81Kubernetes 系列（七十四）揭开 Service 的神秘面纱：用于现代应用程序部署的可靠且可扩展的解决方案

介绍

在快速发展的技术世界中，组织不断努力提供无缝且可扩展的应用程序，容器化已成为游戏规则的改变者。Kubernetes 是一个开源容器编排平台，由于其能够自动部署、扩展和管理容器化应用程序，因此获得了极大的普及。Kubernetes 支持容器和外部服务之间无缝通信的关键组件之一是 Kubernetes Service。在这篇博文中，我们将深入研究 Kubernetes 服务的概念，并探索它的好处和用例。

什么是 Kubernetes Service?

在 Kubernetes 中，服务是一个抽象层，为一组相关的 Pod 提供一致的端点。它充当稳定的网络接口来公开容器，并允许它们在集群内相互通信或与外部服务通信。服务抽象化底层基础架构，并提供一种可靠的方法来访问容器，无论其位置或群集配置的更改如何。

Kubernetes 服务的类型

1. **ClusterIP：** 这是 Kubernetes 中默认的服务类型。它在只能在群集内访问的内部 IP 地址上公开服务。它允许同一集群内的 Pod 之间进行通信，同时使它们与外部流量隔离。

2. **NodePort：** 除了 ClusterIP 之外，NodePort 服务还会在群集的每个节点上的静态端口上公开服务。这种类型的服务允许外部流量通过访问指定端口上任何节点的 IP 地址来到达服务。

3. **LoadBalancer**：负载均衡服务通过在底层基础结构中预配负载均衡器来提供对服务的外部访问。它自动在多个节点之间分配传入流量，确保可扩展性和高可用性。

4. **ExternalName**：这种类型的服务将服务映射到外部 DNS 名称，允许群集内的 Pod 使用用户友好名称访问群集外部的服务。

Kubernetes 服务的好处：

1. **服务发现：** Kubernetes Service 提供内置的基于 DNS 的服务发现机制，允许容器使用服务名称而不是硬编码的 IP 地址来发现和相互通信。这会将应用程序逻辑与底层基础结构分离，使其对更改更具弹性。

2. **负载均衡：** 通过负载均衡服务类型，Kubernetes 会自动在多个 Pod 之间分配传入流量，确保最佳资源利用率和高可用性。此可伸缩性功能使应用程序能够无缝处理增加的流量。

3. **零停机部署：** Kubernetes 服务支持滚动更新，允许您在不中断持续操作的情况下部署应用程序的新版本。通过逐步将流量从旧 Pod 重定向到新 Pod，Service 可确保在部署过程中实现零停机时间。

4. **扩展和弹性：** Kubernetes 服务在水平扩展中起着至关重要的作用。当与服务关联的 Pod 数量增加或减少时，服务会自动调整其配置，从而确保适当的负载分布和弹性。

Kubernetes 服务的用例：

1. **微服务架构：** Kubernetes Service 是部署基于微服务的应用程序的理想选择。每个微服务都可以作为服务公开，从而实现服务间通信和负载均衡。

2. **Web 应用程序 ：** 借助 NodePort 和 LoadBalancer 服务，Kubernetes 可以轻松地将 Web 应用程序暴露给外部流量。这允许从互联网无缝访问应用程序，而无需担心管理复杂的网络配置。

3. **后端服务：** 服务通常用于向应用程序公开后端服务，例如数据库、缓存系统或消息代理。通过抽象底层基础设施细节，Kubernetes Service 确保可靠、安全地访问这些基本组件。

案例

让我们考虑一个例子来说明 Kubernetes Service 的用法。

假设您正在开发一个基于微服务的电子商务应用程序，该应用程序由多种服务（如产品服务、用户服务和支付服务）组成。每个服务都部署为 Kubernetes 集群中的单独容器。要启用这些服务之间的通信，您可以使用 Kubernetes 服务。

1. 创建服务

首先，为每个微服务定义一个服务。让我们以产品服务为例。您将创建一个 YAML 文件，假设为 product-service.yaml ，其中包含以下内容：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: product-service
spec:
  selector:
    app: product-service
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

此 YAML 文件描述名为 product-service 的 ClusterIP 服务，该服务选择标签为 `app: product-service` 的 Pod。服务在内部公开端口 8080，这是产品服务正在侦听的端口。

