

10.在k8s上运行第一个应用

1.kubectl run命令

2.什么是Pod

3.背后的原理

4.访问应用

上一节我们学习了如何使用Minikube搭建一个k8s环境，这一节开始我们来学习如何将之前创建的Node.js应用程序部署到这个k8s环境里

在部署之前，通常我们都会先准备好YAML或者JSON格式的清单（k8s中称作manifest），这个清单详细说明了我们要部署的组件。但是为了让大家更快进入到k8s的世界，我还不打算讲解k8s中的组件类型，先让我们通过一个简单的例子来了解如何让程序先运行起来。我的个人学习理念是先窥全貌、动手实践再深入了解，你都不知道困住你的这个庞然大物是个什么东西，就想去先去了解内部细节，只会让自己一头雾水、理解缓慢。

1.kubectl run命令

现在开始运行之前已经发布到阿里云的镜像，执行如下命令让其在k8s中运行：

```
kubectl run test1 --image=registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0 --port=8080
```

```
[david@dhr-demo root]$ kubectl run test1 --image=registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0 --port=8080
pod/test1 created
```

kubectl run命令会为我们创建一个名为test1的Pod（关于pod下面有介绍）。

查看Pod的状态：

```
kubectl get pods
```

```
[david@dhr-demo root]$ kubectl get pods
NAME    READY   STATUS             RESTARTS   AGE
test1   0/1     ContainerCreating   0          8s
```

可以看到当前状态为ContainerCreating，意味着正在创建容器。

过一会儿再执行这个命令，可以看到状态变为Running了：

```
[david@dhr-demo root]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
test1   1/1     Running   0          2m
```

之所以Pod的状态没有立刻变成Running，是因为Pod所在的工作节点需要先下载容器镜像，当下载完成后就会在Pod中创建容器，然后才会变成Running状态。

可能到此处有些同学会有些疑惑，这个Pod是什么？我们的程序是运行在容器里吗？那么容器在哪里可以看到？能不能有个命令显示所有正在运行的容器？然而kubernetes并不是以容器为单位来进行管理的，它不会直接与容器打交道，而是使用Pod。

2.什么是Pod


Pod就是一组紧密关联的容器的集合，它可以包含一个或者多个容器，还包含了存储、网络等各个容器可以共享的资源。这些容器总是在同一个Linux命名空间下运行在同一台工作节点上。

一个Pod只包含一个容器是最常用的应用方式。除非特别需要，比如应用之间耦合度比较高，一般都不推荐使用多容器Pod的方式。不过对于包含多个容器的Pod，Kubernetes能够保证这些容器都运行在同一台物理主机或虚拟主机中。Pod就像一个独立的逻辑机器，它有自己的IP、主机名和进程等等。同一个Pod中的容器共享IP地址和端口范围，容器之间可以通过 localhost 互相访问。

更多关于Pod的知识，后面我们再专门进行介绍。

通过执行如下命令可以列出：

```
kubectl get pods
```



NAME	READY	STATUS	RESTARTS	AGE
test1	1/1	Running	0	18h

上图中的READY列显示的就是这个名为test1的Pod中的容器数，以及处于READY状态的容器数。因为当前这个Pod只有一个容器，所以显示的都是1。

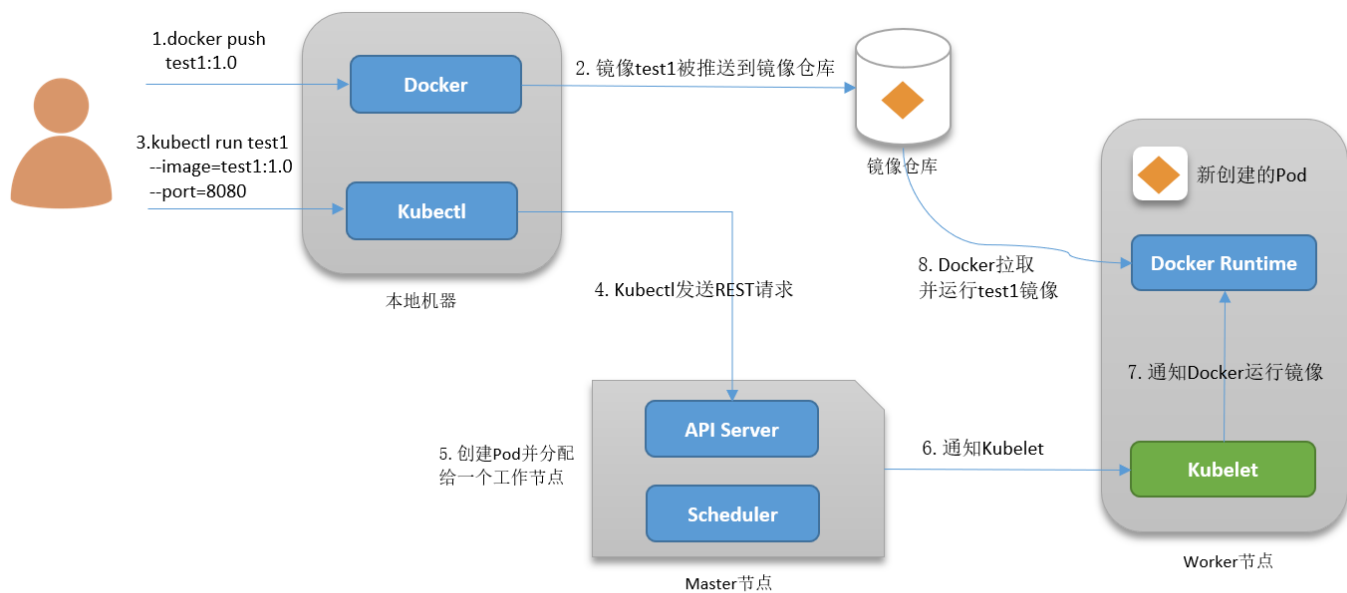
如果想要探查Pod更多的信息，可以使用如下命令：

```
kubectl describe pod
```

