1.k8s部署

环境初始化，所有节点

设置集群命令补全

source <(kubectl completion bash)

echo source <(kubectl completion bash) >> ~/.bashrc

1.1配置静态ip

```
vim /etc/sysconfig/network-script/ifcfg-ens33
```

1.2配置hostname

```
hostnamectl set-hostname master
hostnamectl set-hostname node1
hostnamectl set-hostname node2
```

1.3配置/etc/hosts

```
vim /etc/hosts
192.168.126.136 master
192.168.126.137 node1
192.168.126.138 node2
```

1.4关闭防火墙、selinux、swap

```
#关闭防火墙
systemctl disable firewalld
systemctl mask firewalld
#关闭selinux
sed -i 's/SELINUX\=enforcing/SELINUX=disabled/' /etc/selinux/config
关闭Swap，关闭swap分区，防止启动k8s报错
swapoff -a
sed -i 's/.*swap.*/#&/' /etc/fstab
```

1.5更新安装软件包

```
#更新安装软件包
yum update && yum install vim gcc gcc-c++ bash-com\*  wget curl net-tools git dnf epel\* co
```

1.6kernel升级

```
#首先导入elrepo的公钥
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
#接着安装elrepo源
yum install https://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
#安装最新版内核
yum --enablerepo='elrepo-kernel' install kernel-ml
#将最新内核设置为默认启用
grub2-set-default 0
grub2-mkconfig -o /etc/grub2.cfg

升级内核后重启
reboot
```

1.7时钟同步

```
systemctl restart chronyd
systemctl enable chronyd
chronyc  soureces
```

1.8开启IPV4转发和网桥防火墙支持(配置内核参数)

```
echo 'net.ipv4.ip_forward = 1' >> /etc/sysctl.conf
echo 'net.bridge.bridge-nf-call-ip6tables = 1'  >> /etc/sysctl.conf
echo 'net.bridge.bridge-nf-call-iptables = 1' >> /etc/sysctl.conf
echo 'net.bridge.bridge-nf-call-arptables = 1' >> /etc/sysctl.conf
#生效文件
sysctl -p /etc/sysctl.conf
```

1.9修改内核，增加k8s支持，不满足条件的情况下，k8s启动会报错

```
cat >> /etc/security/limits.conf << EOF
> *               soft    nofile          655360
> *               hard    nofile          655360
> *               soft    noproc          655360
> *               hard    noproc           655360
> *               soft    memlock         unlimited
> *               hard    memlock         unlimited
> EOF
```

1.10添加k8s的阿里源文件

```
vim /etc/yum.repos.d/k8s.repo
[kubernetes]
name=kubernetes.repo
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
gpgcheck=0
enabled=1
```

1.11安装docker-ce

```
#添加docker-ce源文件
wget -O /etc/yum.repos.d/docker-ce.repo http://mirrors.aliyun.com/docker-ce/linux/centos/do
或者执行
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

#安装k8s（管理节点安装）
yum install docker-ce-18.06.3.ce-3.el7 kubectl kubeadm kubelet
 #安装k8s（节点安装）
yum install docker-ce-18.06.3.ce-3.el7 kubectl  kubeadm kubelet
```

- kubeadm：部署集群用的命令
- kubelet：在集群中每台机器上都要运行的组件,负责管理pod、容器的生命周期
- kubectl：集群管理工具

1.12docker镜像加速和启动

```
mkdir -p /etc/docker
tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://z3p4xkf4.mirror.aliyuncs.com"]
```

```
}
EOF

#重载
sudo systemctl daemon-reload
#重启docker
sudo systemctl restart docker
#设置开机自启
systemctl enable docker
#设置kubelet开机自启,在所有节点上执行
systemctl enable kubelet
systemctl restart kubelet
```

