Day06 kubernetes笔记

课程主要内容：

1、kubernetes 指令 实现镜像部署

2、实现扩容

3、局域网访问

4、dns域名访问

5、负载均衡

6、service 外网访问

7、yaml资源文件部署

# 1、镜像部署（指令）

## 1.1、Myapp镜像部署

部署镜像命令： kubectl run deploymentName - -image=镜像地址(myapp) - - port=80

记录指令过程：

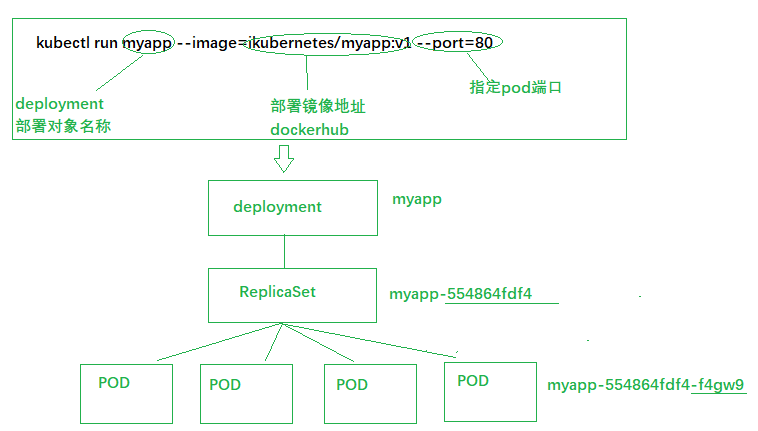
创建pod(部署服务) : kubectl run myapp --image=ikubernetes/myapp:v1 --port=80

查询pod创建详细信息： kubectl describe pod myapp-554864fdf4-f4gw9

查询pod： kubectl get pod

查询pod详细信息： kubectl get pod -o wide -n kube-system

Kubectl run指令执行后创建的资源对象拓扑图：



Kubectl run 首先创建deployment对象，kubectl run 指令执行后创建了3个资源对象（deployment,rs,pod）

注意： 上面这条指令没有指定副本数量，默认创建了1个pod。

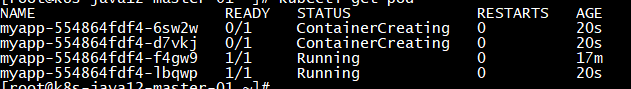
## 1.2、扩容

扩容指令：

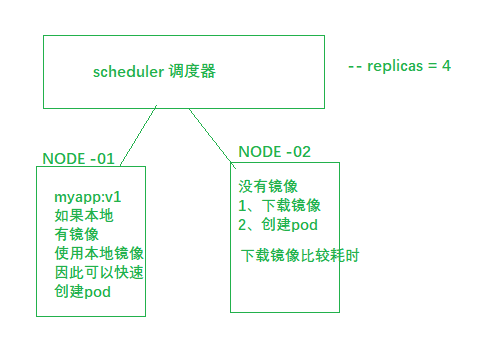
1、kubectl scale deployment myapp - -replicas=4

2、kubectl scale deployment/myapp - -replicas=4

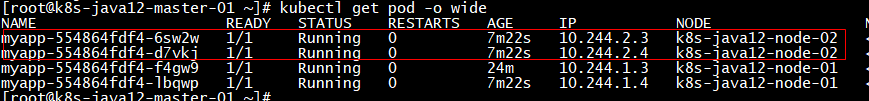
执行扩容指令后： 发现其中2个pod已经快速Running状态，其他2个一直在ContainerCreating，什么原因？？？



Pod创建慢的原因在于：node节点没有镜像，需要下载镜像，下载镜像比较耗时，因此pod创建需要花费更多时间。



过一段时间后，镜像下载完毕，pod立马创建成功。



扩展： 互联网项目瞬时流量增大，可以使用指令进行一键式扩容，非常简单，方便。

## 1.3、自愈

K8s管理的pod不仅仅可以一键式部署，一键式扩容,同时k8s管理的pod还具有自愈能力。也就是说当pod发生故障，宕机，k8s将会自动对pod件重建，因此k8s的pod的副本永远处于高可用状态。