2. 部署 Pod

接下来，您将产品服务部署为 Kubernetes 集群中的 Pod。您将定义另一个 YAML 文件，我们将其称为 product-service-deployment.yaml ，该文件指定产品服务的部署详细信息：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: product-service
spec:
  replicas: 3
  selector:
    matchLabels:
      app: product-service
  template:
    metadata:
      labels:
        app: product-service
    spec:
      containers:
        - name: product-service
          image: your-product-service-image:latest
          ports:
            - containerPort: 8080
```

在此 YAML 文件中，您可以为产品服务定义部署，并指定需要容器的三个副本。Pod 模板包含一个容器定义，该定义使用指定的镜像并公开端口 8080。

3. 服务发现和通信

创建服务和部署后，Kubernetes 负责管理 Pod，并确保运行所需数量的副本。现在，可以通过服务的 DNS 名称在群集内访问产品服务 Pod。

集群中的其他微服务可以使用此 DNS 名称与产品服务通信。例如，用户服务可以向 **http://product-service:8080/api/products** 发出 HTTP 请求以获取产品信息。

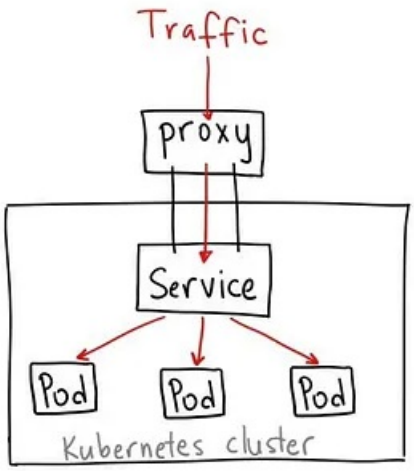
4. 对外公开服务

如果要向外部流量公开产品服务，可以将服务类型更改为 `LoadBalancer` 或 `NodePort` 。例如，如果选择负载均衡器类型，服务将在底层基础结构中预配负载均衡器，并分配外部 IP 地址。这允许外部客户端使用负载均衡器的 IP 地址和指定端口访问产品服务。

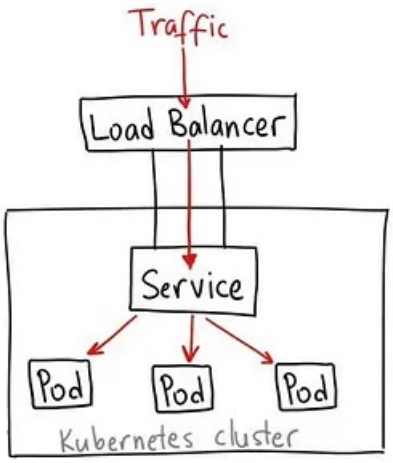
总结

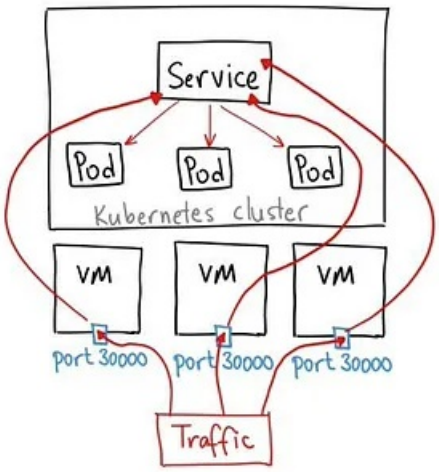
Kubernetes Service 是在容器化环境中部署可扩展且可靠的应用程序的基本构建块。通过抽象网络的复杂性并为 Pod 提供一致的端点，服务可实现无缝通信和负载平衡。无论您是部署微服务、Web 应用程序还是后端服务，Kubernetes 服务都使您能够构建强大且有弹性的架构。采用 Kubernetes 服务可释放容器编排的真正潜力，并在当今快节奏的数字环境中促进高效的应用程序交付。

ClusterIP

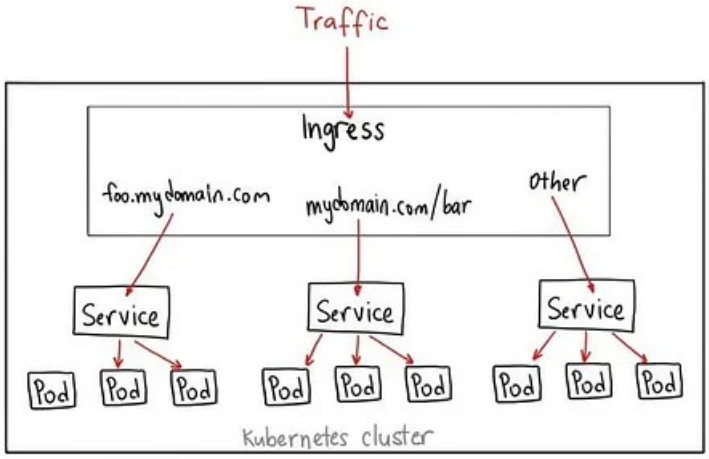


LoadBalancer





NodePort



Ingress