kubeadm部署kubernetes单主集群

一、前言

> Docker中文文档网站：https://yeasy.gitbooks.io/docker_practice/
>
> Docker官方文档网站：https://docs.docker.com/

1.1.系统环境

Linux：Centos_7_5_64 (内核3.10+)

1.2.关闭防火墙

> 防火墙一定要关闭，否则在后续安装K8S集群的时候是个trouble maker。执行下面语句关闭，并禁用开机启动：

```
systemctl stop firewalld & systemctl disable firewalld
```

1.3.关闭SeLinux

```
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=disabled/' /etc/selinux/config
```

1.4.关闭Swap

> 在安装K8S集群时，Linux的Swap内存交换机制是一定要关闭的，否则会因为内存交换而影响性能以及稳定性。这里，我们可以提前进行设置：

- 执行**swapoff -a**可临时关闭，但系统重启后恢复
- 编辑**/etc/fstab**，注释掉包含**swap**的那一行即可，重启后可永久关闭，如下所示：

```
sed -i '/ swap / s/^/#/' /etc/fstab
```

1.5.设置主机名

```
#主节点
hostnamectl --static set-hostname  k8s-master
#从节点1
hostnamectl --static set-hostname  k8s-node1
#从节点2
hostnamectl --static set-hostname  k8s-node2
```

1.6.修改hosts

```
192.168.3.226 k8s-master
192.168.3.225 k8s-node1
192.168.3.228 k8s-node2
```

1.7.配置路由参数

> 防止kubeadm报路由警告，CentOS 7可能会出现iptables被绕过而导致流量被错误路由的问题。确保net.bridge.bridge-nf-call-iptables在sysctl配置中设置为1。

- 将内容写入k8s.conf文件

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

- 立即生效

```
sysctl -p /etc/sysctl.d/k8s.conf
```

二、安装Docker

> 详细的安装信息，参见官网：https://docs.docker.com/install/linux/docker-ce/centos/#prerequisites

2.1.使用存储库安装

> 在新主机上首次安装Docker CE之前，需要设置Docker存储库。之后，就可以从存储库安装和更新Docker了。

2.1.1设置存储库

```
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

2.1.2.安装Docker CE

1. 安装最新版

```
sudo yum install docker-ce
```

2. 安装指定版本

```
yum list docker-ce --showduplicates | sort -r
sudo yum install docker-ce-<VERSION STRING>
```

3. 启动Docker，并设置开机启动

```
sudo systemctl start docker & systemctl enable docker
```

4. 验证是否成功安装，下载一个测试映像并在容器中运行它。当容器运行时，它打印一条信息消息并退出。

```
sudo docker run hello-world
```

5. 查看docker版本

```
docker --version
```

2.1.3.卸载Docker CE

1. 卸载Docker

```
sudo yum remove docker-ce
```

2. 删除自定义配置文件

```
sudo rm -rf /var/lib/docker
```

三、部署Docker私有仓库

> Docker提供了开放的中央仓库dockerhub，同时也允许使用`registry`搭建本地私有仓库。`registry`操作比较繁琐，并且没有管理页面，使用起来不便捷，达不到企业级的要求。有一些开源的私有仓库，可以满足企业及要求，`Harbor`是其中比较好的一款。下面分别来搭建下两种仓库。

搭建私有仓库有如下的优点：

1. 节省网络带宽，提升Docker部署速度，不用每个镜像从DockerHub上去下载，只需从私有仓库下载就可；
2. 私有镜像，包含公司敏感信息，不方便公开对外，只在公司内部使用。

3.1.搭建私有仓库

> 详细信息，参见官网：https://docs.docker.com/registry/configuration/

3.1.1.获取registry

```
docker pull registry
```

3.1.2.运行registry

```
docker run -d -p 5000:5000 --restart=always --name=registry -v
/var/dockerRegistry:/var/lib/registry registry
```

解释：

- -d：后台运行。
- -p：将容器的5000端口映射到宿主机的5000端口。
- --restart：docker服务重启后总是重启此容器。
- --name：容器的名称。
- -v：将容器内的/var/lib/registry映射到宿主机的/var/dockerRegistry目录。

3.1.3.上传镜像

- 修改镜像Tag