2镜像下载准备

2.1初始化获取要下载的镜像列表

```
#查看依赖需要安装的镜像列表
kubeadm config images list
#生成k8s工作所需要的镜像部署配置文件
kubeadm config print init-defaults > kubeadm.conf
#修改默认的仓库地址为阿里云的仓库地址
sed -i "s/imageRepository: .*/imageRepository: registry.aliyuncs.com\/google_containers/g"
#指定kubeadm安装的Kubernetes版本
sed -i "s/kubernetesVersion: .*/kubernetesVersion: v1.13.0/g" kubeadm.conf
#下载k8s运行所需基础镜像
kubeadm config images pull --config /root/kubeadm.conf
#查看镜像
docker images
```

2.2处理镜像名称（在全部节点上运行）

```
docker tag registry.aliyuncs.com/google_containers/kube-proxy:v1.14.0 k8s.gcr.io/kube-proxy
docker tag registry.aliyuncs.com/google_containers/kube-apiserver:v1.14.0 k8s.gcr.io/kube-a
docker tag registry.aliyuncs.com/google_containers/kube-controller-manager:v1.14.0 k8s.gcr.
docker tag registry.aliyuncs.com/google_containers/kube-scheduler:v1.14.0 k8s.gcr.io/kube-s
docker tag registry.aliyuncs.com/google_containers/coredns:1.3.1 k8s.gcr.io/coredns:1.3.1
docker tag registry.aliyuncs.com/google_containers/etcd:3.3.10 k8s.gcr.io/etcd:3.3.10
docker tag registry.aliyuncs.com/google_containers/pause:3.1 k8s.gcr.io/pause:3.1
docker rmi registry.aliyuncs.com/google_containers/pause:3.1
docker rmi registry.aliyuncs.com/google_containers/etcd:3.3.10
docker rmi registry.aliyuncs.com/google_containers/coredns:1.3.1
```

```
docker rmi registry.aliyuncs.com/google_containers/kube-scheduler:v1.14.0
docker rmi registry.aliyuncs.com/google_containers/kube-controller-manager:v1.14.0
docker rmi  registry.aliyuncs.com/google_containers/kube-apiserver:v1.14.0
docker rmi registry.aliyuncs.com/google_containers/kube-proxy:v1.14.0
```

2.3初始化集群

```
#初始化集群，主要指定apiserver所在的地址和k8s版本,定义POD的网段为: 172.22.0.0/16 ,api server地址
kubeadm init --kubernetes-version=v1.14.0  --pod-network-cidr=10.224.0.0/16 --apiserver-adv
#如果需要可以 执行下面的命令重新初始化
kubeadm reset
kubeadm init --kubernetes-version=v1.13.0 --pod-network-cidr=172.22.0.0/16 --apiserver-adve
在此操作后，提醒操作流程（初始化成功后的操作），请记录加入集群的令牌和哈希值
```

2.4初始化成功后操作

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

2.5下载网络组件

```
#下载网络组件，选择的是calico方案
#运行在k8s集群中的每一个节点上，主要用于实现路由（基于BGP的bird）
docker pull calico/node:v3.1.4
#部署在每个节点上，用于管理pod
docker pull calico/cni:v3.1.4
#实现k8s数据存储模式，帮助扩展节点使用
docker pull calico/typha:v3.1.4
```

2.6批量重名calico网络组件基础镜像

```
docker tag calico/node:v3.1.4  quay.io/calico/node:v3.1.4
docker rmi calico/node:v3.1.4
docker tag calico/cni:v3.1.4  quay.io/calico/cni:v3.1.4
docker rmi calico/cni:v3.1.4
docker tag calico/typha:v3.1.4 quay.io/calico/typha:v3.1.4
docker rmi calico/typha:v3.1.4
```

