**16\_01k8s**

**一 什么是k8s**

1.1 Kubernetes的名字来自希腊语，意思是“舵手” 或 “领航员”。K8s是将8个字母“ubernete”替换为“8”的缩写。

1.2 K8S的创造者，是众人皆知的行业巨头——Google。

1.3 然而K8S并不是一件全新的发明。它的前身，是Google自己捣鼓了十多年的Borg系统。

1.4 K8S是2014年6月由Google公司正式公布出来并宣布开源的。

**二 为什么要使用k8s**

2.1 就在Docker容器技术被炒得热火朝天之时，大家发现，如果想要将Docker应用于具体的业务实现，是存在困难的——编排、管理和调度等各个方面，都不容易。于是，人们迫切需要一套管理系统，对Docker及容器进行更高级更灵活的管理，就在这个时候，K8S出现了

2.2 Kubernetes是Google 2014年创建管理的，是Google 10多年大规模容器管理技术Borg的开源版本。

2.3 K8s 是容器集群管理系统，是一个开源的平台，可以实现容器集群的自动化部署、自动扩缩容、维护等功能

**三 什么时候使用k8s**

- 快速部署应用

- 快速扩展应用

- 无缝对接新的应用功能

- 节省资源，优化硬件资源的使用

**四 在什么情况下使用k8s**

可移植: 支持公有云，私有云，混合云，多重云（multi-cloud）

可扩展: 模块化, 插件化, 可挂载, 可组合

自动化: 自动部署，自动重启，自动复制，自动伸缩/扩展

**五 怎么使用k8s**

Build, Ship and Run （创建，发送和运行）\*\*

Build once，Run anywhere（搭建一次，到处能用）\*\*

Docker技术的三大核心概念，分别是：

1. 镜像（Image）

2. 容器（Container）

3. 仓库（Repository）

**六 k8s 架构**

一个K8S系统，通常称为一个K8S集群（Cluster）,这个集群主要包括两个部分：

**6.1 一个Master节点（主节点）**

**Master节点包括API Server、Scheduler、Controller manager**、etcd。

**API Server**: 整个系统的对外接口，供客户端和其它组件调用，相当于“营业厅”。

**Scheduler**: 负责对集群内部的资源进行调度，相当于“调度室”。

**Controller manager**: 负责管理控制器，相当于“大总管”。

**6.2 一群Node节点（计算节点）**

**Node节点包括Docker、kubelet、kube-proxy、Fluentd、kube-dns（可选），还有就是Pod**。

**Pod**: 是Kubernetes最基本的操作单元。

一个Pod代表着集群中运行的一个进程，它内部封装了一个或多个紧密相关的容器。

除了Pod之外，K8S还有一个\*\*Service\*\*，一个Service可以看作一组提供相同服务的Pod的对外访问接口

**Docker**: 不用说了，创建容器的。

**Kubele**t: 主要负责监视指派到它所在Node上的Pod，包括创建、修改、监控、删除等。

**Kube-proxy**: 主要负责为Pod对象提供代理。

**Fluentd**: 主要负责日志收集、存储与查询。

**七 安装配置**

**7.1 设置 ip 和主机名**

/etc/hosts

**7.2 所有主机开启路由转发**

vim /etc/sysctl.conf

net.ipv4.ip\_forward=1

**7.3 配置 yum 源**

使用华为的yum源

将/linux-soft/04/kubernetes/\*传到跳板机ftp下,并生成repodata目录,并以此生成yum源

**7.4 配置私有仓库** registry,端口为5000

yum install docker-distribution

systemctl restart docker-distribution

systemctl enable docker-distribution

**7.5 管理镜像使用 docker**

yum install docker

/etc/sysconfig/docker

导入镜像

docker load -i centos.tar

docker load -i kubernetes-dashboard.tar

docker load -i pod-infrastructure.tar

启动容器运行centos,安装基本软件,生成myos:latest镜像

打标签:

docker tag myos:latest 192.168.1.3:5000/myos:latest

docker tag kubernetes-dashboard:latest \ 192.168.1.3:5000/kubernetes-dashboard:latest

docker tag pod-infrastructure:latest \

192.168.1.3:5000/pod-infrastructure:latest

上传打了标签的镜像到私有仓库

docker push 192.168.1.3:5000/myos:latest

docker push 192.168.1.3:5000/kubernetes-dashboard:latest

docker push 192.168.1.3:5000/pod-infrastructure:latest

私有仓库验证

curl http://192.168.1.3:5000/v2/\_catalog

**八 配置 kubernets - master**

**8.1 安装package:**

etcd #一款非关系型数据库

kubernetes-master

kubernetes-client

**8.2 生成的service:**

kube-apiserver

kube-controller-manager

kube-scheduler

etcd

**8.3 配置文件 conf:**

**/etc/etcd/etcd.conf** #数据库配置文件

6: ETCD\_LISTEN\_CLIENT\_URLS="http://192.168.1.3:2379"