试验：

删除所有的pod,查看pod是否还存在： kubectl delete pod –all

问题：

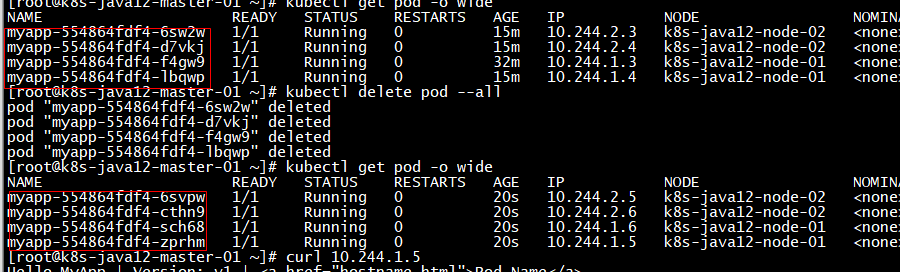
1、删除所有的pod后，pod还在否？

存在，k8s会立马的pod进行重建。

2、如果pod还存在，还是不是原来的pod??

不是原来的pod,这些pod都是k8s重新创建的。

可以发现，新创建的pod 和 旧的pod名称不一样的。但是同时发现k8s下的pod自愈能力，如果pod宕机，故障，k8s会重新创建pod.



## 1.4、负载均衡

Pod有多个副本，访问pod副本必须实现负载均衡访问，那么这个负载均衡访问是通过service来实现的。因此对一组pod副本创建一个service，还行负载均衡。

对一组pod副本创建service：

指令：

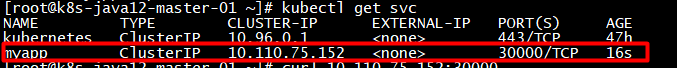
1、kubectl expose deployment myapp –port=30000 –target-port=80

2、kubectl expose deployment\myapp –port=30000 –target-port=80

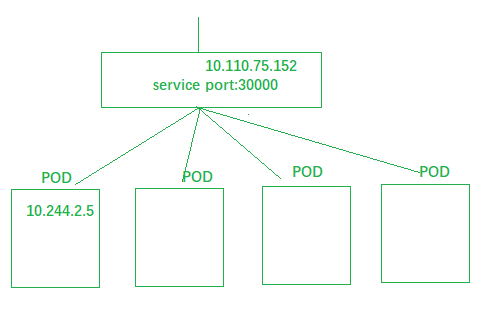
记录：

kubectl expose deployment myapp --port=30000 --target-port=80

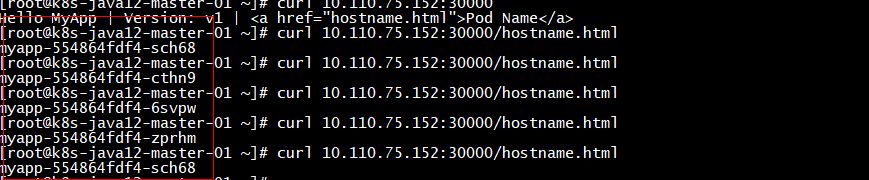
kubectl get svc



发现service资源对象已经创建成功，可以通过service访问pod.service作为4个pod副本访问的统一入口，来对pod副本进行负载均衡。



问题： 此时service访问pod到底采用什么样负载策略？？



通过测试发现，k8s默认使用沦陷策略。

Ip地址的映射规则：

Service IP :

10.110.75.152：30000

-> 10.244.2.5：80

-> 10.244.2.6：80

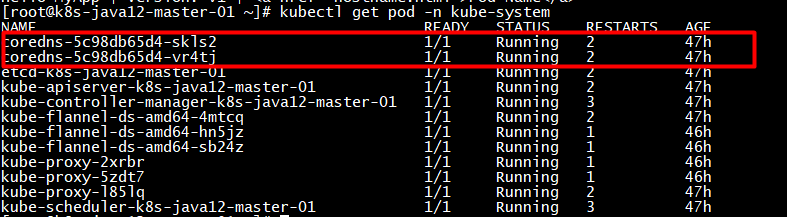
-> 10.244.1.5：80

-> 10.244.1.6：80

Service 通过ip 进行转发请求，然后通过iptables 负载均衡策略，默认使用轮询策略进行负载均衡。

## 1.5、Dns

安装k8s时候，安装dns组件，dns(域名解析服务器，根据域名解析出真实ip地址，通过dns可以通过服务名进行相互的访问)