如果Pod的状态一直未变成Running，就可以通过这个命令查看具体的原因，有可能就是Kubernetes在拉取镜像的时候出错了。

3.背后的原理

上面通过kubectl run命令创建了一个Pod，可以通过下图来了解具体的原理：



- 1.开发人员在本地机器上执行推送镜像test1:1.0的命令
- 2.镜像被推送到镜像仓库（Docker Hub、阿里云等等）
- 3.开发人员在本地机器上运行kubectl run命令
- 4.Kubectl向k8s的API Server发送REST HTTP请求
- 5.创建一个Pod并通过Scheduler分配到某个工作节点
- 6.发送消息给Kubelet
- 7.某个工作节点上的Kubelet发现新创建的Pod被分配给了自己，于是通知Docker去拉取镜像
- 8.Docker去镜像仓库拉取镜像并在这个Pod中创建和运行容器

4.访问应用

Pod已经运行起来了，那如何才能访问到我们创建的Node.js应用程序呢？我们知道每个Pod其实都是有自己独立的网路环境，有自己的IP地址和主机名，但是这个地址是K8s集群内部的地址，无法在集群外部访问。因此需要想办法将我们的应用对外暴露出去。

可以使用一个叫做**Service**的Kubernetes对象来完成这个任务。为了演示，现在我们创建一个类型为LoadBalancer的Service对象：

```
kubectl expose pod test1 --type=LoadBalancer --name test1-http
```

```
[david@dhr-demo root]$ kubectl expose pod test1 --type=LoadBalancer --name test1-http
service/test1-http exposed
[david@dhr-demo root]$
```

通过如下命令可以查看刚创建的Service对象：

```
kubectl get services
```

```
[david@dhr-demo root]$ kubectl get services
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP    10.96.0.1     <none>         443/TCP        22h
test1-http   LoadBalancer 10.99.248.242 <pending>      8080:30768/TCP 115s
```

创建类型为LoadBalancer的Service会自动创建一个外部的**负载均衡器 (load balancer)**，我们可以通过这个负载均衡器的公共IP访问Pod。

test1-http就是我们刚才创建的服务。EXTERNAL-IP列显示的是pending状态，因为创建负载均衡器需要花些时间。一旦负载均衡器创建好了之后，EXTERNAL-IP列就会显示分配的外部IP地址。但是我们发现EXTERNAL-IP的状态似乎一直是pending。

```
[david@dhr-demo root]$ kubectl get services
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP    10.96.0.1     <none>         443/TCP        22h
test1-http   LoadBalancer 10.99.248.242 <pending>      8080:30768/TCP 23m
```

因为我们当前使用的是Minikube，它是不支持LoadBalancer类型的Service的，所以上面的test1-http一直获取不到外部IP地址。但是我们可以发现PORT(S)列其实是有端口映射的，说明通过30768端口是可以访问到这个服务的。

可以通过如下命令来获取外部IP地址：

minikube service test1-http

```
[david@dhr-demo root]$ minikube service test1-http
-----
| NAMESPACE | NAME   | TARGET PORT | URL                  |
|-----|-----|-----|-----|
| default   | test1-http | 8080        | http://172.17.0.3:30768 |
|-----|-----|-----|-----|
* Opening service default/test1-http in default browser...
- http://172.17.0.3:30768
```

这里的172.17.0.3其实就是名为minikube的节点的地址

```
[david@dhr-demo root]$ kubectl get node
NAME     STATUS    ROLES    AGE   VERSION
minikube Ready     master   23h   v1.19.0
```

```
[david@dhr-demo root]$ kubectl describe pod test1
Name:         test1
Namespace:    default
Priority:      0
Node:         minikube/172.17.0.3
Start Time:   Sat, 07 Nov 2020 17:26:18 +0800
Labels:       run=test1
Annotations:  <none>
Status:       Running
IP:           172.18.0.3
IPs:
  IP: 172.18.0.3
Containers:
  test1:
    Container ID:  docker://bc804d1de9e429bd2b3e7d4d1c915d72412b446105d4296218ff4d9f8a06425a
    Image:         registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0
    Image ID:      docker-pullable://registry.cn-shanghai.aliyuncs.com/david-ns01/test1@sha256:672d7b558e4b6e2ee9
```

通过curl命令来向我们的应用程序发送请求：

curl <http://172.17.0.3:30768>

```
[david@dhr-demo root]$ curl http://172.17.0.3:30768
已发送消息至: test1
[david@dhr-demo root]$
```

如果将service删除掉后再访问就会失败:

kubectl delete service test1-http

```
[david@dhr-demo root]$ kubectl delete service test1-http
service "test1-http" deleted
[david@dhr-demo root]$ curl http://172.17.0.3:30768
curl: (7) Failed connect to 172.17.0.3:30768; Connection refused
[david@dhr-demo root]$
```