```
docker tag k8s.gcr.io/coredns:1.2.6  192.168.3.34:5000/coredns:1.2.6
```

- 上传镜像

```
docker push 192.168.3.34:5000/coredns:1.2.6
```

- 上传镜像会报如下错误，解决方案参见下面的`3.1.4`

```
The push refers to repository [192.168.3.34:5000/coredns]
Get https://192.168.3.34:5000/v2/: http: server gave HTTP response to HTTPS client
```

3.1.4.加入注册表

> Docker在1.3.x之后默认docker registry使用的是https，会导致私有仓库报错。

有两种方式解决这个问题：

- 搭建HTTPS证书（推荐），该方法操作复杂，本文档的环境不具备条件。
- 加入注册表，修改本地主机的docker启动配置文件，在/etc/docker/路径下，添加daemon.json文件。

```
{
  "insecure-registries": ["192.168.3.34:5000"]
}
```

- 重启docker

```
systemctl restart docker
```

3.1.5.查看私有仓库

- 查看所有镜像

```
curl -XGET http://192.168.3.34:5000/v2/_catalog
```

- 获取某个镜像的标签列表

```
curl -XGET http://192.168.3.34:5000/v2/镜像名称/tags/list
```

3.2.企业级私有仓库

> Harbor是由VMware公司开源的企业级的Docker Registry管理项目，它包括权限管理(RBAC)、LDAP、日志审核、管理界面、自我注册、镜像复制和中文支持等功能。Harbo依赖于docker，及docker-compose。

3.2.1.安装docker-compose

> docker-compose网址：https://docs.docker.com/compose/install/

- 安装

```
curl -L https://github.com/docker/compose/releases/download/1.24.0-rc1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

- 查看版本

```
docker-compose --version
```

3.2.2.安装Harbor

> Harbor网址：https://github.com/goharbor/harbor/releases

- 安装，下载安装包，上传到服务器，解压：

```
tar -vxf  harbor-offline-installer-v1.7.1.tgz
```

- 修改配置，打开解压目录下的`harbor.cfg`文件，修改如下属性，其他的属性根据需要修改。

```
# hostname设置访问地址，可以使用ip、域名，不可以设置为127.0.0.1或localhost
hostname = 192.168.3.34
```

注意：

1、默认的端口：80，默认协议：HTTP

2、如果已经安装了上一步的`register`，需要先删除容器

3、如果使用`HTTP`协议，同样需要将IP加入注册表

- 启动，运行harbor目录下的`install.sh`

```
./install.sh
```

- 登录，直接输入ip即可登录。http://192.168.3.34:80

默认的用户名和密码：admin / Harbor12345

3.2.3.上传镜像

首先，需要登录到Harbor仓库，其他操作步骤相同：

```
docker login 192.168.3.34:80
```

四、安装Kubernetes

> 详细的安装信息，参见官网：https://kubernetes.io/docs/setup/independent/install-kubeadm/

4.1.使用存储库安装

4.1.1.设置存储库

```
# 切换到存储库路径
cd /etc/yum.repos.d/
# 添加存储库
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kube*
EOF
```

4.1.2.安装kubernetes

1. 安装`kubelet`、 `kubeadm`、`kubectl`

```
yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

2. 启动`kubelet`，并设置开机自启动

```
systemctl enable kubelet && systemctl start kubelet
```

3. 配置主节点上的`kubelet`使用`cgroup`驱动程序

```
# 查看docker的cgroup驱动
docker info | grep -i cgroup
# 输出结果
Cgroup Driver: cgroupfs
```

- 确保kubelet 的cgroup drive 和docker的cgroup drive一样:

```
sed -i "s/cgroup-driver=systemd/cgroup-driver=cgroupfs/g"
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

