## 开源容器应用自动化部署技术Kubernetes

主讲：Cat 老师

北京动力节点教育科技有限公司  
2020 • 北京

动力节点•版权所有•禁止传播

## 快速回顾

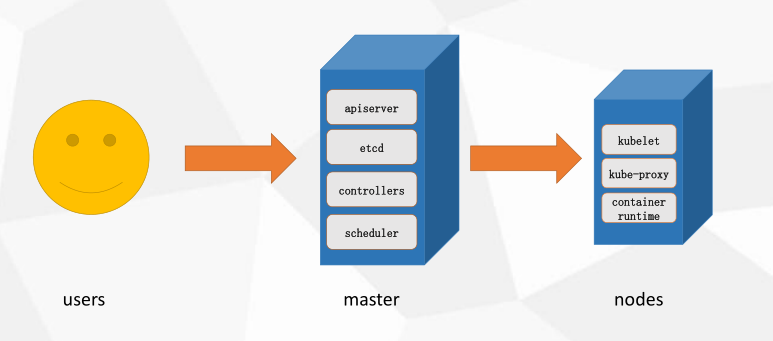
What is Kubernetes？

Kubernetes整体架构

Kubernetes环境搭建方式

Kubeadm部署Kubernetes

Kubernetes部署容器化应用



## Kubernetes部署“容器化应用”

何为“容器化应用”？

通俗点来说，就是你把一个程序放在docker里部署，这个docker应用就是容器化应用；

比如：在docker里面部署一个springboot，这个docker+springboot一起就是一个容器化应用；

在Docker里面部署一个nginx，这个docker+nginx一起就是一个容器化应用；

在Docker里面部署一个应用怎么部？

简单来说就是 镜像 -> 启动镜像得到一个Docker容器；

在k8s里面部署一个应用怎么部？

SpringBoot程序-->打包jar包或war包-->通过Dockerfile文件生成docker镜像-->通过k8s部署这个镜像-->部署完毕；

总结一下Kubernetes部署容器化应用的步骤：

1、制作镜像；（自己编写Dockerfile文件制作，或者 从仓库pull镜像）

2、通过控制器管理pod（其实就是把镜像启动得到一个容器，容器在pod里）

3、暴露应用，以便外界可以访问；

## 在Kubernetes集群中部署一个Nginx

1、制作镜像（有现成的，从官方仓库pull即可，不需要制作）

2、kubectl create deployment nginx --image=nginx  
3、kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort

4、访问应用；

查看一下，相关命令：

kubectl get node(s)

kubectl get service(s)

kubectl get deployment (deploy)

kubectl get pod(s)

//删除service

kubectl delete service nginx

//删除nginx的控制器

kubectl delete deployment nginx

//删除pod

kubectl delete pod nginx-6799fc88d8-zc48m（pod名字）

//kubectl命令帮助

kubectl --help

访问地址：<http://NodeIP:Port>

Docker镜像加速器：

/etc/docker/daemon.json

{

"registry-mirrors": ["https://gg3gwnry.mirror.aliyuncs.com"]

}

## 在Kubernetes集群中部署一个Tomcat

kubectl create deployment tomcat --image=tomcat

kubectl expose deployment tomcat --port=8080 --type=NodePort

访问地址：<http://NodeIP:Port>

## 在Kubernetes集群中部署SpringBoot应用

1. 项目打包（jar、war）-->可以采用一些工具git、maven、jenkins；
2. 制作Dockerfile文件，生成镜像；
3. kubectl create deployment your-springboot --image=你的镜像
4. 你的springboot就部署好了，是以docker容器的方式运行在pod里面的；

Mater控制Node --> service --> deployment(控制器) --> pod --> docker

### 自定义JDK镜像

FROM centos:latest

MAINTAINER cat

ADD jdk-8u251-linux-x64.tar.gz /usr/local/java

ENV JAVA\_HOME /usr/local/java/jdk1.8.0\_251

ENV CLASSPATH $JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

ENV PATH $PATH:$JAVA\_HOME/bin

CMD java -version

构建镜像：docker build -t jdk1.8.0\_251 .

运行镜像：docker run -d ac84bde53958 （镜像id）

### 构建项目镜像

FROM jdk1.8.0\_251

MAINTAINER cat

ADD 38-springboot-k8s-1.0.0.jar /opt

RUN chmod +x /opt/38-springboot-k8s-1.0.0.jar

CMD java -jar /opt/38-springboot-k8s-1.0.0.jar

构建镜像：docker build -t 38-springboot-k8s-1.0.0-jar .

