Server-side Service Discovery and Load Balancing using Kubernetes

1. Client-side Service Discovery và Load Balancing (với Eureka)

- Các microservice khi khởi động sẽ **đăng ký và gửi heartbeat** định kỳ tới **Eureka Server**.
- Khi một microservice (ví dụ: accounts-service) cần gọi đến một service khác (ví dụ: loans-service), nó sẽ **truy vấn Eureka** để lấy danh sách các instance đang hoạt động.
- Sau đó, **load balancing** sẽ được thực hiện **ở phía client**, nhờ Spring Cloud LoadBalancer, để chọn ra một instance phù hợp để gửi request.
- **Ưu điểm**: client có toàn quyền kiểm soát chiến lược cân bằng tải.
- **Nhược điểm**: developer phải tự quản lý Eureka Server, cấu hình thủ công ở tất cả các service.

2. Server-side Service Discovery và Load Balancing (với Kubernetes)

- Trong Kubernetes, không cần các service **tự đăng ký hoặc gửi heartbeat** như với Eureka.
- **Kubernetes Discovery Server** sẽ tự động sử dụng API của Kubernetes để truy vấn thông tin về các service đang chạy.
- Các client microservice (ví dụ: accounts-service) chỉ cần gửi request thông qua **DNS name của service**, và Kubernetes sẽ tự động **phân phối request** đến một instance phù hợp (load balancing được thực hiện ở phía server).
- **Ưu điểm**:
    - Không cần viết thêm code đăng ký/dỡ đăng ký service.

    - Không cần duy trì hoặc cấu hình Eureka Server.
    - Tối giản cấu hình và giảm tải công việc cho developer.
- **Nhược điểm**:
    - Client không có quyền kiểm soát chiến lược load balancing — toàn bộ được quyết định bởi Kubernetes.

Discovery Server trong Kubernetes cluster bằng cách sử dụng Spring Cloud Kubernetes

 Sử dụng Spring Cloud Kubernetes

- Dự án này hỗ trợ các tính năng như service discovery, load balancing mà không cần triển khai riêng Eureka.

 Blog chính thức từ Spring Cloud Kubernetes (năm 2021)

- Cung cấp một **Kubernetes manifest file** để tạo Discovery Server.
- Nhưng phải **chỉnh sửa lại** file để tương thích với các phiên bản mới.

 Tạo folder và file cần thiết

- Folder: section_17/Kubernetes
- File manifest: kubernetes-discoveryserver.yaml

 Nội dung file manifest

- kind: List → chứa nhiều đối tượng Kubernetes.
- **Service**: để expose cổng 80 (cho microservices bên trong cluster) → ánh xạ tới cổng 8761 (ứng dụng chạy).

    - Loại service: ClusterIP → không gây xung đột với các dịch vụ như Keycloak dùng LoadBalancer.
- **ServiceAccount** + **Role** + **RoleBinding**
    - Gán quyền namespace-reader cho service account → có thể đọc services, endpoints, và cần thêm pods (nếu không sẽ lỗi).
- **Deployment**:
    - Dùng image chính thức từ Spring Cloud Kubernetes để chạy một **Discovery Server**.
    - Không cần viết lại ứng dụng Discovery server từ đầu như Eureka.
    - Image cũ: 2.1.0-M3 → nên tìm bản ổn định hơn trên DockerHub.

Lý do không dùng Helm Chart:

1. **Discovery Server chỉ cần cài một lần**, không cần cập nhật thường xuyên.
2. **Không có Helm Chart chính thức** từ cộng đồng (như Bitnami) cho Discovery Server.

Các bước triển khai Discovery Server:

1. **Chạy lệnh sau trong terminal:**

kubectl apply -f kubernetes-discoveryserver.yaml

→ File này sẽ tạo: Service, ServiceAccount, Role, RoleBinding, và Deployment.

Kiểm tra trên Kubernetes Dashboard:

- Truy cập vào **Pods**
- Mở **Logs** của Discovery Server → Xác nhận đã khởi động thành công (Spring Boot log).