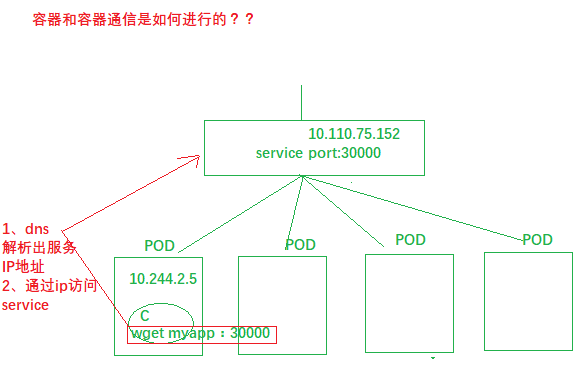


试验： 登录pod内部容器，在容器内部使用服务名称进行连接，查询是否能联通？？

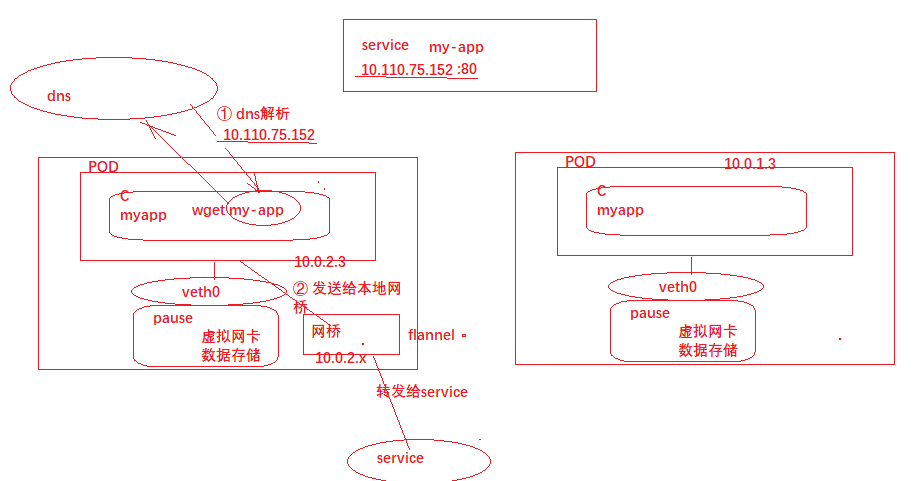
登录容器：

1、exec -it myapp-554864fdf4-6svpw – sh

2、wget myapp:30000(service名称) --- 测试通过，说明可以通过服务名称，访问到外部网络，因此dns服务器是ok的。

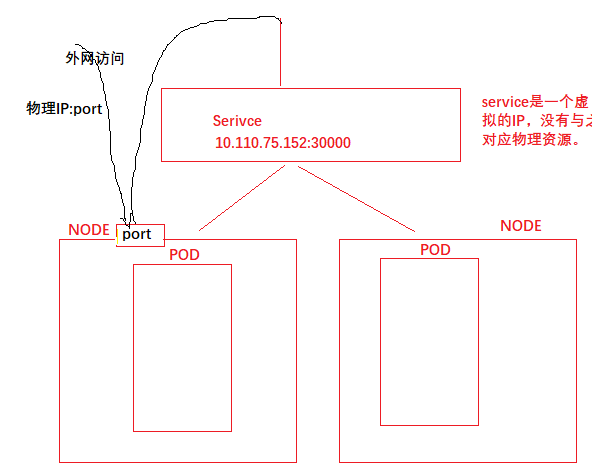


Dns解析域名流程：



## 1.6、外网访问

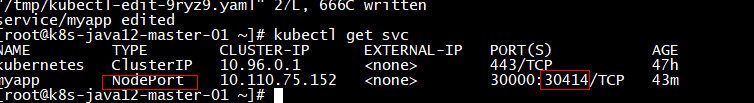
Service 和pod 之间的通信都属于局域网通信访问。无法对外网直接提供服务，service是一个虚拟化的概念，是一个虚拟IP地址，没有与之对应的物理硬件资料，无法直接对外网提供服务，必须借助于物理硬件资源进行数据包转发。



