链滴

手把手教你用 kubeadm 安装 k8s v1.20

作者：chenteng

kubeadmin安装k8s v1.20

一、环境准备

PS:本次实验采用单mster节点，两个worker节点组成kubernetes集群，版本是v1.20.9。由于是笔记自建虚拟机，所以配置相对低一些，有条件的同学可以适当调高配置

操作系统	角色	硬盘	CPU & MEM
Centos 7.9	master	40GB	核8G
Centos 7.9	node01	40GB	核4G
Centos 7.9	node02	40GB	核4G

二、安装容器运行时---Docker

2.1、添加阿里镜像源

```
sudo yum install -y yum-utils
sudo yum-config-manager \
--add-repo \
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

2.2、安装docker

```
yum install -y docker-ce-20.10.7 docker-ce-cli-20.10.7  containerd.io-1.4.6
```

2.3、启动

```
systemctl enable docker --now
```

2.4、配置加速

原文链接：手把手教你用 kubeadm 安装 k8s v1.20

这里额外添加了docker的生产环境核心配置cgroup

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://82m9ar63.mirror.aliyuncs.com"],
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

三、安装kubernetes

3.1、基础环境

所有机器执行以下操作

```
#各个机器设置自己的域名
hostnamectl set-hostname xxxx


# 将 SELinux 设置为 permissive 模式（相当于将其禁用）
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

#关闭swap
swapoff -a
sed -ri 's/.*swap.*/#&/' /etc/fstab

#允许 iptables 检查桥接流量
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

3.2、安装kubelet、kubeadm、kubectl

```
cat <<EOF | sudo tee /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
exclude=kubelet kubeadm kubectl
EOF


sudo yum install -y kubelet-1.20.9 kubeadm-1.20.9 kubectl-1.20.9 --disableexcludes=kuberne
es

sudo systemctl enable --now kubelet
```

kubelet 现在每隔几秒就会重启，因为它陷入了一个等待 kubeadm 指令的死循环

3.3、下载镜像

```
sudo tee ./images.sh <<-'EOF'
#!/bin/bash
images=(
kube-apiserver:v1.20.9
kube-proxy:v1.20.9
kube-controller-manager:v1.20.9
kube-scheduler:v1.20.9
coredns:1.7.0
etcd:3.4.13-0
pause:3.2
)
for imageName in ${images[@]} ; do
docker pull registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images/$imageName
done
EOF

chmod +x ./images.sh && ./images.sh
```

3.4、配置hosts解析

```
echo -e "192.168.1.10 master \n192.168.1.11 node01\n192.168.1.12 node02" >> /etc/hosts
```

3.5、master节点初始化集群

<font color=red>**注意：**

<font color=red>1. 所有网络范围不重叠

2. advertise-address等于master的IP

3. plane-endpoint等于填写在hosts文件中的master

</font>

```
kubeadm init \
--apiserver-advertise-address=192.168.1.10 \
--control-plane-endpoint=master \
--image-repository registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images \
--kubernetes-version v1.20.9 \
--service-cidr=10.96.0.0/16 \
--pod-network-cidr=10.168.0.0/16
```

初始化成功

提示信息保存一下，以后加入主节点或worker节点使用

```
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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 control-plane nodes by copying certificate authorities
and service account keys on each node and then running the following as root:

  kubeadm join master:6443 --token 4qw94d.i5fa5tgtztd5o5nu \
    --discovery-token-ca-cert-hash sha256:ec7c8b230762c533c055493a4e4c210e7a5587a8b15
cf38da7d5e2bbeb621c6 \
    --control-plane

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

kubeadm join master:6443 --token 4qw94d.i5fa5tgtztd5o5nu \
    --discovery-token-ca-cert-hash sha256:ec7c8b230762c533c055493a4e4c210e7a5587a8b15
cf38da7d5e2bbeb621c6
```

3.6、执行初始化命令

仅在master节点执行如下命令

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

3.7、加入剩余两个worker节点

两个worker节点分别执行如下命令(在上面有)

```
kubeadm join master:6443 --token 4qw94d.i5fa5tgtztd5o5nu \
>     --discovery-token-ca-cert-hash
sha256:ec7c8b230762c533c055493a4e4c210e7a5587a8b153cf38da7d5e2bbeb621c6
[preflight] Running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions
 20.10.7. Latest validated version: 19.03
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-
onfig -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-fl
gs.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

3.8、验证

可以看到我们的集群已经搭建完成，但是节点都是Notready，是因为k8s各节点通信通过第三方的网插件，所以接下来我们需要安装第三方网络插件

