



K8s Service與Virtual Service



詳細說明:k8s的vitrtul service、service



在 Kubernetes (K8s) 中，**Virtual Service** 和 **Service** 是兩個不同的概念，分別來自 **Istio** 和 **Kubernetes 原生**，主要用於服務的網絡流量管理。以下是詳細的說明：

1. Kubernetes Service

概述

Kubernetes 的 **Service** 是一種內建的 Kubernetes 資源，用來定義一組 **Pod** 的訪問規則。它解決了 **Pod 動態變更 IP** 的問題，使外部或內部的流量可以穩定地訪問後端 Pod。

類型

1. ClusterIP（預設）：

- 只能在 Kubernetes 內部存取該 Service，無法從外部存取。
- 使用內部 DNS 解析 `my-service.default.svc.cluster.local`。

2. NodePort：

- 在每個節點上開放一個固定端口（範圍 30000-32767）。
- 可以透過 `<NodeIP>:<NodePort>` 存取服務。

3. LoadBalancer：

- 用於雲端環境（如 AWS, GCP, Azure），會自動創建一個負載均衡器。





- 能夠暴露服務到外部，適用於需要外部流量的應用。

4. ExternalName :

- 將 Kubernetes 內部 DNS 解析到外部的 DNS（如 `google.com`）。
- 適用於將內部應用指向外部服務。

示例

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app # 綁定的 Pod
  ports:
    - protocol: TCP
      port: 80 # Service 內部 Port
      targetPort: 8080 # Pod 上的 Port
  type: ClusterIP
```

流量處理方式

1. 內部流量：透過 **ClusterIP** 進行 Pod 負載均衡。
2. 外部流量：透過 **NodePort** 或 **LoadBalancer** 來進入 Kubernetes 集群。

2. Istio Virtual Service

概述

Virtual Service 是 **Istio** 服務網格中的流量管理資源，用於控制流量的路由行為。它可以基於請求的內容來決定流量應該流向哪個 Kubernetes Service 或 Pod，例如：



- 藍綠部署
- A/B 測試
- 金絲雀部署
- 超時與重試策略

功能

1. 流量控制：
 - 基於 **HTTP 路徑、標頭、權重分流** 來決定流量的去向。
2. 重試與超時：
 - 設定請求的 **重試次數** 和 **超時時間**。
3. 流量鏡像：
 - **複製** 請求到另一個服務（如新版本的測試環境）。
4. **Fault Injection**（錯誤注入）：
 - 模擬延遲或錯誤來測試應用的容錯能力。

示例

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: my-virtual-service
spec:
  hosts:
    - my-service.default.svc.cluster.local # 對應 Kubernetes Service
  http:
    - match:
        - headers:
            user-agent:
              regex: ".*Mobile.*" # 針對 Mobile 用戶
```



```
route:
  - destination:
      host: my-service.default.svc.cluster.local
      subset: v2 # 流量導向 v2 版本的 Pod
  - route:
      - destination:
          host: my-service.default.svc.cluster.local
          subset: v1 # 預設流量導向 v1 版本的 Pod
```

流量處理方式

1. Virtual Service **不會** 直接處理 Pod，而是透過 **Kubernetes Service** 來轉發流量。
2. 透過 `destination.subset` 來選擇流量應該發送到哪個 **Pod 版本 (v1 或 v2)**。
3. 用於服務網格內的高級流量控制，例如 Canary 部署、A/B 測試等。

3. Virtual Service vs Service 的差異

比較項目	Kubernetes Service	Istio Virtual Service
用途	提供 Pod 負載均衡 ，統一服務訪問方式	控制 進入 Kubernetes Service 的流量
類型	ClusterIP、NodePort、LoadBalancer、ExternalName	HTTP、TCP、gRPC 流量控制
流量分流	只能基於 Pod selector	可基於 標頭、請求路徑、User-Agent 進行分流
重試與超時	無內建支持	可自定義重試機制、超時時間
錯誤注入	無	可模擬請求延遲或錯誤
A/B 測試	需自行調整 Deployment	直接透過權重設定流量比例
金絲雀部署	需手動修改 Replica 數量	直接設定比例（如 v1:80%, v2:20%）