2.7calico配置

```
下载calico配置
curl https://docs.projectcalico.org/v3.1/gettingstarted/kubernetes/installation/hosted/kube
部署calico
curl https://docs.projectcalico.org/v3.1/gettingstarted/kubernetes/installation/hosted/rbac
#修改calico的配置/root/calico.yaml，大概17行左右
typha_service_name: "calico-typha"
#修改calico的配置，大概89行左右,启用副本，数量为1，默认不启动
replicas: 1
#启用网络类型为bgp中的bird模式，211行
value: "bird"
#修改pod网络范围和集群中保持一致，大概在231行
value: "10.224.0.0/16"
```

calico.yaml注意镜像版本一致：所以可能需要修改镜像版本

2.8部署master节点上的calico网络（master节点上运行）

```
kubectl apply -f rbac-kdd.yaml
kubectl apply -f calico.yaml
```

2.9将工作节点加入集群（在node节点运行）

```
kubeadm join 192.168.126.136:6443 --token js691f.3h69scvdiljz3325 \
--discovery-token-ca-cert-hash sha256:194bc33da749c082a42a7c36c498dab4df423bb66b551fc34a505
```

2.10加入节点验证

```
#在master管理节点查看节点是否准备
kubectl get nodes
#查看集群的健康状态
kubectl get cs
#执行获取pods列表命令,查看相关状态
kubectl get pods --all-namespaces
```

3.部署dashboard

3.1生成私钥和证书签名请求

```
mkdir -p /etc/kubernetes/certs
cd /etc/kubernetes/certs
openssl genrsa -des3 -passout pass:x -out dashboard.pass.key 2048 openssl rsa -passin pass:
#删除刚才生成的dashboard.pass.key
rm -rf dashboard.pass.key

openssl req -new -key dashboard.key -out dashboard.csr

 #生成SSL证书
openssl x509 -req -sha256 -days 365 -in dashboard.csr -signkey dashboard.key -out dashboard
```

3.2创建secret

```
kubectl create secret generic kubernetes-dashboard-certs --from-file=/etc/kubernetes/certs
```

3.3下载并且修改dashboard镜像（在所有节点运行）

```
docker pull registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0
docker tag registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0 k
docker rmi registry.cn-hangzhou.aliyuncs.com/kubernete/kubernetes-dashboard-amd64:v1.10.0
```

3.4下载 kubernetes-dashboard.yaml 部署文件(在master上执行)

```
 #下载kubernetes-dashboard.yaml配置文件
 wget https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended
 更改配置文件，修改更改的镜像文件名，或者与物理的端口映射
 vim kubernetes-dashboard.yaml
 type: NodePort
 nodePort: 30005

#创建dashboard的pod组件
kubectl create -f kubernetes-dashboard.yaml
```

3.5查看服务器的运行状态。（master）

```
kubectl get deployment kubernetes-dashboard -n kube-system
kubectl --namespace kube-system get pods -o wide
kubectl get services kubernetes-dashboard -n kube-system
netstat -ntlp|grep 30005
```

3.6 dashboard web 验证

验证方式：
在物理机浏览器中输入：
https://nodeip:nodeport
如果发现不是私密连接。那可以试着换一个浏览器，在chrome在实验现实就是不是私密链接。
不是私密链接的解决办法：

1、查看kubernetes-dashboard 容器跑在哪台node节点上，这里跑在docker-slave2上
kubectl get pod -n kube-system -o wide
2、在docker-slave2节点上查看kubernetes-dashboard容器ID
docker ps | grep dashboard
3、查看kubernetes-dashboard容器certs所挂载的宿主主机目录
docker inspect -f {{.Mounts}} 384d9dc0170b
4、将生成的dashboard.crt dashboard.key放到certs对应的宿主主机souce目录
scp dashboard.crt dashboard.key 192.168.20.214:/var/lib/kubelet/pods/94c8c50b-f484-11e8-80e8-000c29c3dca5/volumes/kubernetes.io~secret/kubernetes-dashboard-certs
5、重启kubernetes-dashboard容器
docker restart 384d9dc0170b
6、再次web验证
详细参考：
https://www.cnblogs.com/harlanzhang/p/10045975.html