### 空运行测试

kubectl create deployment springboot-k8s --image=38-springboot-k8s-1.0.0-jar --dry-run -o yaml

kubectl create deployment springboot-k8s --image=38-springboot-k8s-1.0.0-jar --dry-run -o json

kubectl create deployment springboot-k8s --image=38-springboot-k8s-1.0.0-jar --dry-run -o yaml > deploy.yaml

kubectl create deployment springboot-k8s --image=38-springboot-k8s-1.0.0-jar --dry-run -o json> deploy.json

### Yml文件方式部署（也可以通过命令部署）

kubectl apply -f boot-deploy.yaml (yaml是资源清单)

等价于：

kubectl create deployment springboot-k8s --image=38-springboot-k8s-1.0.0-jar

部署后查看一下：

kubectl get pods

注意：boot-deploy.yaml文件里面镜像从本地拉取；

containers:

- image: 38-springboot-k8s-1.0.0-jar

name: 38-springboot-k8s-1-0-0-jar-8ntrx

imagePullPolicy: Never

把镜像拉取策略改为Never；

查看pod详细信息：

kubectl describe pods 38-springboot-k8s-5c5c46d65f-4s52c（pod名称）

检查是否创建了deployments任务：kubectl get deployments

查看pod日志：kubectl logs 38-springboot-k8s-575778f6bf-drmbm（pod名称）

暴露服务端口：

kubectl expose deployment springboot-k8s --port=8080 --type=NodePort

### 命令方式部署

kubectl create deployment springboot-k8s --image=38-springboot-k8s-1.0.0-jar

## 部署Kubernetes Dashbaord

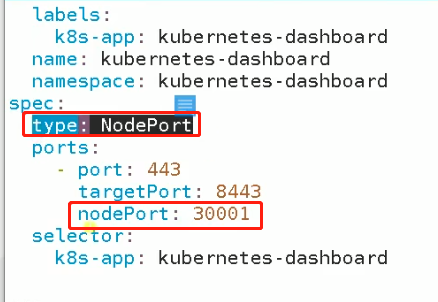
Kubernetes仪表盘是Kubernetes集群的基于Web的通用UI，它允许用户管理群集中运行的应用程序并对其进行故障排除，以及管理群集本身；

Github：<https://github.com/kubernetes/dashboard>

下载yaml的资源清单

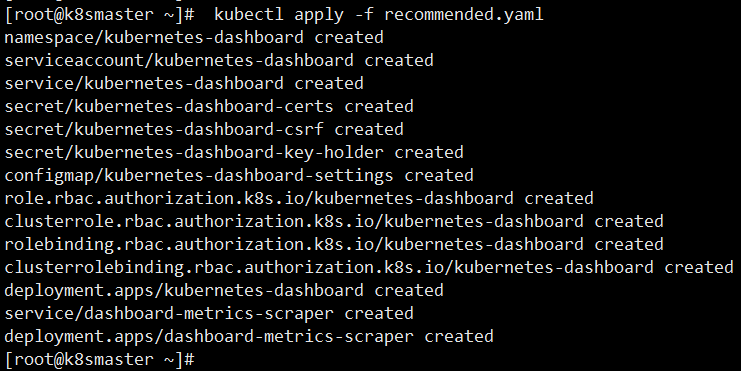
wget https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.4/aio/deploy/recommended.yaml

修改一下下载下面的recommended.yaml文件，修改红框的两个地方



应用yaml的资源清单

kubectl apply -f recommended.yaml （清单）



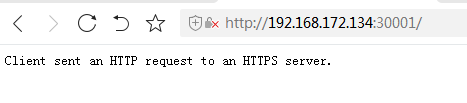
或者直接一步到位：

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.4/aio/deploy/recommended.yaml

查看一下pod是否成功，注意命名空间是在kubernetes-dashboard下面；

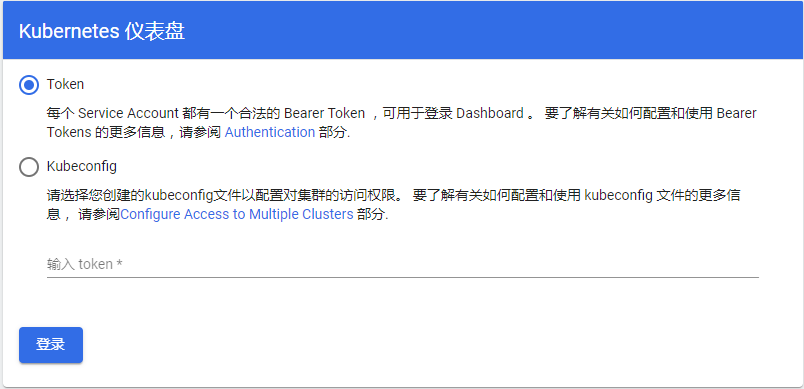
kubectl get pod -n kubernetes-dashboard

然后浏览器访问：



表示需要使用https访问；

<https://192.168.172.134:30001/>



需要输入token，token的生成采用下面的三条命令（固定的，照着操作即可，不用记住）：

kubectl create serviceaccount dashboard-admin -n kube-system

kubectl create clusterrolebinding dashboard-admin --clusterrole=cluster-admin --serviceaccount=kube-system:dashboard-admin

kubectl describe secrets -n kube-system $(kubectl -n kube-system get secret | awk '/dashboard-admin/{print $1}')

得到token；