安装记录

服务器配置信息

```
虚拟机：  VM14
系统：  Centos7
master  k8s1 4G 4Core  192.168.233.129
node1   k8s2 4G 4Core  192.168.233.130
node2   k8s3 4G 4Core  192.168.233.131
```

1、设置主机名

```
hostnamectl set-hostname k8s1
hostnamectl set-hostname k8s2
```

2、修改host文件

```
cat >> /etc/hosts <<EOF
192.168.233.129  k8s1
192.168.233.130  k8s2
192.168.233.131  k8s3
EOF
```

3、关闭selinux

```
setenforce 0 #实时动态关闭
sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config #禁止重启后自动开启
```

4、关闭交换分区

```
swapoff -a #实时动态关闭
sed -i '/ swap / s/^/#/' /etc/fstab #禁止重启后自动开启
```

5、网络配置文件

```
cat >> /etc/sysctl.d/k8s.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
vm.swappiness=0
EOF

modprobe br_netfilter  #执行该命令 如果不执行就会在应用k8s.conf时出现加载错误

sysctl -p /etc/sysctl.d/k8s.conf #应用配置文件
```

6、配置资源地址

```
yum install wget -y  #安装wget命令行

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo  #配置yum源

yum makecache #更新缓存

yum install -y yum-utils device-mapper-persistent-data lvm2 #安装yum扩展工具

yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo #配置docker下载的地址
```

7、安装Docker

```
yum list docker-ce --showduplicates|sort -r  #展示版本列表

yum install -y docker-ce  #默认安装最新版，也可以指定版本下载

systemctl start docker #启动docker
systemctl enable docker #将docker加入到开机启动

docker version #查看docker启动情况 和版本信息
```

8、配置k8s资源的下载地址

```
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
```

9、安装 kubelet kubeadm kubectl

```
yum install -y kubelet-1.15.1 kubeadm-1.15.1 kubectl-1.15.1
systemctl enable kubelet.service #开机启动
systemctl enable kubectl.service
```

10、查看kubeadm config所需的镜像

```
kubeadm config images list

k8s.gcr.io/kube-apiserver:v1.15.2
k8s.gcr.io/kube-controller-manager:v1.15.2
k8s.gcr.io/kube-scheduler:v1.15.2
k8s.gcr.io/kube-proxy:v1.15.2
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.3.10
k8s.gcr.io/coredns:1.3.1
```

11、使用github与dockerhub制作以上镜像

```
#由于外网的限制，且没有科学上网的条件，所以通过DockerHub拉取指定k8s所需镜像，具体方法为使用gitHub帐户关联DockerHub制作df利用dokcerhub的自动构建拉取我们k8s需要的镜像文件
#制作DockerFile
From k8s.gcr.io/kube-apiserver:v1.15.1
MAINTAINER yanjy <1207778252@qq.com>
```

12、拉取镜像

```
docker pull 1207778252/kube-proxy:latest
docker pull 1207778252/kube-apiserver:latest
docker pull 1207778252/kube-controller-manager:latest
docker pull 1207778252/kube-scheduler:latest
docker pull 1207778252/pause:latest
docker pull 1207778252/etcd:latest
docker pull 1207778252/coredns:latest
#打新标签
docker tag 1207778252/kube-proxy:latest k8s.gcr.io/kube-proxy:v1.15.2
docker tag 1207778252/kube-apiserver:latest k8s.gcr.io/kube-apiserver:v1.15.2
docker tag 1207778252/kube-controller-manager:latest k8s.gcr.io/kube-controller-manager:v1.15.2
docker tag 1207778252/kube-scheduler:latest k8s.gcr.io/kube-scheduler:v1.15.2
docker tag 1207778252/pause:latest k8s.gcr.io/pause:3.1
docker tag 1207778252/etcd:latest k8s.gcr.io/etcd:3.3.10
docker tag 1207778252/coredns:latest k8s.gcr.io/coredns:1.3.1
```

```
#关闭防火墙
firewall-cmd --state
systemctl disable firewalld.service
```

