### 2.2.13 Service

Service主要用于Pod之间的通信，对于Pod的IP地址而言，Service是提前定义好并且是不变的资源类型。

2.2.13.1 基本概念

Kubernetes Pod具有生命周期的概念，它可以被创建、删除、销毁，一旦被销毁就意味着生命周期的结束。通过ReplicaSet能够动态地创建和销毁Pod，例如进行扩缩容和执行滚动升级。每个Pod都会获取到它自己的IP地址，但是这些IP地址不总是稳定和可依赖的，这样就会导致一个问题：在Kubernetes集群中，如果一组Pod（比如后端的Pod）为其他Pod（比如前端的Pod）提供服务，那么如果它们之间使用Pod的IP地址进行通信，在Pod重建后，将无法再进行连接。

为了解决上述问题，Kubernetes引用了Service这样一种抽象概念：逻辑上的一组Pod，即一种可以访问Pod的策略——通常称为微服务。这一组Pod能够被Service访问到，通常是通过Label Selector（标签选择器）实现的。

举个例子，有一个用作图片处理的backend（后端），运行了3个副本，这些副本是可互换的，所以frontend（前端）不需要关心它们调用了哪个backend副本，然而组成这一组backend程序的Pod实际上可能会发生变化，即便这样frontend也没有必要知道，而且也不需要跟踪这一组backend的状态，因为Service能够解耦这种关联。

对于Kubernetes集群中的应用，Kubernetes提供了简单的Endpoints API，只要Service中的一组Pod发生变更，应用程序就会被更新。对非Kubernetes集群中的应用，Kubernetes提供了基于VIP的网桥的方式访问Service，再由Service重定向到backend Pod。

2.2.13.2 定义Service

一个Service在Kubernetes中是一个REST对象，和Pod类似。像所有REST对象一样，Service的定义可以基于POST方式，请求APIServer创建新的实例。例如，假定有一组Pod，它们暴露了9376端口，同时具有app=MyApp标签。此时可以定义Service如下：

kind: Service

apiVersion: v1

metadata:

name: my-service

spec:

selector:

app: MyApp

ports:

- protocol: TCP

port: 80

targetPort: 9376

上述配置创建一个名为my-service的Service对象，它会将请求代理到TCP端口为9376并且具有标签app=MyApp的Pod上。这个Service会被分配一个IP地址，通常称为ClusterIP，它会被服务的代理使用。

需要注意的是，Service能够将一个接收端口映射到任意的targetPort。默认情况下，targetPort将被设置为与Port字段相同的值。targetPort可以设置为一个字符串，引用backend Pod的一个端口的名称。

2.2.13.3 定义没有Selector的Service

Service抽象了该如何访问Kubernetes Pod，但也能够抽象其他类型的backend，例如：

* 希望在生产环境中访问外部的数据库集群。
* 希望Service指向另一个NameSpace中或其他集群中的服务。
* 正在将工作负载转移到Kubernetes集群，和运行在Kubernetes集群之外的backend。

在任何这些场景中，都能定义没有Selector的Service：

kind: Service

apiVersion: v1

metadata:

name: my-service

spec:

ports:

- protocol: TCP

port: 80

targetPort: 9376

由于这个Service没有Selector，就不会创建相关的Endpoints对象，可以手动将Service映射到指定的Endpoints：

kind: Endpoints

apiVersion: v1

metadata:

name: my-service

subsets:

- addresses:

- ip: 1.2.3.4

ports:

- port: 9376

注 意

Endpoint IP 地址不能是 loopback（127.0.0.0/8）、link-local（169.254.0.0/16）或者link-local 多播地址（224.0.0.0/24）。

访问没有Selector的Service与有Selector的Service的原理相同。请求将被路由到用户定义的Endpoint，该示例为1.2.3.4:9376。

ExternalName Service是Service的特例，它没有Selector，也没有定义任何端口和Endpoint，它通过返回该外部服务的别名来提供服务。

比如当查询主机my-service.prod.svc时，集群的DNS服务将返回一个值为my.database.example.com的CNAME记录：

kind: Service

apiVersion: v1

metadata:

name: my-service

namespace: prod

spec:

type: ExternalName

externalName: my.database.example.com

2.2.13.4 多端口Service

在许多情况下，Service可能需要暴露多个端口，对于这种情况Kubernetes支持Service定义多个端口，但使用多个端口时，必须提供所有端口的名称，例如：

kind: Service

apiVersion: v1

metadata:

name: my-service

spec:

selector:

app: MyApp

ports:

- name: http

protocol: TCP

port: 80

targetPort: 9376

- name: https

protocol: TCP

port: 443

targetPort: 9377

2.2.13.5 发布服务/服务类型

对于应用程序的某些部分（例如前端），一般要将服务公开到集群外部供用户访问。这种情况下都是用Ingress通过域名进行访问。

Kubernetes ServiceType（服务类型）主要包括以下几种：

* ClusterIP：在集群内部使用，默认值，只能从集群中访问。
* NodePort：在所有节点上打开一个端口，此端口可以代理至后端Pod，可以通过NodePort从集群外部访问集群内的服务，格式为NodeIP:NodePort。
* LoadBalancer：使用云提供商的负载均衡器公开服务，成本较高。
* ExternalName：通过返回定义的CNAME别名，没有设置任何类型的代理，需要1.7或更高版本kube-dns支持。

以NodePort为例。如果将type字段设置为NodePort，则Kubernetes将从--service-node-port-range参数指定的范围（默认为30000-32767）中自动分配端口，也可以手动指定NodePort，并且每个节点将代理该端口到Service。

一般格式如下：

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

type: NodePort

ports:

- port: 443

targetPort: 8443

nodePort: 30000

selector:

k8s-app: kubernetes-dashboard

常用的服务访问是NodePort和Ingress（关于Ingress参看2.2.14节），其他服务访问方式详情参看以下网址：

https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types