一、架构信息

系统版本：CentOS 7.6

内核：3.10.0-1062.4.1.el7.x86_64

Kubernetes：v1.16.2

Docker-ce：19.03

推荐硬件配置：2核4G

Keepalived保证apiserever服务器的IP高可用

Haproxy实现apiserver的负载均衡

节点名称	角色	IP	安装软件
负载VIP	VIP	192.16.0..200	
k8s-110	master	192.16.0..110	kubeadm、kubelet、kubectl、docker、haproxy、keepalived
k8s-111	master	192.16.0..111	kubeadm、kubelet、kubectl、docker、haproxy、keepalived
k8s-112	master	192.16.0..112	kubeadm、kubelet、kubectl、docker
k8s-113	node	192.16.0..113	kubeadm、kubelet、kubectl、docker
k8s-114	node	192.16.0..114	kubeadm、kubelet、kubectl、docker
service网段		10.244.0.0/16	

二、部署前准备（初始化系统）

1) 关闭selinux和防火墙

```
sed -ri 's#(SELINUX=).*#\1disabled#' /etc/selinux/config
setenforce 0
systemctl disable firewalld
systemctl stop firewalld
```

2) 关闭swap

```
swapoff -a    ##临时关闭，重启失效
```

```
sed -ri 's/.*swap.*/#&/' /etc/fstab   ##永久关闭，需重启
```

3) 为每台服务器添加host解析记录

```
cat >>/etc/hosts<<EOF
192.16.0.110 k8s-110
192.16.0.111 k8s-111
192.16.0.112 k8s-112
192.16.0.113 k8s-113
192.16.0.114 k8s-114
EOF
```

4) 创建并分发密钥

在 k8s-110 创建ssh密钥。

[root@k8s-110 ~]# ssh-keygen -t rsa

分发 k8s-110 的公钥，用于免密登录其他服务器

```
for n in `seq -w 110 114`;do ssh-copy-id k8s-$n;done
```

5) 配置内核参数

```
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
```

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
```

7) 添加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

wget http://mirrors.aliyun.com/repo/Centos-7.repo -O /etc/yum.repos.d/CentOS-Base.repo
wget http://mirrors.aliyun.com/repo/epel-7.repo -O /etc/yum.repos.d/epel.repo
wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -O /etc/yum.repos.d/docker-ce.repo
```

三、部署keepalived和haproxy

1) 安装keepalived和haproxy

在k8s-110和k8s-111安装keepalived和haproxy

```
yum install -y keepalived haproxy
```

2) 修改配置

keepalived配置

k8s-110的`priority`为100，k8s-111的`priority`为90，其他配置一样。

```
[root@node-01 ~]# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   notification_email {
        yuanyouwei@163.com
   }
   notification_email_from Alexandre.Cassen@firewall.loc
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
```

```
   router_id LVS_1
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    lvs_sync_daemon_inteface ens160
    virtual_router_id 88
    advert_int 1
    priority 100
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
      192.16.0..200/24
    }
}
```

haproxy配置

k8s-110和k8s-111的haproxy配置是一样的。此处我们监听的是192.16.0.200的8443端口，因为haproxy是和k8s apiserver是部署在同一台服务器上，都用6443会冲突。

```
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 192.16.0..200:8443
        mode tcp
        balance roundrobin
        timeout server 900s
        timeout connect 15s

        server apiserver01 192.16.0.110:6443 check port 6443 inter 5000 fall 5
        server apiserver02 192.16.0.111:6443 check port 6443 inter 5000 fall 5
        server apiserver03 192.16.0.112:6443 check port 6443 inter 5000 fall 5
```

3) 启动服务

```
systemctl enable keepalived && systemctl start keepalived
systemctl enable haproxy && systemctl start haproxy
```

四、部署kubernetes

1) 安装软件

由于kubeadm对Docker的版本是有要求的，需要安装与kubeadm匹配的版本。
由于版本更新频繁，请指定对应的版本号，本文采用1.16.2版本

```
yum install -y kubelet-1.16.2 kubeadm-1.16.2 kubectl-1.16.2 ipvsadm ipset docker-ce

##启动docker

systemctl enable docker && systemctl start docker

##设置kubelet开机启动

systemctl enable kubelet
```

2) 修改初始化配置

使用kubeadm config print init-defaults > kubeadm-init.yaml 打印出默认配置，然后在根据自己的环境修改配置.**标注地方需要修改或增加**

```
[root@k8s-110 ~]# cat kubeadm-init.yaml
apiVersion: kubeadm.k8s.io/v1beta2
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s                                    ##token有效期，添加节点如果token过期需要重新生成
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.16.0.110                  ###本机真实IP
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: k8s-110
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: "192.16.0.200:8443"         ###单节点这里写真实IP+6443，多master需要写VIP地址：端口
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
```

```
  dataDir: /var/lib/etcd
imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers        ##镜像仓库地址修改为国内阿里云
kind: ClusterConfiguration
kubernetesVersion: v1.16.2                ##版本信息
networking:
  dnsDomain: cluster.local
  podSubnet: "10.244.0.0/16"        ##增加此行，为pod网段配置，若没有则flannel网络启动失败
  serviceSubnet: 10.96.0.0/12
scheduler: {}
```

3) 预下载镜像

```
[root@k8s-110 ~]# kubeadm config images pull --config kubeadm-init.yaml
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/kube-apiserver:v1.16.2
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/kube-controller-manager:v1.16.2
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler:v1.16.2
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/kube-proxy:v1.16.2
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/pause:3.1
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/etcd:3.3.15-0
[config/images] Pulled registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:1.6.2
```