```
[root@master ~]# kubectl get node
NAME     STATUS     ROLES                  AGE     VERSION
master   NotReady   control-plane,master   7m51s   v1.20.9
node01   NotReady   <none>                 2m3s    v1.20.9
node02   NotReady   <none>                 2m      v1.20.9
```

四、安装calico网络组件

4.1、下载calico的配置文件

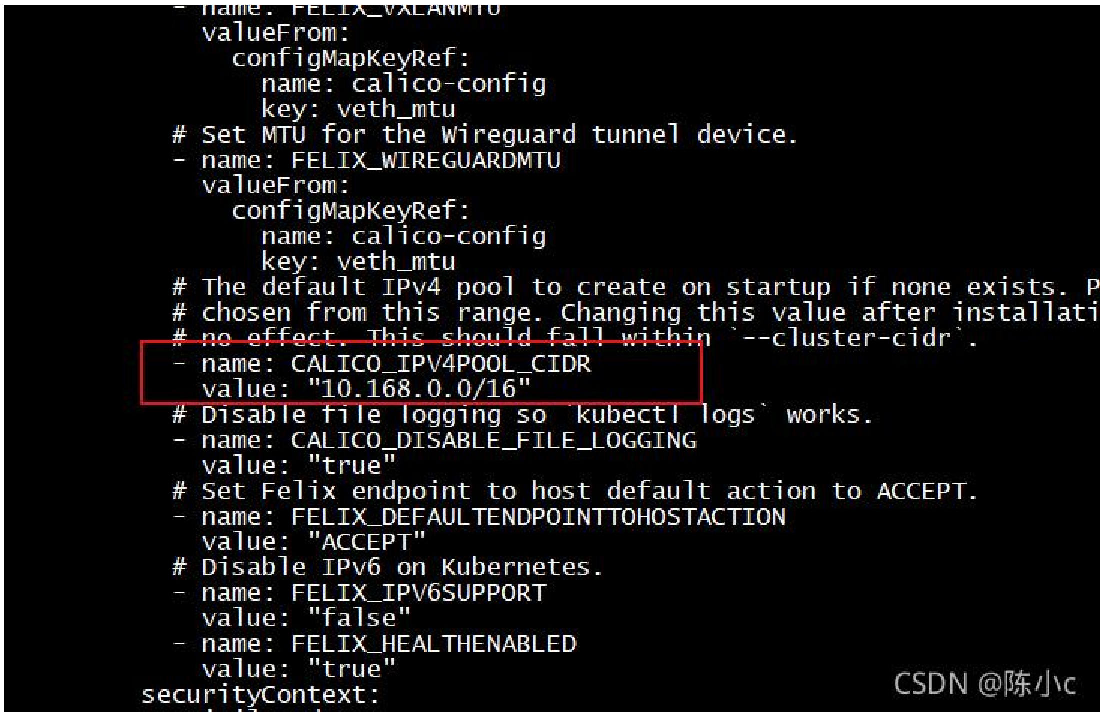
```
yum isntall -y wget &&
wget https://docs.projectcalico.org/manifests/calico.yaml --no-check-certificate
```

4.2、修改配置文件

注意：

calico默认的CALICO_IPV4POOL_CIDR是192.168.0.0/16，在上面3.5、master节点初始化集群骤中,为避免网段重复。所以更改了CIDR的值，所以在calico的配置文件中，要<font color=red>改成相同的值，并取消注释<font>

如下图：



```
  - name: FELIX_VXLANMTU
    valueFrom:
      configMapKeyRef:
        name: calico-config
        key: veth_mtu
  # Set MTU for the Wireguard tunnel device.
  - name: FELIX_WIREGUARDMTU
    valueFrom:
      configMapKeyRef:
        name: calico-config
        key: veth_mtu
  # The default IPv4 pool to create on startup if none exists. P
  # chosen from this range. Changing this value after installati
  # no effect. This should fall within `--cluster-cidr`.
  - name: CALICO_IPV4POOL_CIDR
    value: "10.168.0.0/16"
  # Disable file logging so `kubectl logs` works.
  - name: CALICO_DISABLE_FILE_LOGGING
    value: "true"
  # Set Felix endpoint to host default action to ACCEPT.
  - name: FELIX_DEFAULTENDPOINTTOHOSTACTION
    value: "ACCEPT"
  # Disable IPv6 on Kubernetes.
  - name: FELIX_IPV6SUPPORT
    value: "false"
  - name: FELIX_HEALTHENABLED
    value: "true"
securityContext:
```

4.3、通过命令安装calico网络插件

```
kubectl create -f calico.yaml
##等待4-5分钟
kubectl get pod -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-659bd7879c-jndd7   1/1     Running   0          8m20s
calico-node-f7jbc                          1/1     Running   0          8m20s
calico-node-x64f6                          1/1     Running   0          8m20s
calico-node-zcxl2                          1/1     Running   0          8m20s
coredns-5897cd56c4-js7h4                   1/1     Running   0          37m
coredns-5897cd56c4-t46w8                   1/1     Running   0          37m
etcd-master                                1/1     Running   0          37m
kube-apiserver-master                      1/1     Running   0          37m
kube-controller-manager-master             1/1     Running   0          37m
kube-proxy-4tmqd                           1/1     Running   0          31m
kube-proxy-d5sxh                           1/1     Running   0          37m
kube-proxy-hh6fv                           1/1     Running   0          31m
kube-scheduler-master                      1/1     Running   0          37m
```

当我们看到都是Running的时候，就代表我们calico网络插件安装成功了，

此时我们去看一下node的状态,发现已经全部ready了，至此kubernetes安装成功

```
[root@master ~]# kubectl get node
NAME     STATUS   ROLES                  AGE   VERSION
master   Ready    control-plane,master   39m   v1.20.9
node01   Ready    <none>                 33m   v1.20.9
node02   Ready    <none>                 33m   v1.20.9
```