从零开始搭建 kubernetes 集群

作者：arrayMi
原文链接：https://ld246.com/article/1584972682183
来源网站：链滴

前言

Kubernetes 是 Google 开源的一个容器编排引擎，它支持自动化部署、大规模可伸缩、应用器化管理。在生产环境中部署一个应用程序时，通常要部署该应用的多个实例以便对应用请求进行负均衡。

在 Kubernetes 中，我们可以创建多个容器，每个容器里面运行一个应用实例，然后通过内置负载均衡策略，实现对这一组应用实例的管理、发现、访问，而这些细节都不需要运维人员去进行复的手工配置和处理。也不用开发人员针对不同环境去做服务的适配了，大大提高了工作效率。

环境搭建

目标：使用 kubeadm 搭建一个 3 台机器组成的 k8s 集群，1 台 master 节点，2 台 worker 节点

配置要求：

- One or more machines running one of:
  - Ubuntu 16.04+
  - Debian 9+
  - CentOS 7【课程中使用】
  - Red Hat Enterprise Linux (RHEL) 7
  - Fedora 25+
  - HypriotOS v1.0.1+
  - Container Linux (tested with 1800.6.0)
- 2 GB or more of RAM per machine (any less will leave little room for your apps)
- 2 CPUs or more

- Full network connectivity between all machines in the cluster (public or private network is fi e)
- Unique hostname, Mac address, and product_uuid for every node. See here for more details.
- Certain ports are open on your machines. See here for more details.
- Swap disabled. You **MUST** disable swap in order for the kubelet to work properly.

1.1 版本统一

```
Docker      18.09.0
---
kubeadm-1.14.0-0
kubelet-1.14.0-0
kubectl-1.14.0-0
---
k8s.gcr.io/kube-apiserver:v1.14.0
k8s.gcr.io/kube-controller-manager:v1.14.0
k8s.gcr.io/kube-scheduler:v1.14.0
k8s.gcr.io/kube-proxy:v1.14.0
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.3.10
k8s.gcr.io/coredns:1.3.1
---
calico:v3.9.5
```

1.2 准备 3 台 CentOS

os: centos8

master: 4U8G 192.168.2.106

worker1: 2U4G 192.168.2.240

worker2: 2U4G 192.168.2.149

1.3 更新并安装依赖

3 台机器都需要执行

```
yum -y update
yum install -y conntrack ipvsadm ipset jq sysstat curl iptables libseccomp
```

1.4 安装 Docker

在每一台机器上都安装好 Docker，版本为 18.09.0

```
01 安装必要的依赖
    sudo yum install -y yum-utils \
 device-mapper-persistent-data \
 lvm2
```

02 设置docker仓库(使用阿里云镜像仓库)
    sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

【设置一下阿里云镜像加速器】
```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
"registry-mirrors": ["这边替换成自己的实际地址"]
}
EOF
sudo systemctl daemon-reload
```

03 安装docker

```
yum install -y docker-ce-18.09.0 docker-ce-cli-18.09.0 containerd.io
```

04 启动docker
    sudo systemctl start docker && sudo systemctl enable docker

1.5 修改 hosts 文件

(1)master

```
# 设置master的hostname，并且修改hosts文件
sudo hostnamectl set-hostname m

vi /etc/hosts
192.168.2.106 m
192.168.2.240 w1
192.168.2.149 w2
```

(2)两个 worker

```
# 设置worker01/02的hostname，并且修改hosts文件
sudo hostnamectl set-hostname w1
sudo hostnamectl set-hostname w2

vi /etc/hosts
192.168.2.106 m
192.168.2.240 w1
192.168.2.149 w2
```

(3)使用 ping 测试一下

1.6 提前配置系统设置

```
# (1)关闭防火墙
systemctl stop firewalld && systemctl disable firewalld
```

```
# (2)关闭selinux
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

# (3)关闭swap
swapoff -a
sed -i '/swap/s/^\(.*\)$/#\1/g' /etc/fstab

# (4)配置iptables的ACCEPT规则
iptables -F && iptables -X && iptables -F -t nat && iptables -X -t nat && iptables -P FORWA
D ACCEPT

# (5)设置系统参数
cat <<EOF >  /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sysctl --system
```

1.7 安装 kubeadm、kubelet、kubectl

(1)配置 yum 源

```
cat <<EOF > /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
EOF
```

(2)安装 kubeadm&kubelet&kubectl

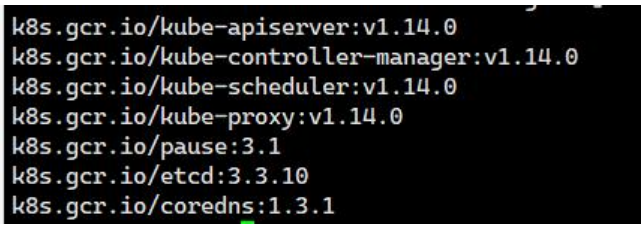
```
yum install -y kubeadm-1.14.0-0 kubelet-1.14.0-0 kubectl-1.14.0-0
```

(3)docker 和 k8s 设置同一个 cgroup

```
# docker
vi /etc/docker/daemon.json
添加(注意json格式) "exec-opts": ["native.cgroupdriver=systemd"],

systemctl restart docker

# kubelet，这边如果发现输出directory not exist，也说明是没问题的，大家继续往下进行即可
sed -i "s/cgroup-driver=systemd/cgroup-driver=cgroupfs/g" /etc/systemd/system/kubelet.se
vice.d/10-kubeadm.conf

systemctl enable kubelet && systemctl start kubelet
```

