1.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)

```
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)"
```

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 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：

```
[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/kuberr
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 14:43:06.238
381f269cfd3937eff9fa80d97f2aeb4755597a7702be3585b931f085"
8月 01 14:43:06 vm81 kubelet[8553]: 2020-08-01 14:43:06.240
7eff9fa80d97f2aeb4755597a7702be3585b931f085" host="vm81"
8月 01 14:43:06 vm81 kubelet[8553]: 2020-08-01 14:43:06.240
```

故障排查咨询