《快速部署一套 K8s 集群》

v1.25，kubeadm 方式

1、前置知识点

1.1 生产环境可部署 Kubernetes 集群的两种方式

目前生产部署 Kubernetes 集群主要有两种方式：

- **kubeadm**

Kubeadm 是一个 K8s 部署工具，提供 kubeadm init 和 kubeadm join，用于快速部署 Kubernetes 集群。

- **二进制包**

从 github 下载发行版的二进制包，手动部署每个组件，组成 Kubernetes 集群。

这里采用 kubeadm 搭建集群。

kubeadm 工具功能：

- **kubeadm init**：初始化一个 Master 节点
- **kubeadm join**：将工作节点加入集群
- **kubeadm upgrade**：升级 K8s 版本

- **kubeadm token**：管理 kubeadm join 使用的令牌
- **kubeadm reset**：清空 kubeadm init 或者 kubeadm join 对主机所做的任何更改
- **kubeadm version**：打印 kubeadm 版本
- **kubeadm alpha**：预览可用的新功能

1.2 准备环境

服务器要求：

- 建议最小硬件配置：2 核 CPU、2G 内存、20G 硬盘
- 服务器最好可以访问外网，会有从网上拉取镜像需求，如果服务器不能上网，需要提前下载对应镜像并导入节点

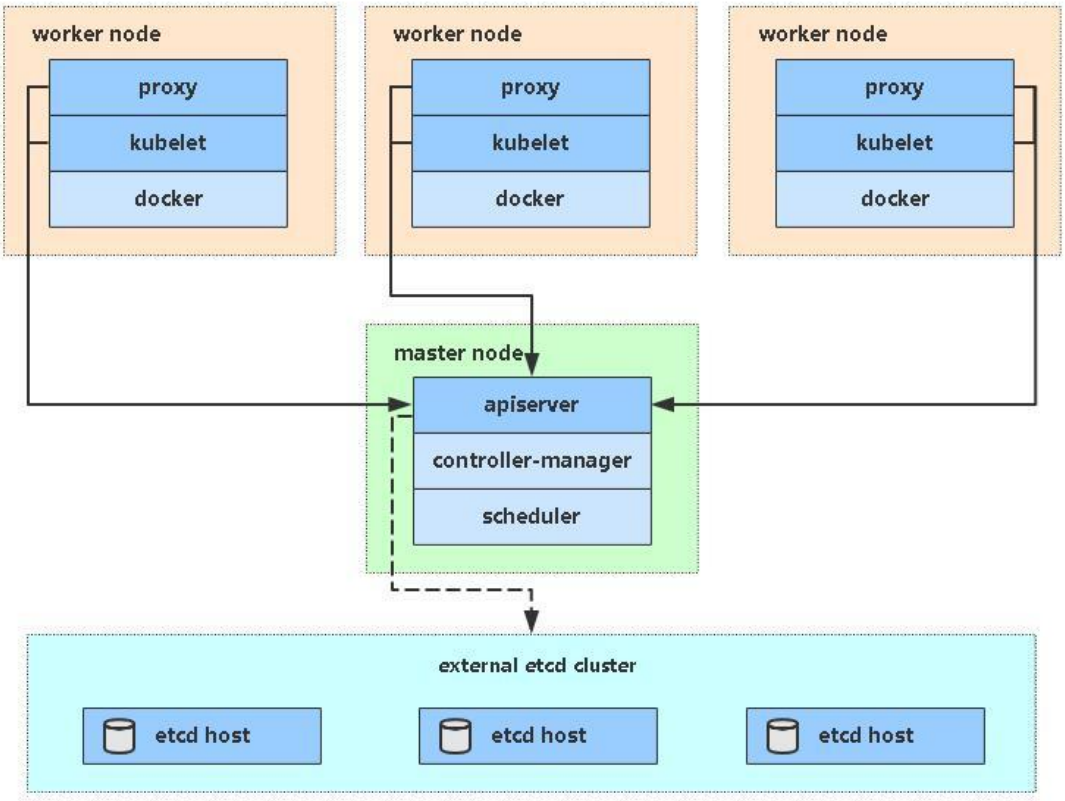
软件环境：

软件	版本
操作系统	CentOS7.9_x64 （mini）
Docker	20-ce
Kubernetes	1.25

服务器规划：

角色	IP
k8s-master	192.168.1.71
k8s-node1	192.168.1.72
k8s-node2	192.168.1.73

架构图：



1.3 操作系统初始化配置【所有节点】

```
# 关闭防火墙
systemctl stop firewalld
systemctl disable firewalld

# 关闭 selinux
sed -i 's/enforcing/disabled/' /etc/selinux/config  # 永久
setenforce 0  # 临时

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

# 根据规划设置主机名
hostnamectl set-hostname <hostname>

# 在 master 添加 hosts
```

```
cat >> /etc/hosts << EOF
192.168.1.71 k8s-master
192.168.1.72 k8s-node1
192.168.1.73 k8s-node2
EOF

# 将桥接的 IPv4 流量传递到 iptables 的链
cat > /etc/sysctl.d/k8s.conf << EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system  # 生效

# 时间同步
yum install ntpdate -y
ntpdate time.windows.com
```

