26.连接外部服务

之前我们讨论的服务在集群中是由一个或多个pod在后端支撑。但是有时候我们也希望利用Kubernetes的Service特性将外部的服务暴露给Kubernetes集群内部，让这个Service将连接重定向到外部服务的IP和端口，而不是重定向到集群中的pod。
通过这种方式，我们可以利用服务负载均衡和服务发现的功能。集群中运行的客户端pod可以像连接内部服务一样连接到外部服务。首先来了解什么是Endpoints。

什么是服务Endpoints

服务与pod不是直接相连的，而是在它们之间有一个被称为Endpoints的资源。可以使用kubectl desribe查看：
kubectl describe svc test-svc

```
[david@dhr-demo root]$ kubectl describe svc test-svc
Name:              test-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=test1
Type:              ClusterIP
IP:                10.106.143.212
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         172.18.0.10:8080,172.18.0.8:8080,172.18.0.9:8080
Session Affinity:  None
Events:            <none>
[david@dhr-demo root]$
```

上图中Selector：app=test1表示服务的pod标签选择器，用于创建Endpoints列表。
Endpoints:172.18.0.10:8080,172.18.0.8:8080,172.18.0.9:8080代表pod列表，其中包含每个pod的IP和端口，表示该服务的所有endpoint。
Endpoints资源就是暴露一个服务的IP地址和端口的列表，与其他Kubernetes资源类似，可以使用kubectl get显示它的基本信息：
kubectl get endpoints test-svc

```
[david@dhr-demo root]$ kubectl get endpoints test-svc
NAME       ENDPOINTS                                            AGE
test-svc   172.18.0.10:8080,172.18.0.8:8080,172.18.0.9:8080   5d11h
[david@dhr-demo root]$
```

虽然在服务的spec中定义了pod选择器，但是在重定向传入的连接时不会直接使用它。Pod选择器是用于构建一个IP和端口列表，然后将其存储在Endpoints资源中。当一个客户端连接服务时，服务代理就会从这些IP和端口对中选择一个，然后将传入的连接重定向到监听这个地址的服务器。

手动配置Endpoint

在Kubernetes中，Endpoints和service是两个独立的资源。将服务的Endpoints与服务本身解耦合使我们能够手动配置和更新它们。

如果创建一个没有指定pod选择器的服务，Kubernetes就不会为我们创建Endpoints资源，因为没有了pod选择器，Kubernetes就没法知道服务中应该包含哪些pod。

如果要创建一个服务并希望手动配置endpoints，我们需要创建一个Service和一个Endpoints资源。

首先创建一个不指定选择器的服务：

```
apiVersion: v1
kind: Service
metadata:
  name: external-service
spec:
  ports:
  - port: 80
```

在上面的YAML文件中，我们定义了一个名为external-service的服务，在端口80上接收传入的连接，并且没有指定pod选择器。

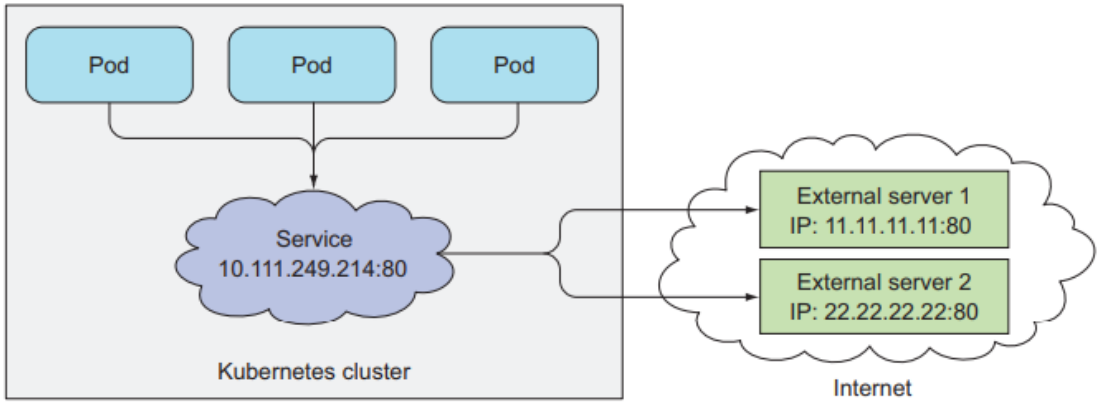
然后为这个服务创建一个Endpoints资源。

需要注意的是，Endpoints是一个独立的资源，不是服务的一个属性。由于我们创建了一个不带pod选择器的服务，所以相应的Endpoints资源就不会被自动创建，因此需要手动创建它：

```
apiVersion: v1
kind: Endpoints
metadata:
  name: external-service
subsets:
  - addresses:
    - ip: 11.11.11.11
    - ip: 22.22.22.22
    ports:
    - port: 80
```

上面定义的Endpoints对象需要与服务具有相同的名称，并且包含该服务的目标IP地址和端口列表。addresses属性定义了endpoints的所有IP，服务会将传入的连接重定向到这些IP地址；ports属性定义了endpoint的目标端口。当创建好这个Service和Endpoints资源后，就可以正常使用这个服务，就像使

用具有pod选择器的服务一样。如果我们在这个服务被创建之后创建了一个容器，那么它会包含该服务的环境变量，而且所有流入到IP:Port对的连接都会在该服务的endpoints之间被负载均衡处理。
如下图所示，三个pod消费一个带有两个外部endpoints的服务：



如果决定将外部服务迁移到Kubernetes集群中的pod中，我们可以为这个服务添加一个pod选择器，从而对Endpoints进行自动管理。反之也是一样——将选择器从服务中移除，Kubernetes将停止更新服务的Endpoints。这意味着一个服务的IP地址可以保持不变，而服务的实际实现已经改变了。

为外部服务创建别名

除了通过手动配置服务的Endpoints来暴露外部服务之外，还有一种更简单的方法，就是通过服务的完全限定域名（FQDN）来引用一个外部服务。

创建一个ExternalName服务

如果要创建一个服务资源作为外部服务的别名（类似CNAME），我们需要在创建服务资源时指定type属性值为ExternalName。
例如，我们假定有一个公共API: api.somecompany.com，可以按如下方式定义一个服务指向这个API：

```
apiVersion: v1
kind: Service
metadata:
  name: external-service
spec:
  type: ExternalName
  externalName: api.somecompany.com
  ports:
  - port: 80
```

上面的YAML定义中，“type: ExternalName”表示服务类型被设定为ExternalName。“externalName: api.somecompany.com”表示服务实际的FQDN。

这个服务被创建后，pod可以通过external-service.default.svc.cluster.local域名（或者external-service）连接到外部服务，而不是使用服务实际的FQDN。
这种方式对消费该服务的那些pod隐藏了实际的服务名称以及地址，使我们能够在任何时候修改服务定义信息使其指向到不同的服务，只需要修改externalName属性或者将type改回ClusterIP且为该服务创建一个Endpoints对象——手动创建或者通过在服务中指定标签选择器使其自动创建。
ExternalName仅在DNS层面实现——为服务创建一个简单的CNAME DNS记录。因此连接这个服务的客户端将直接连接到外部服务，完全跳过了服务代理。由于这个原因，这种类型的服务甚至都不会获得集群IP。

> CNAME记录指向FQDN，而不是数字IP地址。