4. 如何搭配使用

在 **Kubernetes + Istio** 環境下，Service 和 Virtual Service 通常是搭配使用的：

1. **Service** 提供 **Pod 負載均衡** 和 **內部服務解析**。
2. **Virtual Service** 透過 **更細緻的路由規則** 來管理請求，例如：
 - 將 **90%** 的流量導向 **v1** 版本，**10%** 給 **v2**。
 - 針對不同 **User-Agent** 設定不同的後端版本。
 - 在流量高峰時對部分請求進行限流或錯誤注入。

示例

1. 定義 Kubernetes Service

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - port: 80
      targetPort: 8080
  type: ClusterIP
```

2. 定義 Virtual Service 來控制流量

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: my-virtual-service
spec:
```



```
hosts:
- my-service.default.svc.cluster.local
http:
- route:
  - destination:
    host: my-service.default.svc.cluster.local
    subset: v1
    weight: 80
  - destination:
    host: my-service.default.svc.cluster.local
    subset: v2
    weight: 20
```

5. 總結

1. Kubernetes Service

- 用於 **提供 Pod 的統一訪問點**，負責負載均衡。
- 適用於內部流量（ClusterIP）或外部流量（NodePort, LoadBalancer）。

2. Istio Virtual Service

- **進階的流量控制**，適用於微服務架構。
- 可以基於 **請求的 header、路徑、流量權重** 來決定流量的去向。
- 適合 **A/B 測試、金絲雀部署、超時與重試策略**。

若需要 **基本的負載均衡**，只需 Kubernetes Service；若需要 **更細緻的流量控制**，則需搭配 Istio Virtual Service。

這樣的組合可讓 Kubernetes 集群內部的流量管理更靈活！🚀





```
詳細說明:spec:
  selector:
    app: my-app # 綁定的 Pod
  ports:
    - protocol: TCP
      port: 80 # Service 內部 Port
      targetPort: 8080 # Pod 上的 Port
  type: ClusterIP

、
spec:
  hosts:
    - my-service.default.svc.cluster.local # 對應
      Kubernetes Service
  http:
    - match:
        - headers:
            user-agent:
              regex: ".*Mobile.*" # 針對 Mobile 用戶
      route:
        - destination:
            host: my-service.default.svc.cluster.local
            subset: v2 # 流量導向 v2 版本的 Pod
        - route:
            - destination:
                host: my-service.default.svc.cluster.local
                subset: v1 # 預設流量導向 v1 版本的 Pod
```



解析 Kubernetes Service 和 Istio VirtualService 配置

這兩段 YAML 設定分別來自 **Kubernetes Service** 和 **Istio VirtualService**，我們將詳細解析它們的結構、用途與運作方式。



1. Kubernetes Service

```
spec:
  selector:
    app: my-app # 綁定的 Pod
  ports:
    - protocol: TCP
      port: 80 # Service 內部 Port
      targetPort: 8080 # Pod 上的 Port
  type: ClusterIP
```

(1) `selector`：綁定 Service 與 Pod

```
selector:
  app: my-app # 綁定的 Pod
```

- `selector` 是 **Service** 如何識別要管理的 **Pod**。
- 這裡 `app: my-app` 表示 **Kubernetes** 會尋找具有 `app=my-app` 標籤的 **Pod**，並將它們加入此 Service 的負載均衡範圍。

(2) `ports`：服務端口設定

```
ports:
  - protocol: TCP
    port: 80 # Service 內部 Port
    targetPort: 8080 # Pod 上的 Port
```

- `protocol: TCP`：指定使用 **TCP** 協議（預設值）。
- `port: 80`：
 - 這是 **Service** 對外開放的 **Port**。
 - 任何訪問這個 Service 的流量都要透過這個 Port 進入。
- `targetPort: 8080`：