- 重新启动kubelet:

```
systemctl daemon-reload
systemctl restart kubelet
```

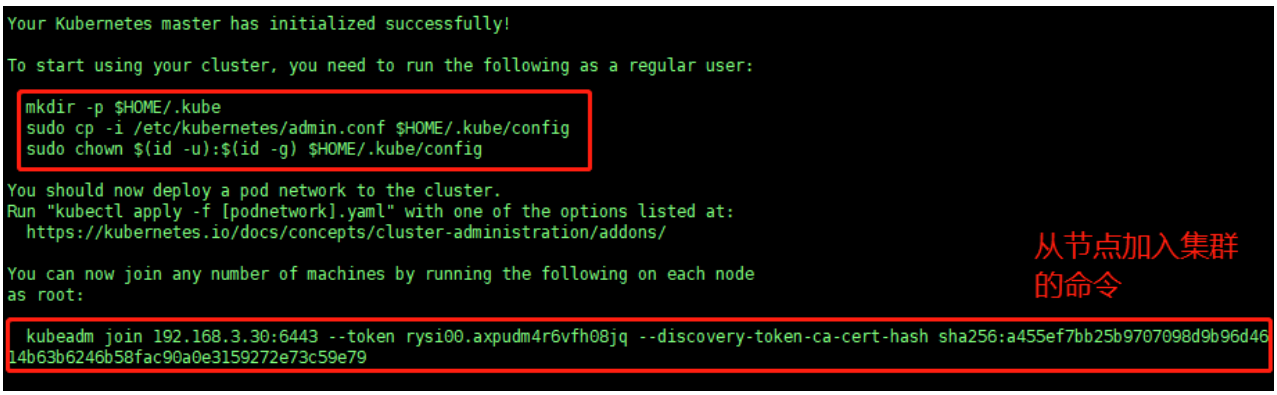
4. 初始化master

```
kubeadm init --pod-network-cidr=10.244.0.0/16 --kubernetes-version=v1.13.0 --apiserver-advertise-address=192.168.3.30
```

含义：

- **--pod-network-cidr**：表示集群将使用的子网范围。
- **--kubernetes-version**：表示K8S版本，这里必须与导入到Docker镜像版本一致，否则会访问谷歌去重新下载K8S最新版的Docker镜像。
- **--apiserver-advertise-address**：表示绑定的主节点的IP。
- 若执行**kubeadm init**出错或强制终止，则再需要执行该命令时，需要先执行**kubeadm reset**重置。

  注意，记录下如下信息



  详细说明，参见官网：https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/

5. 要使kubectl为非root用户工作，请运行以下命令

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

- 如果是**root**用户，则可以运行：

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

6. 安装pod网络附加组件

```
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml

kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml
```

下载calico.yaml文件，修改其中的配置，

```
# Auto-detect the BGP IP address.
- name: IP
  value: "autodetect"
# 添加如下的配置，设置使用的网卡
- name: IP_AUTODETECTION_METHOD
  value: "interface=ens*"
```

- 查看pod状态：

```
kubectl get pods --all-namespaces
```

7. 将Master作为工作节点（可选）

> K8S集群默认不会将Pod调度到Master上，这样Master的资源就浪费了。在Master上，可以运行以下命令使其作为一个工作节点：

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

8. 将其他节点加入集群

- 在其他两个节点上，执行主节点生成的`kubeadm join`命令即可加入集群：

```
kubeadm join 192.168.3.30:6443 --token rysi00.axpudm4r6vfh08jq --discovery-token-ca-cert-hash sha256:a455ef7bb25b9707098d9b96d4614b63b6246b58fac90a0e3159272e73c59e79
```

- 验证集群是否正常，当所有节点加入集群后，在主节点上运行如下命令，即可看到集群情况

```
kubectl get nodes
```

- 查看所有pod状态，status全部为Running则表示集群正常。

```
kubectl get pods -n kube-system
```

9.修改apiserver的端口范围（可选）

> 编辑/etc/kubernetes/manifests下的kube-apiserver.yaml文件，在command参数下添加如下信息

```
- --service-node-port-range=1-65535
```