4) 初始化

```
[root@k8s-110 ~]# kubeadm init --config kubeadm-init.yaml
```

kubeadm init主要执行了以下操作：

- [init]：指定版本进行初始化操作
- [preflight] ： 初始化前的检查和下载所需要的Docker镜像文件
- [kubelet-start] ： 生成kubelet的配置文件”/var/lib/kubelet/config.yaml”，没有这个文件kubelet无法启动，所以初始化之前的kubelet实际上启动失败。
- [certificates]：生成Kubernetes使用的证书，存放在/etc/kubernetes/pki目录中。
- [kubeconfig] ： 生成 KubeConfig 文件，存放在/etc/kubernetes目录中，组件之间通信需要使用对应文件。
- [control-plane]：使用/etc/kubernetes/manifest目录下的YAML文件，安装 Master 组件。
- [etcd]：使用/etc/kubernetes/manifest/etcd.yaml安装Etcd服务。
- [wait-control-plane]：等待control-plan部署的Master组件启动。
- [apiclient]：检查Master组件服务状态。
- [uploadconfig]：更新配置
- [kubelet]：使用configMap配置kubelet。
- [patchnode]：更新CNI信息到Node上，通过注释的方式记录。
- [mark-control-plane]：为当前节点打标签，打了角色Master，和不可调度标签，这样默认就不会使用Master节点来运行Pod。
- [bootstrap-token]：生成token记录下来，后边使用kubeadm join往集群中添加节点时会用到
- [addons]：安装附加组件CoreDNS和kube-proxy

5) 为kubectl准备Kubeconfig文件

kubectl默认会在执行的用户家目录下面的.kube目录下寻找config文件。这里是将在初始化时[kubeconfig]步骤生成的admin.conf拷贝到.kube/config。

```
[root@k8s-110 ~]# mkdir -p $HOME/.kube
[root@k8s-110 ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-110 ~]# chown $(id -u):$(id -g) $HOME/.kube/config
```

在该配置文件中，记录了API Server的访问地址，所以后面直接执行kubectl命令就可以正常连接到API Server中。

6) 其他master部署

USER=root

CONTROL_PLANE_IPS="k8s-111 k8s-112"

for host in ${CONTROL_PLANE_IPS}; do

ssh "${USER}"@$host "mkdir -p /etc/kubernetes/pki/etcd"

scp -r /etc/kubernetes/pki/ca.* "${USER}"@$host:/etc/kubernetes/pki/

scp -r /etc/kubernetes/pki/sa.* "${USER}"@$host:/etc/kubernetes/pki/

scp -r /etc/kubernetes/pki/front-proxy-ca.* "${USER}"@$host:/etc/kubernetes/pki/

scp -r /etc/kubernetes/pki/etcd/ca.* "${USER}"@$host:/etc/kubernetes/pki/etcd/

scp -r /etc/kubernetes/admin.conf "${USER}"@$host:/etc/kubernetes/

done

在其他master执行,注意--control-plane参数

kubeadm join 192.16.0.200:8443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:07eabe96310a5a83f8d3447a98cc713ce707466d6fbe721f15dea1b743dd79fb \

--control-plane

注意：token有效期是有限的，如果旧的token过期，可以使用`kubeadm token create --print-join-command`重新创建一条token。

7) node部署

在k8s-113、 k8s-114执行,注意没有`--control-plane`参数

kubeadm join 192.16.0.200:8443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:07eabe96310a5a83f8d3447a98cc713ce707466d6fbe721f15dea1b743dd79fb

8) 部署网络插件flannel

Master节点NotReady的原因就是因为没有使用任何的网络插件，此时Node和Master的连接还不正常。目前最流行的Kubernetes网络插件有Flannel、Calico、Canal、Weave这里选择使用flannel。

[root@k8s-110 ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

9) 查看节点状态

```
[root@k8s-110 ~]# kubectl get node
NAME      STATUS   ROLES    AGE   VERSION
k8s-110   Ready    master   21h   v1.16.2
k8s-111   Ready    master   21h   v1.16.2
k8s-112   Ready    master   21h   v1.16.2
k8s-113   Ready    <none>   21h   v1.16.2
k8s-114   Ready    <none>   21h   v1.16.2
```

查看pod状态

```
[root@k8s-110 ~]# kubectl get pod -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-67c766df46-k9wtb            1/1     Running   0          21h
coredns-67c766df46-wxb86            1/1     Running   0          21h
etcd-k8s-110                        1/1     Running   0          21h
etcd-k8s-111                        1/1     Running   0          21h
etcd-k8s-112                        1/1     Running   0          21h
kube-apiserver-k8s-110              1/1     Running   0          21h
kube-apiserver-k8s-111              1/1     Running   0          21h
kube-apiserver-k8s-112              1/1     Running   0          21h
kube-controller-manager-k8s-110     1/1     Running   1          21h
kube-controller-manager-k8s-111     1/1     Running   0          21h
kube-controller-manager-k8s-112     1/1     Running   0          21h
kube-flannel-ds-amd64-5dnrq         1/1     Running   0          21h
kube-flannel-ds-amd64-7zbqz         1/1     Running   0          21h
kube-flannel-ds-amd64-bmxfx         1/1     Running   0          21h
kube-flannel-ds-amd64-l9zsg         1/1     Running   1          21h
kube-flannel-ds-amd64-wtwrc         1/1     Running   0          21h
kube-proxy-8bsfp                    1/1     Running   0          21h
kube-proxy-8lwb7                    1/1     Running   0          21h
kube-proxy-ckxkl                    1/1     Running   0          21h
kube-proxy-l4h8q                    1/1     Running   0          21h
kube-proxy-xrjf6                    1/1     Running   0          21h
kube-scheduler-k8s-110              1/1     Running   1          21h
kube-scheduler-k8s-111              1/1     Running   0          21h
kube-scheduler-k8s-112              1/1     Running   0          21h
```

K8S部署完成！！！