指令： kubectl edit svc myapp --- 编辑service对应资源对象配置文件，通过修改ip类型，来绑定物理端口。



可以发现30414就是我们所开辟物理端口，此端口可以在配置文件中自定义，也可以默认即可，默认端口范围：30000 – 32767



此时就可以通过物理ip+物理端口访问pod内部服务



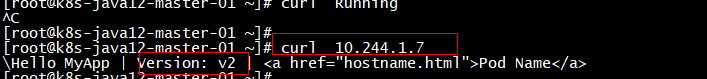
## 1.7、滚动更新

项目迭代，不停的发布新的版本，如何实现不停机更新。一键式更新操作

一键式更新：

kubectl set image deployment myapp –myapp=新的镜像版本

滚动更新，发现访问时候，已经是v2版本的服务。



# 2、YAML方式部署

## 2.1、资源清单

参考note笔记。

## 2.2、资源清单定义方式

#如果没有给定group名称，那么默认为core，可以使用kubectlapi-versions命令获取当前k8s版本上所有的apiversion版本信息（每个版本可能不同)

apiVersion: group/apiversion

#资源类别

kind: Pod / Service / Deployment

#资源元数据

metadata:

name:

namespace:

lables:

annotations: #主要目的是方便用户阅读查找

spec: #期望的状态（disired state)

status: #当前状态，本字段由Kubernetes自身维护，用户不能去定义

## 2.2、yaml部署

部署服务：

1、deployment部署 --- yaml配置文件

2、service部署 --- yaml配置文件

Nignx镜像部署：

Deployment配置文件： 三个部分

1、资源对象定义区（deployment,statefulset）

apiVersion: v1 # 自定义

kind: Deployment

metadata:

name: nginx-deployment

namespace: default

2、rs资源对象定义区

spec:

replicas: 3 # rs控制副本数量，指定副本数量为3

selector:

matchLabels:

app: mynginx

release: stable

env: test

3、pod资源对象定义区

template:

metadata:

labels:

app: mynginx

release: stable

env: test

spec:

containers:

- name: my-nginx

image: nginx:v1

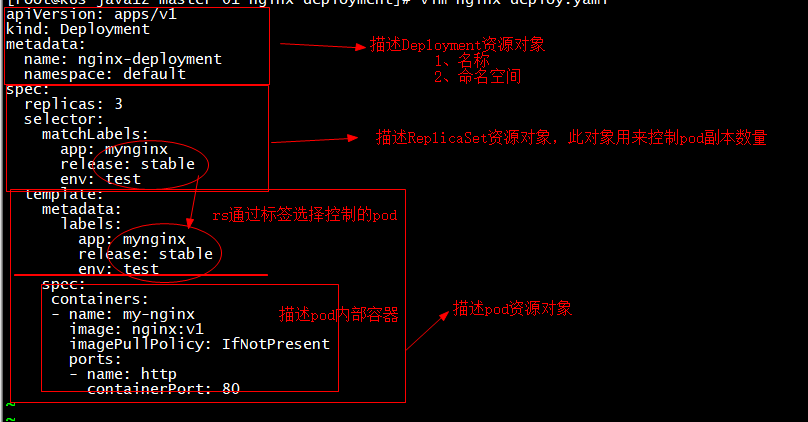
imagePullPolicy: IfNotPresent

ports:

- name: http

containerPort: 80

deployment资源对象部署：配置文件描述



Service配置文件：

1、service对象定义

2、选择器—和pod进行关联

kind: Service

metadata:

name: nginx-svc

namespace: default

spec:

type: ClusterIP

selector:

app: mynginx

release: stable

env: test

ports:

- name: http

port: 80

targetPort: 80

service 定义，service 和pod关联关系。



实际工作中：

1、可视化界面部署方式（rancher, dashboard）

2、指令方式

3、yaml文件方式

Jenkins + docker + git + k8s 线上部署ci/cd