2. 安装 Docker/kubeadm/kubelet【所有节点】

2.1 安装 Docker

```
wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -O
/etc/yum.repos.d/docker-ce.repo
yum -y install docker-ce
systemctl enable docker && systemctl start docker
```

配置镜像下载加速器：

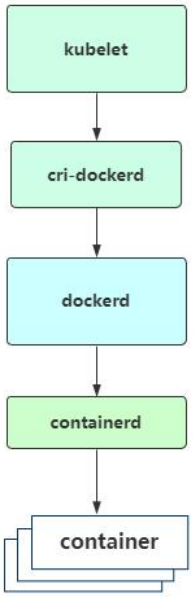
```
cat > /etc/docker/daemon.json << EOF
  {
    "registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"],
    "exec-opts": ["native.cgroupdriver=systemd"]
  }
EOF

systemctl restart docker
docker info
```

2.2 安装 cri-dockerd

Kubernetes v1.24 移除 docker-shim 的支持，而 Docker Engine 默认又不支持 CRI 标准，因此二者默认无法再直接集成。为此，Mirantis 和 Docker 联合创建了 cri-dockerd 项目，用于为

Docker Engine 提供一个能够支持到 CRI 规范的桥梁，从而能够让 Docker 作为 Kubernetes 容器引擎。

如图所示：




```
wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.2.5/cri-dockerd-0.2.5-3.el7.x86_64.rpm
rpm -ivh cri-dockerd-0.2.5-3.el7.x86_64.rpm
```

指定依赖镜像地址：

```
vi /usr/lib/systemd/system/cri-docker.service
ExecStart=/usr/bin/cri-dockerd --container-runtime-endpoint fd:// --pod-infra-container-image=registry.aliyuncs.com/google_containers/pause:3.7

systemctl daemon-reload
systemctl enable cri-docker && systemctl start cri-docker
```

2.3 添加阿里云 YUM 软件源

```
cat > /etc/yum.repos.d/kubernetes.repo << EOF
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
```

```
repo_gpgcheck=0
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
```

2.4 安装 kubeadm，kubelet 和 kubectl

由于版本更新频繁，这里指定版本号部署：

```
yum install -y kubelet-1.25.0 kubeadm-1.25.0 kubectl-1.25.0
systemctl enable kubelet
```

3. 部署 Kubernetes Master

在 192.168.1.71（Master）执行。

```
kubeadm init \
  --apiserver-advertise-address=192.168.1.71 \
  --image-repository registry.aliyuncs.com/google_containers \
  --kubernetes-version v1.25.0 \
  --service-cidr=10.96.0.0/12 \
  --pod-network-cidr=10.244.0.0/16 \
  --cri-socket=unix:///var/run/cri-dockerd.sock \
  --ignore-preflight-errors=all
```

- --apiserver-advertise-address 集群通告地址
- --image-repository 由于默认拉取镜像地址 k8s.gcr.io 国内无法访问，这里指定阿里云镜像仓库地址
- --kubernetes-version K8s 版本，与上面安装的一致
- --service-cidr 集群内部虚拟网络，Pod 统一访问入口
- --pod-network-cidr Pod 网络，与下面部署的 CNI 网络组件 yaml 中保持一致
- --cri-socket 指定 cri-dockerd 接口，如果是 containerd 则使用--cri-socket unix:///run/containerd/containerd.sock

初始化完成后，最后会输出一个 join 命令，先记住，下面用。

拷贝 kubectl 使用的连接 k8s 认证文件到默认路径：

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

查看工作节点：

```
kubectl get nodes
NAME               STATUS     ROLES            AGE   VERSION
k8s-master   NotReady   control-plane        20s   v1.25.0
```

注：由于网络插件还没有部署，还没有准备就绪 NotReady，先继续

参考资料：

https://kubernetes.io/zh/docs/reference/setup-tools/kubeadm/kubeadm-init/#config-file

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#initializing-your-control-plane-node

4. 加入 Kubernetes Node

在 192.168.1.72/73（Node）执行。

向集群添加新节点，执行在 kubeadm init 输出的 kubeadm join 命令并手动加上--cri-socket=unix:///var/run/cri-dockerd.sock：

```
kubeadm join 192.168.1.71:6443 --token 7gqt13.kncw9hg5085iwclx \
--discovery-token-ca-cert-hash
sha256:66fbfcf18649a5841474c2dc4b9ff90c02fc05de0798ed690e1754437be35a01 --cri-socket=unix:///var/run/cri-dockerd.sock
```

默认 token 有效期为 24 小时，当过期之后，该 token 就不可用了。这时就需要重新创建 token，可以直接使用命令快捷生成：

```
kubeadm token create --print-join-command
```