1.8 proxy/pause/scheduler 等国内镜像

(1)查看 kubeadm 使用的镜像 kubeadm config images list 可以发现这里都是国外的镜像



(2)解决国外镜像不能访问的问题

创建 kubeadm.sh 脚本，用于拉取镜像/打 tag/删除原有镜像

```
#!/bin/bash

set -e

KUBE_VERSION=v1.14.0
KUBE_PAUSE_VERSION=3.1
ETCD_VERSION=3.3.10
CORE_DNS_VERSION=1.3.1

GCR_URL=k8s.gcr.io
ALIYUN_URL=registry.cn-hangzhou.aliyuncs.com/google_containers

images=(kube-proxy:${KUBE_VERSION}
kube-scheduler:${KUBE_VERSION}
kube-controller-manager:${KUBE_VERSION}
kube-apiserver:${KUBE_VERSION}
pause:${KUBE_PAUSE_VERSION}
etcd:${ETCD_VERSION}
coredns:${CORE_DNS_VERSION})

for imageName in ${images[@]} ; do
  docker pull $ALIYUN_URL/$imageName
  docker tag  $ALIYUN_URL/$imageName $GCR_URL/$imageName
  docker rmi $ALIYUN_URL/$imageName
done
```

运行脚本和查看镜像

```
# 运行脚本
sh ./kubeadm.sh

# 查看镜像
docker images
```

将这些镜像推送到自己的阿里云仓库【可选，根据自己实际的情况】

```
# 登录自己的阿里云仓库
docker login --username=xxx registry.cn-hangzhou.aliyuncs.com

#!/bin/bash

set -e
```

```
KUBE_VERSION=v1.14.0
KUBE_PAUSE_VERSION=3.1
ETCD_VERSION=3.3.10
CORE_DNS_VERSION=1.3.1

GCR_URL=k8s.gcr.io
ALIYUN_URL= 此处换成自己阿里云的镜像仓库

images=(kube-proxy:${KUBE_VERSION}
kube-scheduler:${KUBE_VERSION}
kube-controller-manager:${KUBE_VERSION}
kube-apiserver:${KUBE_VERSION}
pause:${KUBE_PAUSE_VERSION}
etcd:${ETCD_VERSION}
coredns:${CORE_DNS_VERSION})

for imageName in ${images[@]} ; do
  docker tag $GCR_URL/$imageName $ALIYUN_URL/$imageName
  docker push $ALIYUN_URL/$imageName
  docker rmi $ALIYUN_URL/$imageName
done
```

运行脚本 sh ./kubeadm-push-aliyun.sh

1.9 kube init 初始化 master

(1)kube init 流程

01-进行一系列检查，以确定这台机器可以部署kubernetes

02-生成kubernetes对外提供服务所需要的各种证书可对应目录
/etc/kubernetes/pki/*

03-为其他组件生成访问kube-ApiServer所需的配置文件

```
ls /etc/kubernetes/
admin.conf  controller-manager.conf  kubelet.conf  scheduler.conf
```

04-为 Master组件生成Pod配置文件。

```
ls /etc/kubernetes/manifests/*.yaml
kube-apiserver.yaml
kube-controller-manager.yaml
kube-scheduler.yaml
```

05-生成etcd的Pod YAML文件。

```
ls /etc/kubernetes/manifests/*.yaml
kube-apiserver.yaml
kube-controller-manager.yaml
kube-scheduler.yaml
etcd.yaml
```

06-一旦这些 YAML 文件出现在被 kubelet 监视的/etc/kubernetes/manifests/目录下，kubelet就自动创建这些yaml文件定义的pod，即master组件的容器。master容器启动后，kubeadm会通过检localhost：6443/healthz这个master组件的健康状态检查URL，等待master组件完全运行起来

07-为集群生成一个bootstrap token

08-将ca.crt等 Master节点的重要信息，通过ConfigMap的方式保存在etcd中，工后续部署node节使用

09-最后一步是安装默认插件，kubernetes默认kube-proxy和DNS两个插件是必须安装的

(2)初始化 master 节点

官网：https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/

注意： **此操作是在主节点上进行**

```
# 本地有镜像
kubeadm init --kubernetes-version=1.14.0 --apiserver-advertise-address=192.168.2.106 --po
-network-cidr=10.244.0.0/16
【若要重新初始化集群状态：kubeadm reset，然后再进行上述操作】
```

(3)根据日志提示

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

(4)查看 pod 验证一下

等待一会儿，同时可以发现像 etc，controller，scheduler 等组件都以 pod 的方式安装成功了

注意： coredns 没有启动，需要安装网络插件

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

(5)健康检查

```
curl -k https://localhost:6443/healthz
```

1.10 部署 calico 网络插件

选择网络插件：https://kubernetes.io/docs/concepts/cluster-administration/addons/

calico 网络插件：https://docs.projectcalico.org/v3.9/getting-started/kubernetes/

calico，同样在master节点上操作

```
####  在k8s中安装calico
kubectl apply -f https://docs.projectcalico.org/v3.9/manifests/calico.yaml

####  确认一下calico是否安装成功
kubectl get pods --all-namespaces -w
```

1.11 kube join

记得保存初始化 master 节点的最后打印信息【注意这边大家要自己的，下面我的只是一个参考】

```
kubeadm join 192.168.2.106:6443 --token 2x0odw.1z8pda6zlov83z0u \
    --discovery-token-ca-cert-hash sha256:ca77eaa6ddaf0fec34894c521a139a1a444c35b0cd1
39ebe6c5080ffb3c8b97
```

(1)在 woker01 和 worker02 上执行上述命令

(2)在 master 节点上检查集群信息

```
NAME   STATUS   ROLES    AGE   VERSION
m      Ready    master   8d    v1.14.0
w1     Ready    <none>   8d    v1.14.0
w2     Ready    <none>   8d    v1.14.0
```

到此为止我们已经成功搭建了我们第一个 kubernetes 集群。