#指定数据库地址

**/etc/kubernetes/config** #全局配置文件

22: KUBE\_MASTER="--master=http://192.168.1.3:8080"

#说明谁是master

**/etc/kubernetes/apiserver** #主配置文件

8: KUBE\_API\_ADDRESS="--insecure-bind-address=0.0.0.0"

#设置监听所有主机

17: KUBE\_ETCD\_SERVERS="--etcd-servers=http://192.168.1.3:2379"

#指定数据库地址

23: 删除 ServiceAccount 参数 #不使用证书

**/etc/kubernetes/controller-manager #仅查看**

**/etc/kubernetes/scheduler #仅查看**

**8.5 重启\开启自启服务,并验证**

systemctl restart kube-apiserver kube-controller-manager kube-scheduler etcd

systemctl enable kube-apiserver kube-controller-manager kube-scheduler etcd

kubectl get cs

kubectl get csr

kubectl get node

**九 配置 kubernets - minion(配置node主机)**

**9.1 安装package:**

kubernetes-node

docker

**9.2 生成的service:**

kubelet

kube-proxy

docker

**9.3 配置文件 conf:**

**/etc/sysconfig/docker**

4: 添加参数 --insecure-registry=192.168.1.3:5000 --add-registry 192.168.1.3:5000

**/etc/kubernetes/config**

22: KUBE\_MASTER="--master=http://192.168.1.3:8080"

**/etc/kubernetes/kubelet**

5: KUBELET\_ADDRESS="--address=0.0.0.0"

11: KUBELET\_HOSTNAME="--hostname-override=本机名称"

14: 添加 --kubeconfig=/etc/kubernetes/kubelet.kubeconfig

--pod-infra-container-image=pod-infrastructure:latest

**/etc/kubernetes/kubelet.kubeconfig(yaml格式)**

apiVersion: v1

kind: Config

clusters:

- cluster:

server: [http://192.168.1.3:8080](http://192.168.1.10:8080) #Master的IP

name: local

contexts:

- context:

cluster: local

name: local

current-context: local

**9.3 重启并开机自启服务**

systemctl enable docker kubelet.service kube-proxy

systemctl restart docker kubelet.service kube-proxy

**9.4 master主机验证**

kubectl get cs

kubectl get csr

kubectl get node

**十 网络**

**10.1 master**:

编辑配置文件:/etc/etcd/etcd.conf

6 ETCD\_LISTEN\_CLIENT\_URLS="http://0.0.0.0:2379"

重启服务

systemctl restart kube-apiserver kube-controller-manager kube-scheduler etcd

**10.2 master主机执行以下命令**

etcdctl mk /atomic.io/network/config \

'{"Network": "10.254.0.0/16", "Backend": {"Type": "vxlan"}}'

**10.3 安装minion**:

package: flannel(node主机安装)

**10.4 修改配置文件(所有主机):**

/etc/sysconfig/flanneld

4 FLANNEL\_ETCD\_ENDPOINTS="http://192.168.1.3:2379"

**10.5 停止docker服务,再重启服务,查看node主机IP**

systemctl stop docker

systemctl restart flanneld docker**(docker 后启动,先停止)**

[root@1-4\5\6 ~]# ifconfig

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 10.254.23.1

flannel.1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1450

inet 10.254.23.0

**十一 创建容器，测试**

**11.1 在任意node主机上创建容器,并查看容器的IP**

[root@1-4 ~]# docker run -it myos:latest

[root@49b6905899cb /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1450

inet 10.254.23.2

**11.2 在其他node主机上ping容器IP**

[root@1-6 ~]# ping 10.254.23.2

PING 10.254.23.2 (10.254.23.2) 56(84) bytes of data.