参考资料：https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm-join/

5. 部署容器网络（CNI）

Calico 是一个纯三层的数据中心网络方案，是目前 Kubernetes 主流的网络方案。

下载 YAML：

```
wget https://docs.projectcalico.org/manifests/calico.yaml
```

下载完后还需要修改里面定义 Pod 网络（CALICO_IPV4POOL_CIDR），与前面 kubeadm init 的 --pod-network-cidr 指定的一样。

修改完后文件后，部署：

```
kubectl apply -f calico.yaml
kubectl get pods -n kube-system
```

等 Calico Pod 都 Running，节点也会准备就绪。

注：以后所有 yaml 文件都只在 Master 节点执行。

安装目录：/etc/kubernetes/

组件配置文件目录：/etc/kubernetes/manifests/

参考资料：https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#pod-network

6. 部署 Dashboard

Dashboard 是官方提供的一个 UI，可用于基本管理 K8s 资源。

YAML 下载地址：

```
https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/recommended.yaml
```

课件中文件名是：kubernetes-dashboard.yaml

默认 Dashboard 只能集群内部访问，修改 Service 为 NodePort 类型，暴露到外部：

```
vi recommended.yaml
...
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
```

```
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 30001
  selector:
    k8s-app: kubernetes-dashboard
  type: NodePort
...

kubectl apply -f recommended.yaml
kubectl get pods -n kubernetes-dashboard
```

访问地址：https://NodeIP:30001

创建 service account 并绑定默认 cluster-admin 管理员集群角色：

```
# 创建用户
kubectl create serviceaccount dashboard-admin -n kubernetes-dashboard
# 用户授权
kubectl create clusterrolebinding dashboard-admin --clusterrole=cluster-admin --
serviceaccount=kubernetes-dashboard:dashboard-admin
# 获取用户 Token
kubectl create token dashboard-admin -n kubernetes-dashboard
```

使用输出的 token 登录 Dashboard。

7. 切换容器引擎为 Containerd

containerd 是一个主流的容器引擎，与 Docker 相兼容，相比 Docker 轻量很多，目前较为成熟。

参考资料：https://kubernetes.io/zh/docs/setup/production-environment/container-runtimes/#containerd

1、配置先决条件

如果是由 docker 切 containerd 这步可省略。

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
```

```
sudo modprobe overlay
sudo modprobe br_netfilter

# 设置必需的 sysctl 参数，这些参数在重新启动后仍然存在。
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

sudo sysctl --system
```

2、安装 containerd

```
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
yum install -y containerd.io
mkdir -p /etc/containerd
containerd config default > /etc/containerd/config.toml
```

3、修改配置文件

- pause 镜像设置过阿里云镜像仓库地址
- 拉取 Docker Hub 镜像配置加速地址设置为阿里云镜像仓库地址

```
vi /etc/containerd/config.toml
...
   [plugins."io.containerd.grpc.v1.cri"]
      sandbox_image = "registry.aliyuncs.com/google_containers/pause:3.2"
       ...
        [plugins."io.containerd.grpc.v1.cri".registry.mirrors]
          [plugins."io.containerd.grpc.v1.cri".registry.mirrors."docker.io"]
            endpoint = ["https://b9pmyelo.mirror.aliyuncs.com"]

systemctl restart containerd
```

4、配置 kubelet 使用 containerd

```
vi /var/lib/kubelet/kubeadm-flags.env
KUBELET_KUBEADM_ARGS="--container-runtime=remote --container-runtime-
endpoint=unix:///run/containerd/containerd.sock --pod-infra-container-
image=registry.aliyuncs.com/google_containers/pause:3.8"

systemctl restart kubelet
```

5、验证

```
kubectl get node -o wide

k8s-node1   xxx   containerd://1.5.6
```

6、管理容器工具

containerd 提供了 ctr 命令行工具管理容器，但功能比较简单，所以一般会用 crictl 工具检查和调试容器。

项目地址：https://github.com/kubernetes-sigs/cri-tools/

设置 crictl 连接 containerd：

```
vi /etc/crictl.yaml
runtime-endpoint: unix:///run/containerd/containerd.sock
image-endpoint: unix:///run/containerd/containerd.sock
timeout: 10
debug: false
```

下面是 docker 与 crictl 命令对照表：

镜像相关功能	Docker	Containerd
显示本地镜像列表	docker images	crictl images
下载镜像	docker pull	crictl pull
上传镜像	docker push	无，例如 buildk
删除本地镜像	docker rmi	crictl rmi
查看镜像详情	docker inspect IMAGE-ID	crictl inspecti IMAGE-ID

容器相关功能	Docker	Containerd
显示容器列表	docker ps	crictl ps

创建容器	docker create	crictl create
启动容器	docker start	crictl start
停止容器	docker stop	crictl stop
删除容器	docker rm	crictl rm
查看容器详情	docker inspect	crictl inspect
附加容器	docker attach	crictl attach
执行命令	docker exec	crictl exec
查看日志	docker logs	crictl logs
查看容器资源	docker stats	crictl stats

POD 相关功能	Docker	Containerd
显示 POD 列表	无	crictl pods
查看 POD 详情	无	crictl inspectp
运行 POD	无	crictl runp
停止 POD	无	crictl stopp