五、安装K8S Dashboard

> 详细信息，参见官网：https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

5.1.安装

- 默认情况下不部署仪表板UI。要部署它，首先从官网获取**kubernetes-dashboard.yaml**，在末尾添加如下配置（主要是设置端口类型为 NodePort）：

```yaml
# ------------------- Dashboard Service ------------------- #

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
spec:
  type: NodePort
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 30001
  selector:
    k8s-app: kubernetes-dashboard
```

- 运行以下命令：

```
kubectl create -f kubernetes-dashboard.yaml
```

5.2.查看节点端口

```
kubectl get service -n kube-system -o wide
```

- 记录下端口号，打开页面时需要用到。

```
[root@k8s-master ~]# kubectl get service -n kube-system -o wide
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE     SELECTOR
calico-typha           ClusterIP   10.111.171.61    <none>        5473/TCP        9d      k8s-app=calico-typha
kube-dns               ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP   9d      k8s-app=kube-dns
kubernetes-dashboard   NodePort    10.103.148.129   <none>        443:30001/TCP   6d17h   k8s-app=kubernetes-dashboard
```

5.3.创建用户

- 创建dashboard-rbac.yaml文件，内容如下：

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin
  namespace: kube-system
```

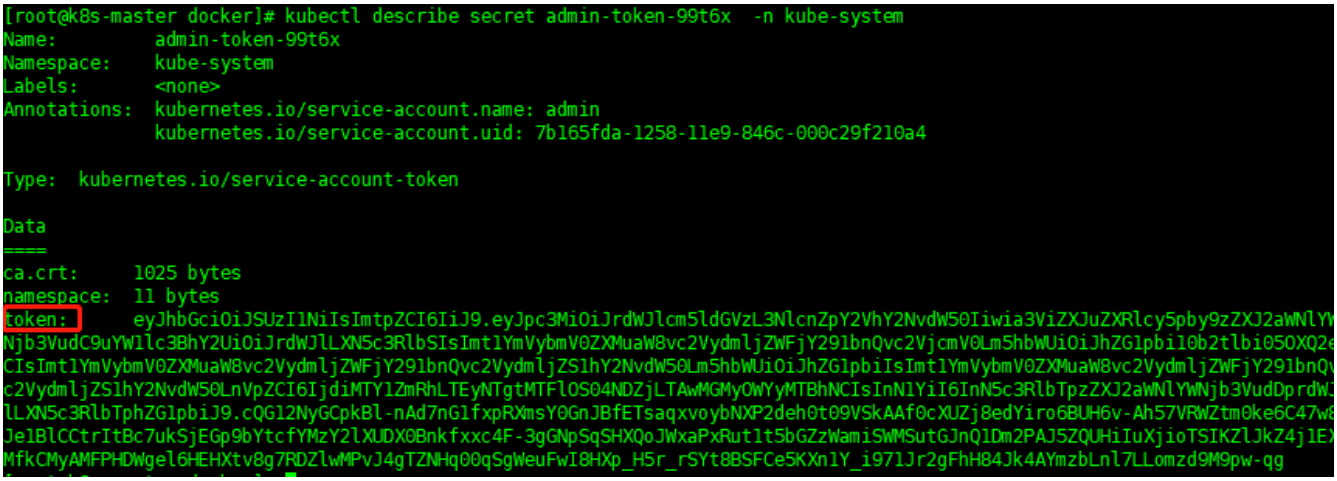
- 运行命令：

```
kubectl create -f dashboard-rbac.yaml
```

5.4.获取登录token

- 获取tokens

```
kubectl describe secret admin  -n kube-system
```



5.5.登录页面

- 打开连接（**火狐**）： https://192.168.3.30:30001
- 选择**令牌**登录方式
- 输入上图中的token，点击登录