64 bytes from 10.254.23.2: icmp\_seq=1 ttl=63 time=1.93 ms

**11.3 在容器内ping其他node主机**

[root@49b6905899cb /]# ping 10.254.6.1

PING 10.254.6.1 (10.254.6.1) 56(84) bytes of data.

64 bytes from 10.254.6.1: icmp\_seq=1 ttl=63 time=1.82 ms

**十二 文件**

**/linux-soft/04/docker**

**baseos.yaml kube-dashboard.yaml**

**12.1 baseos.yaml**

---

apiVersion: extensions/v1beta1 #当前格式的版本

kind: Deployment #当前创建资源的类型， 当前类型是Deployment

metadata: #当前资源的元数据

name: **myos** #当前资源的名字 是元数据必须的项

spec: #是当前Deployment的规格说明

replicas: **4** #指当前创建的副本数量 默认不填 默认值就为‘1’

template: #定义pod的模板

metadata: #当前pod的元数据

labels: #至少顶一个labels标签，可任意创建一个 key:value

app: **myos**

spec: #当前pod的规格说明

containers: #容器

- name: centos #是容器的名字容器名字是必须填写的

image: **myos:latest** #镜像 镜像的名字和版本

stdin: true

tty: true

**12.2 使用baseos.yaml创建\删除容器**

**kubectl create\delete -f baseos.yaml**

[root@1-3 ~]# **kubectl create -f baseos.yaml**

deployment.extensions "test-os" created

**12.3 查看创建的容器**

[root@1-3 ~]# **kubectl get pod -o wide**

NAME READY STATUS RESTARTS AGE IP NODE

myos-7548d47cc4-cx2hx 1/1 Running 0 38m 10.254.49.3 1-5

myos-7548d47cc4-dnxgp 1/1 Running 0 38m 10.254.6.2 1-6

myos-7548d47cc4-gx78g 1/1 Running 0 38m 10.254.23.2 1-4

myos-7548d47cc4-hzpzk 1/1 Running 0 38m 10.254.49.2 1-5

**12.4 进入容器,ping其他容器**

[root@1-3 ~]# **kubectl exec -it** myos-7548d47cc4-cx2hx **/bin/bash**

[root@myos-7548d47cc4-cx2hx /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1450

inet 10.254.49.3

[root@myos-7548d47cc4-cx2hx /]# ping 10.254.6.2

[root@myos-7548d47cc4-cx2hx /]# ping 10.254.23.2

[root@myos-7548d47cc4-cx2hx /]# ping 10.254.49.2

#都能ping通

**12.5 kube-dashboard.yaml**

kind: Deployment

apiVersion: apps/v1beta2

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

replicas: 4

revisionHistoryLimit: 10

selector:

matchLabels:

k8s-app: kubernetes-dashboard

template:

metadata:

labels:

k8s-app: kubernetes-dashboard

spec:

containers:

- name: kubernetes-dashboard

image: **192.168.1.3:5000**/kubernetes-dashboard-amd64:v1.8.3

ports:

- containerPort: 9090

protocol: TCP

args:

- --apiserver-host=http://**192.168.1.3:8080**  #修改为Master的IP

volumeMounts:

- mountPath: /tmp

name: tmp-volume

livenessProbe:

httpGet:

path: /

port: 9090

initialDelaySeconds: 30

timeoutSeconds: 30

volumes:

- name: tmp-volume

emptyDir: {}

serviceAccountName: kubernetes-dashboard

# Comment the following tolerations if Dashboard must not be deployed on master

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

---

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

type: NodePort

ports:

- port: 80

targetPort: 9090

nodePort: 30090

selector:

k8s-app: kubernetes-dashboard

**12.6**  **执行kube-dashboard.yaml文件**

[root@1-3 ~]# **kubectl create -f kube-dashboard.yaml**

deployment.apps "kubernetes-dashboard" created

service "kubernetes-dashboard" created

**12.7 node上查看30090端口**

[root@1-4 ~]# ss -antulp | grep 30090

**12.8 华为云上购买一个公网IP,绑定到任意一台node主机上**

**12.9 网页访问公网IP:5000**

kubectl get nodes: 查看节点主机的状态,正常为Ready

kubectl get pod -o wide: 查看创建好了的容器

kubectl -n 空间名称 子命令 子参数

kubectl -n kube-system get pod #查看由kube-dashboard.yaml文件创建的容器

kubectl describe pods #查看容器的详细信息