13、开始初始化 （只在主节点Master上面 操作）

```
#只在Master的主机上面执行 版本信息与你要安装的相同
kubeadm init --kubernetes-version=v1.15.2 --pod-network-cidr=192.244.0.0/16 --
apiserver-advertise-address=192.168.233.129

#当出现 类似 如下说明 master 安装成功
kubeadm join 192.168.233.129:6443 --token r6vo99.ixkb27zv3o9e154m \
    --discovery-token-ca-cert-hash
sha256:0a2641c6af4e107c83022372bade060e526cb64aa768e9d15968ffe84ad9dcd3

#然后执行 安装成功提示的 命令行  这部分执行你安装成功后的部分 可能每个人的有所不同
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
#执行init后输出的日志
[init] Using Kubernetes version: v1.15.2
[preflight] Running pre-flight checks
        [WARNING Firewalld]: firewalld is active, please ensure ports [6443
10250] are open or your cluster may not function correctly
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup
driver. The recommended driver is "systemd". Please follow the guide at
https://kubernetes.io/docs/setup/cri/
        [WARNING SystemVerification]: this Docker version is not on the list of
validated versions: 19.03.1. Latest validated version: 18.09
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your
internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
images pull'
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [k8s1 kubernetes
kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local]
and IPs [10.96.0.1 192.168.233.129]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [k8s1 localhost] and
IPs [192.168.233.129 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8s1 localhost] and IPs
[192.168.233.129 127.0.0.1 ::1]
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
```

```
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in
"/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as
static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 20.506985 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in
the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.15" in namespace kube-system
with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node k8s1 as control-plane by adding the label
"node-role.kubernetes.io/master=''"
[mark-control-plane] Marking the node k8s1 as control-plane by adding the taints
[node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: r6vo99.ixkb27zv3o9e154m
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC
Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post
CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller
automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all
node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public"
namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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/

Then you can join any number of worker nodes by running the following on each as
root:

kubeadm join 192.168.233.129:6443 --token r6vo99.ixkb27zv3o9e154m \
    --discovery-token-ca-cert-hash
sha256:0a2641c6af4e107c83022372bade060e526cb64aa768e9d15968ffe84ad9dcd3
```

14、节点node加入（默认此事上面一步骤 都进行了操作）

```
# 执行master 安装成功后的 kubeadm join命令 注意是你自己的，下面是举例
kubeadm join --token r6vo99.ixkb27zv3o9e154m 192.168.233.129:6443 --discovery-token-ca-cert-hash
sha256:0a2641c6af4e107c83022372bade060e526cb64aa768e9d15968ffe84ad9dcd3
```

15、node 节点删除

```
 kubectl drain k8s3 --delete-local-data --force --ignore-daemonsets
 kubectl delete node k8s3
```

16、主节点查看命令

```
#命令一
kubectl get cs #显示内容如下说明Master安装没问题
NAME                 STATUS    MESSAGE              ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health": "true"}

#命令二
kubectl get pod -n kube-system #查看pod状态 下面只是部分pod内容
NAME                                 READY   STATUS             RESTARTS   AGE
coredns-86c58d9df4-j9g8d             1/1     Running            0
128m
coredns-86c58d9df4-pg45w             1/1     Running            0
128m
etcd-k8s1                            1/1     Running            0
127m
kube-apiserver-k8s1                  1/1     Running            0
127m
kube-controller-manager-k8s1         1/1     Running            0
127m


#在这里你可能发现你的coredns 的状态并不是Running 不要着急，后面还有个配置，配置好就自动Running了

#命令三
kubectl get node #查看节点状态
NAME   STATUS   ROLES    AGE    VERSION
k8s1   Ready    master   131m   v1.13.4
k8s2   Ready    <none>   93m    v1.13.4

#如果你添加了节点里面看的话 可能还未初始化，显示的是NoReady多等会儿。
```

17、安装Flannel 网络配置工具,用于配置第三层(网络层)网络结构 (只需要Master安装)

```
#如果是国内 通过执行yml无法直接下载的话执行下面命令
docker pull registry.cn-shenzhen.aliyuncs.com/cp_m/flannel:v0.10.0-amd64
docker tag registry.cn-shenzhen.aliyuncs.com/cp_m/flannel:v0.10.0-amd64
quay.io/coreos/flannel:v0.10.0-amd64
docker rmi registry.cn-shenzhen.aliyuncs.com/cp_m/flannel:v0.10.0-amd64

#安装Flannel

mkdir -p ~/k8s/
cd k8s/
ll

wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml

kubectl apply -f kube-flannel.yml

#安装成功后查看pod
kubectl get pod -n kube-system
NAME                                    READY   STATUS             RESTARTS   AGE
coredns-86c58d9df4-j9g8d                1/1     Running            0
128m
coredns-86c58d9df4-pg45w                1/1     Running            0
128m
etcd-k8s1                               1/1     Running            0
127m
kube-apiserver-k8s1                     1/1     Running            0
127m
kube-controller-manager-k8s1            1/1     Running            0
127m
kube-flannel-ds-amd64-7btlw             1/1     Running            0          91m
kube-flannel-ds-amd64-9vq42             1/1     Running            0
106m
kube-flannel-ds-amd64-kdf42             1/1     Running            0          90m
kube-proxy-dtmfs                        1/1     Running            0
128m
kube-proxy-p76tc                        1/1     Running            0          90m
kube-proxy-xgw28                        1/1     Running            0          91m
kube-scheduler-k8s1                     1/1     Running            0
128m

#全部Running则表示 成功了

#如果发现哪一项的STATUS的状态不是Running,执行如下命令
kubectl describe pod  [这里是复制上面的Name列] -n kube-system

#这里会看到具体的错误内容，然后根据提示进行解决。
```