**K8S及云原生应用服务workshop Lab Guide**

**K8S基础**

A picture containing clipart

Description automatically generated



2019/12/12

# Pod，Job，DaemonSet

## Pod

**目标：**

通过yaml文件创建一个pod。

**方法：**

新建一个名为<file-name>的yaml文件，例如pod-nginx.yaml，内容可参考下面：

apiVersion: v1

kind: Pod

metadata:

name: nginx

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

执行命令#kubectl create -f <file-name>

执行命令#kubectl get pod -o wide，查看pod状态。输出应该类似下图：

A close up of a logo

Description automatically generated

执行命令#kubectl describe pod <pod-name>，观察pod详细信息。

执行命令#curl http://<pod-ip>，确保能够获取到nginx欢迎页面。

## Job

**目标：**

通过yaml文件执行一个Job，输出“Hello Kubernetes!”。

**方法：**

新建一个名为<file-name>的yaml文件，例如job-hello-k8s.yaml，内容可参考下面：

apiVersion: batch/v1

kind: Job

metadata:

name: hello-k8s

spec:

template:

spec:

containers:

- name: hello-k8s

image: busybox

command: ['sh', '-c', 'echo Hello Kubernetes!']

imagePullPolicy: IfNotPresent

restartPolicy: Never

执行命令#kubectl create -f <file-name>

执行命令#kubectl get job查看job执行情况

A close up of a logo

Description automatically generated

执行命令#kubectl get pod查看job生成的pod

执行命令#kubectl logs <pod-name>查看pod的输出，输出应该为“Hello Kubernetes!”

## DaemonSet

**目标：**

通过yaml文件创建一个DaemonSet。

**方法：**

新建一个名为<file-name>的yaml文件，例如ds-nginx.yaml，内容可参考下面：

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: ds-nginx

spec:

selector:

matchLabels:

name: ds-nginx

template:

metadata:

labels:

name: ds-nginx

spec:

containers:

- image: nginx

name: ds-nginx

imagePullPolicy: IfNotPresent

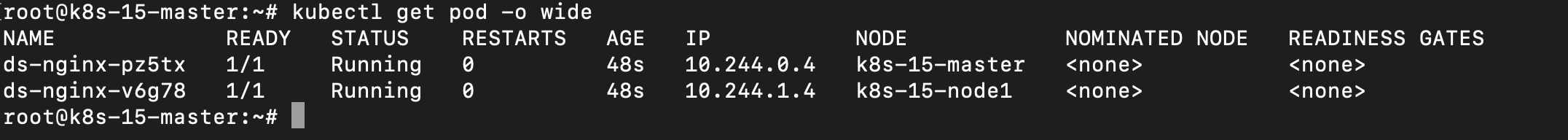
执行命令#kubectl create -f <file-name>

执行命令#kubectl get daemonset查看执行情况，结果应该如下：

A screenshot of a cell phone

Description automatically generated

执行命令#kubectl get pod -o wide查看DaemonSet生成的pod，应该能看到DaemonSet在每个Node都创建了一个pod。



实验结束后，可以通过一下命令清除实验生成的k8s对象：

#kubectl delete -f <file-name>

# 资源调度

## Node亲和调度

**目标：**

将pod调度到指定node。

**方法：**

执行命令# kubectl label node <node-name> <key>=<value>，给一个node打上标签。

编辑一个yaml文件，创建一个pod，并通过在yaml文件中加入nodeSelector让这个pod被调度到上述被标记的node。yaml文件示例：

apiVersion: v1

kind: Pod

metadata:

  name: nginx

  labels:

    app: nginx

spec:

  containers:

  - name: nginx

    image: nginx

    imagePullPolicy: Never

    ports:

    - containerPort: 80

  nodeSelector:

    <key>: <value>

执行命令#kubectl get pod -o wide，查看pod是否被调度到了指定node

## Taints

**目标：**

让一个node不再接受pod调度。

**方法：**

执行命令#kubectl taint nodes <node-name> <key>=<value>:NoSchedule，给一个node加上Taint。

创建一个DaemonSet，通过命令观察DaemonSet创建的pod情况，被加了Taint的node是否会生成pod？

执行命令#kubectl taint nodes <node-name> <key>:NoSchedule-，将刚才加上的Taint删除，重新观察DaemonSet和pod的状态。

# PV和PVC

**目标：**

通过PV和PVC实现pod中数据的持久化存储。

**方法：**

**创建PV**

新建一个名为<file-name>的yaml文件，例如pv-local.yaml，内容可参考下面：

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv-local

spec:

storageClassName: "local"