- 這是 **Pod** 內部實際運行應用的端口。
- 當流量進入 Service 的 **port:80**，會被轉發到目標 Pod 的 **8080**。

示意圖

```
Client -----> Service (port 80) -----> Pod (targetPort 8080)
```

這樣 Kubernetes Service 就能作為 **流量中介**，將請求導向正確的 Pod。

(3) `type: ClusterIP`

```
type: ClusterIP
```

- `ClusterIP` 是 Kubernetes Service 的預設類型：
 - 僅限 **Kubernetes** 內部訪問，不能直接被外部流量存取。
 - 主要用於集群內部的 **服務間通信**。

示例

```
curl http://my-service.default.svc.cluster.local:80 # 內部存取
```

如果要讓外部存取，需要使用 **NodePort** 或 **LoadBalancer**。

2. Istio VirtualService

```
spec:
  hosts:
    - my-service.default.svc.cluster.local # 對應 Kubernetes Service
  http:
    - match:
```



```
- headers:
  user-agent:
    regex: ".*Mobile.*" # 針對 Mobile 用戶
route:
- destination:
  host: my-service.default.svc.cluster.local
  subset: v2 # 流量導向 v2 版本的 Pod
- route:
  destination:
    host: my-service.default.svc.cluster.local
    subset: v1 # 預設流量導向 v1 版本的 Pod
```

這是一個 **Istio VirtualService** 配置，主要作用是 **控制請求流向 Kubernetes Service**，並根據條件（如 ``User-Agent``）進行流量分配。

(1) ``hosts``：目標 **Kubernetes Service**

```
hosts:
- my-service.default.svc.cluster.local # 對應 Kubernetes Service
```

- ``hosts`` 定義 **VirtualService** 作用的目標服務。
- 這裡的 ``my-service.default.svc.cluster.local`` 是 Kubernetes Service 的 DNS 名稱，表示 **VirtualService** 會攔截發送到這個 **Service** 的流量。

(2) ``http.match``：匹配條件

```
- match:
  - headers:
    user-agent:
      regex: ".*Mobile.*" # 針對 Mobile 用戶
```



- ``match`` 定義 流量的匹配條件，這裡根據 **HTTP** 標頭的 ``User-Agent`` 來決定流向：
 - 如果請求的 ``User-Agent`` 包含 ``Mobile``，則流量將導向 ``v2`` 版本。
 - 這可用於 行動端用戶導向新版服務，而桌面端仍使用 ``v1``。

(3) ``http.route``：流量路由

```
route:
  - destination:
      host: my-service.default.svc.cluster.local
      subset: v2 # 流量導向 v2 版本的 Pod
```

- 當匹配 ``Mobile`` 用戶時，流量會導向 ``subset: v2``。
- ``subset: v2`` 是 **DestinationRule** 定義的一個 Pod 群組（v2 版本）。

```
- route:
  - destination:
      host: my-service.default.svc.cluster.local
      subset: v1 # 預設流量導向 v1 版本的 Pod
```

- 未匹配條件的流量，預設導向 ``subset: v1``（舊版本 Pod）。

示意圖

```
Mobile 用戶  --->  my-service (VirtualService)  ---> Pod v2 (新版本)
桌面用戶    --->  my-service (VirtualService)  ---> Pod v1 (舊版本)
```

3. 總結



設定項目	Kubernetes Service	Istio VirtualService
目標	負責 Pod 負載均衡	負責 流量路由與控制
識別 Pod	使用 <code>`selector: app=my-app`</code>	不直接識別 Pod，而是透過 <code>`destination.subset`</code>
流量來源	內部 Pod、NodePort 或 LoadBalancer	內部 Service（需經 Istio Sidecar）
流量處理方式	透過 <code>`port → targetPort`</code> 負載均衡	透過 <code>`match`</code> 條件決定流向
應用場景	服務發現、負載均衡	Canary 部署、A/B 測試、金絲雀發布

何時使用？

- 只需要基本的負載均衡 → 只用 **Kubernetes Service**。
- 需要控制流量的分配方式（如 A/B 測試）→ 搭配 **Istio VirtualService**。

這樣的組合可以讓 **Kubernetes** 內部流量管理更靈活，適用於微服務架構！