\_conntrack\_max = 65536  net.ipv4.tcp\_max\_syn\_backlog = 16384  net.ipv4.tcp\_timestamps = 0  net.core.somaxconn = 16384  EOF  sysctl --system  reboot |

## 查看内核配置是否生效

|  |
| --- |
| lsmod | grep --color=auto -e ip\_vs -e nf\_conntrack |

# 5安装并配置docker(所有机器)

|  |
| --- |
| yum install docker-ce-19.03.\* docker-ce-cli-19.03.\* -y  ## 配置cgroups驱动  mkdir /etc/docker  cat > /etc/docker/daemon.json <<EOF  {  "exec-opts": ["native.cgroupdriver=systemd"]  }  EOF  systemctl daemon-reload && systemctl enable --now docker |

# 6安装K8S(所有机器)

|  |
| --- |
| ##安装k8s  yum install kubeadm-1.22\* kubelet-1.22\* kubectl-1.22\* -y  ## 可选配置，配置阿里云的pause镜像  cat >/etc/sysconfig/kubelet<<EOF  KUBELET\_EXTRA\_ARGS="--pod-infra-container-image=registry.cn-hangzhou.aliyuncs.com/google\_containers/pause:3.5"  EOF  systemctl daemon-reload  systemctl enable --now kubelet |

## 集群初始化(根据情况替换配置，仅master)

**我本机使用了三台，master为8.81**

|  |
| --- |
| ## 集群初始化  cat >/EOF>/root/cfg.yaml <<EOF  apiVersion: kubeadm.k8s.io/v1beta2  bootstrapTokens:  - groups:  - system:bootstrappers:kubeadm:default-node-token  token: 7t2weq.bjbawausm0jaxury  ttl: 24h0m0s  usages:  - signing  - authentication  kind: InitConfiguration  localAPIEndpoint:  advertiseAddress: **192.168.8.81**  bindPort: 6443  nodeRegistration:  criSocket: /var/run/dockershim.sock  name: vm81  taints:  - effect: NoSchedule  key: node-role.kubernetes.io/master  ---  apiServer:  certSANs:  **- vm82**  **- vm83**  **- vm81**  timeoutForControlPlane: 4m0s  apiVersion: kubeadm.k8s.io/v1beta2  certificatesDir: /etc/kubernetes/pki  clusterName: kubernetes  controlPlaneEndpoint: 192.168.8.81:6443  controllerManager: {}  dns:  type: CoreDNS  etcd:  local:  dataDir: /var/lib/etcd  imageRepository: registry.cn-hangzhou.aliyuncs.com/google\_containers  kind: ClusterConfiguration  kubernetesVersion: v1.22.0  networking:  dnsDomain: cluster.local  podSubnet: 172.16.0.0/12  serviceSubnet: 192.168.0.0/16  scheduler: {}  EOF |

## 配置K8S(仅master)

首先将配置文件复制到/root目录下

|  |
| --- |
| ##配置k8s  kubeadm config migrate --old-config cfg.yaml --new-config new.yaml  ##提前下载镜像  kubeadm config images pull --config /root/new.yaml  ### master初始化  kubeadm init --config /root/new.yaml --upload-certs |

**执行完初始化，会打印出要执行的命令：其中蓝色背景的需要在当前master上执行，粉色背景的需要在其他的master节点执行。**

|  |
| --- |
| *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* |

如果初始化失败或者初始化过程中有报错，可以重置：

|  |
| --- |
| Kubectl reset |

## 配置worker(仅worker)

根据master打出的命令，亮蓝色背景，在每个worker节点上执行：

|  |
| --- |
| *kubeadm join 192.168.1.201:6443 --token zlfmmm.a41tyorwikg336fx \*  *--discovery-token-ca-cert-hash sha256:44f5622e441e88e172a103f084ea150e62b2a5cdd11cb6fb65f314a0ac92fb9a* |

# 7安装插件

安装必备安装的所有插件

|  |
| --- |
| Kubectl apply -f …  #calico，metric-server,dashboard |

通过以下命令获取登录dashboard的token（非必须，可选）

|  |
| --- |
| kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}') |

# 8故障排查

### 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：

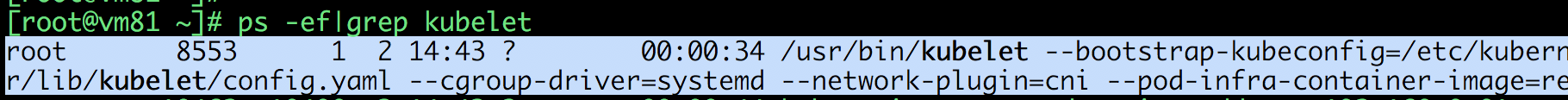
图形用户界面, 文本, 聊天或短信

描述已自动生成

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：

文本

描述已自动生成

如果node没有处于ready状态，检查pod：

文本

描述已自动生成

### 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：

文本, 聊天或短信

描述已自动生成

# 咨询

QR 代码

描述已自动生成