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data"

执行命令#kubectl create -f <file-name>

**创建PVC**

新建一个名为<file-name>的yaml文件，例如pvc-local.yaml，内容可参考下面：

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-local

spec:

storageClassName: "local"

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

执行命令#kubectl create -f <file-name>

**验证持久化存储**

新建一个名为<file-name>的yaml文件，例如pod-data1.yaml，内容可参考下面：

apiVersion: v1

kind: Pod

metadata:

name: data1

spec:

containers:

- image: busybox

name: busybox

imagePullPolicy: IfNotPresent

command: ["sleep", "3600"]

volumeMounts:

- name: data-local

mountPath: /data

restartPolicy: Never

nodeSelector:

name: k8s-master

volumes:

- name: data-local

persistentVolumeClaim:

claimName: pvc-local

*#为了确保Pod被调度到固定节点上，这里使用了NodeSelector，请根据需要自行修改yaml*

执行命令#kubectl create -f <file-name>

执行命令# kubectl exec -it data1 – sh，连接busybox 容器并打开一个shell。

在/data目录下新建一个文本文件，随意写点内容，例如hello world

A close up of a logo

Description automatically generated

删除这个pod，并根据以上yaml模版再创建一个pod，名字改为data2。

执行命令# kubectl exec -it data2 – sh，连接busybox 容器并打开一个shell。

查看/data目录下，之前data1创建的文本文件应该依然存在。



# Deployment

**目标：**

通过yaml文件创建一个Deployment，并且理解k8s的Rolling Update。

**方法：**

新建一个名为<file-name>的yaml文件，例如deployment-nginx.yaml，内容可参考下面：

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.7.9

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

执行命令#kubectl create -f <file-name>

执行命令#kubectl get deployment -o wide查看执行情况，结果应该如下：

A close up of a logo

Description automatically generated

执行命令#kubectl get pod -o wide查看Deployment生成的pod。

执行命令# kubectl scale deployment nginx-deployment --replicas 3，将部署扩容到3个副本。

执行命令#kubectl get deployment -o wide和#kubectl get pod -o wide查看变化。

执行命令#kubectl set image deployment nginx-deployment nginx=nginx:latest，更新部署的nginx镜像版本。

马上重复执行命令# kubectl get pod，观察k8s Rolling Update的方式。

执行命令#kubectl get deployment -o wide查看更新后的部署。

执行命令#kubectl rollout history deployment nginx-deployment，观察部署的更新记录。可以通过--revision=n参数查看详细情况。例如：# kubectl rollout history deployment nginx-deployment --revision=1

执行命令#kubectl rollout undo deployment nginx-deployment，将部署进行回滚。

执行命令#kubectl get deployment -o wide查看回滚后的部署。

可以尝试通过命令# kubectl rollout undo deployment nginx-deployment --to-revision=1将部署回滚到指定版本。

# Service

**目标：**

学习通过Service创建一个虚拟服务，并且将这个虚拟服务暴露到集群外部。

**方法：**

根据前面学习的内容，创建一个Deployment，docker镜像使用f5devcentral/f5-hello-world，容器端口为8080。

*注：docker镜像已经拉取，可以加入imagePullPolicy: IfNotPresent，防止从互联网下载镜像。*

apiVersion: apps/v1

kind: Deployment

metadata:

name: f5-hello-world

spec:

replicas: 2

selector:

matchLabels:

app: f5-hello-world

template:

metadata:

labels:

app: f5-hello-world

spec:

containers:

- name: f5-hello-world

image: f5devcentral/f5-hello-world

imagePullPolicy: IfNotPresent

ports:

- containerPort: 8080

部署完成后，通过命令查看部署状态

通过curl命令访问pod，确保web服务正常。

## ClusterIP Service

**目标：**

以默认的方式发布一个Service。

**方法：**

新建一个名为<file-name>的yaml文件，例如service-clusterip.yaml，内容可参考下面：

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

selector:

app: f5-hello-world

ports:

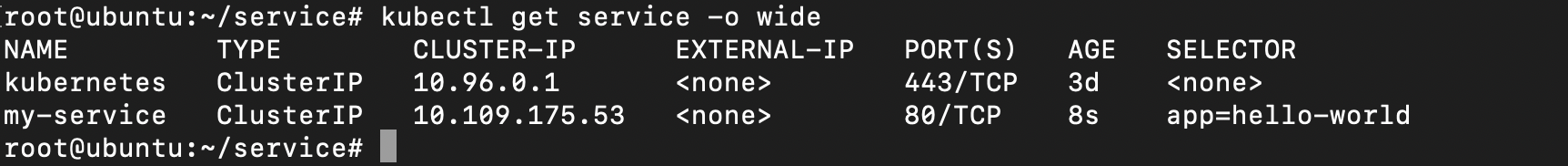
- protocol: TCP

port: 80

targetPort: 8080

执行命令#kubectl create -f <file-name>

执行命令#kubectl get service -o wide查看执行情况，结果应该如下：



执行命令#kubectl describe service <service-name>查看service的详细信息。

执行命令#curl http://<ClusterIP>，应该可以获取到Web页面。

执行命令#kubectl delete service <service-name>将服务删除。

## NodePort Service

**目标：**

将一个Deployment发布到集群外部，让Web应用可以通过外部网络访问。

**方法：**

新建一个名为<file-name>的yaml文件，例如service-nodeport.yaml，内容可参考下面：

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

type: NodePort

selector:

app: f5-hello-world

ports:

- port: 80

targetPort: 8080

nodePort: 30000

执行命令#kubectl create -f <file-name>

执行命令#kubectl get service -o wide查看执行情况，结果应该如下：

A screenshot of a cell phone

Description automatically generated

执行命令#kubectl describe service <service-name>查看service的详细信息。

执行命令#curl http://<ClusterIP>，应该可以获取到Web页面。

在笔记本电脑上打开浏览器，通过k8s虚拟机的IP和NodePort端口访问，应该可以正常打开Web页面。

A screenshot of a cell phone

Description automatically generated

执行命令#kubectl delete service <service-name>将服务删除。

# Ingress

## 安装Nginx-Ingress

**目标：**

安装Nginx ingress controller。

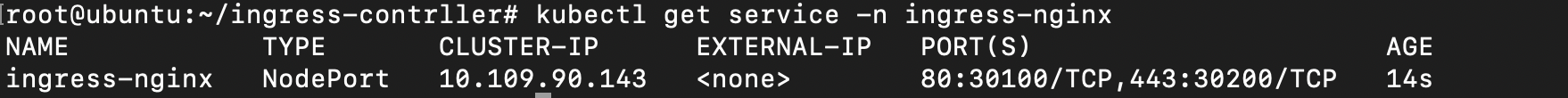
**方法：**

进入/root/ingress-controller/目录

执行命令#kubectl create -f mandatory.yaml ---部署ingress controller

执行命令#kubectl create -f service-nodeport.yaml ---部署service

执行命令#kubectl get service -n ingress-nginx，观察ingress controller的运行情况，同时观察本实验中ingress controller的服务暴露方式。



## 发布Ingress

**目标：**

通过Nginx ingress controller向集群外部发布 7层负载均衡服务。

**方法：**

按照前面学到的内容，发布两个ClusterIP类型的服务，docker镜像分别使用nginx和f5-hello-world。

首先在k8s虚拟机上通过curl命令确保这两个服务在集群内部可以正常被访问。

新建一个名为<file-name>的yaml文件，例如ingress-host-select.yaml，内容可参考下面：

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: ingress-host-select

annotations:

kubernetes.io/ingress.class: "nginx"

spec:

rules:

- host: nginx.will.com

http:

paths:

- path: /

backend:

serviceName: nginx-svc

servicePort: 80

- host: f5.will.com

http:

paths:

- path: /

backend:

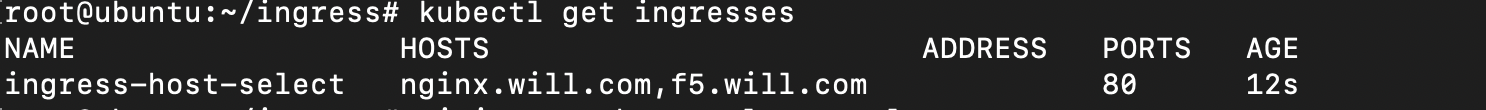
serviceName: my-service

servicePort: 8080

*#注：以上标红部分需要改成符合你自己环境的内容。*

执行命令#kubectl create -f <file-name>

执行命令#kubectl get ingress查看执行情况，结果应该如下：



执行命令#kubectl describe ingress <ingress-name>查看ingress的详细信息。

修改笔记本电脑的hosts文件，将ingress中发布的域名对应到k8s虚拟机的ip。

在笔记本电脑host上打开浏览器，通过以上两个域名和Ingress controller的暴露端口进行访问，应该可以分别打开nginx测试页面和f5-hello-world页面。