Chuyển từ Eureka sang Kubernetes Discovery Client trong Microservices

Cấu hình các Service (Accounts, Loans, Cards)

1. Thay thế Dependency Eureka

Trong file pom.xml của mỗi microservice:

- **Xóa**:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

- Thêm:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-kubernetes-discoveryclient</artifactId>
</dependency>
```

2. Kích hoạt Discovery Client

Trong Application class của mỗi service (vd: AccountsApplication.java), thêm:

```
@EnableDiscoveryClient
```

3. Cập nhật cấu hình application.yml

- **Xóa** các cấu hình liên quan đến Eureka:

```
eureka:
```

```
  client:
    register-with-eureka: true
    fetch-registry: true
  instance:
    prefer-ip-address: true
```

- **Thêm** cấu hình Kubernetes Discovery:

```
spring:
  cloud:
    kubernetes:
      discovery:
        enabled: true
        all-namespaces: true
```

4. Tạo Docker Image mới

Build lại Docker images cho từng service với tag mới:

```
docker build -t easybank/accounts:s17 .
docker build -t easybank/loans:s17 .
docker build -t easybank/cards:s17 .
```

Cập nhật Gateway Server

1. Cập nhật pom.xml

- **Xóa**:

<dependency>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

- Thêm:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-kubernetes-discoveryclient</artifactId>

</dependency>

2. Kích hoạt Discovery Client

Trong GatewayserverApplication.java:

@EnableDiscoveryClient

3. Cập nhật cấu hình route

Thay thế lb://SERVICE-NAME bằng tên service trong cluster:

routes:

- id: accounts

uri: http://accounts:8080

predicates:

*- Path=/easybank/accounts/***

- id: loans

uri: http://loans:8090

predicates:

*- Path=/easybank/loans/***

- id: cards

uri: http://cards:9000

predicates:

*- Path=/easybank/cards/***

4. Chỉnh sửa application.yml

- **Xóa:**

```
spring:
  cloud:
    discovery:
      locator:
        enabled: true
        lower-case-service-id: true
```

- **Thêm:**

```
spring:
  cloud:
    kubernetes:
      discovery:
        enabled: true
```

```
    all-namespaces: true
  discovery:
    client:
      health-indicator:
        enabled: false
```

Cập nhật Helm Chart cho các thay đổi của Discovery Server trong Kubernetes

1. Cập nhật Docker Images

Tôi đã build lại Docker images cho toàn bộ các microservice và gán tag mới là s17. Việc này đảm bảo rằng các image được triển khai lên cluster sẽ bao gồm toàn bộ những thay đổi mới trong chương học hiện tại.

2. Gỡ bỏ Eureka Server khỏi Helm Chart

Trong thư mục helm/easybank-services, tôi đã xóa thư mục liên quan đến eureka-server vì hiện tại chúng ta sử dụng cơ chế Service Discovery dựa trên Kubernetes, không còn phụ thuộc vào Eureka nữa. Đồng thời, trong các tệp Chart.yaml thuộc các environment (dev, qa, prod), tôi cũng xóa khai báo dependency của eureka-server.

3. Cập nhật values.yaml của từng microservice

Tôi đã cập nhật tag Docker image từ s14 thành s17 cho toàn bộ các microservice. Ngoài ra, với accounts-service, tôi cấu hình replicaCount = 2 để kiểm thử khả năng load balancing phía server.

4. Biên dịch lại Helm Chart

Do đã thay đổi dependencies, tôi xóa các tệp Chart.lock trong thư mục dev, qa, và prod để tránh lỗi. Sau đó, tôi lần lượt chạy lệnh helm dependency build trong từng environment để biên dịch lại Helm chart.

5. Cài đặt lại các microservice

Sau khi biên dịch thành công, tôi tiến hành cài đặt lại các microservice bằng lệnh:

helm install easybank ./prod

Tôi xác nhận quá trình triển khai qua Kubernetes Dashboard. Đặc biệt, accounts-service có 2 Pod như kỳ vọng nhờ cấu hình replica.

6. Phát hiện lỗi và khắc phục

Mặc dù các pod được tạo ra, nhưng một số service như gateway, loans, cards và accounts không khởi động được. Sau khi kiểm tra log, tôi phát hiện nguyên nhân là do thiếu cấu hình spring.cloud.kubernetes.discovery.discovery-server-url – một property quan trọng để các service có thể thực hiện Service Discovery với Discovery Server.

7. Khắc phục cấu hình thiếu

Tôi đã mở Helm chart chung easybank-common, chỉnh sửa tệp configmap.yaml, và thêm biến môi trường SPRING_CLOUD_KUBERNETES_DISCOVERY_DISCOVERY_SERVER_URL với giá trị lấy từ biến Helm discoveryServerURL. Sau đó, tôi chạy lệnh helm uninstall easybank để gỡ release cũ và sẵn sàng triển khai lại hệ thống với cấu hình chính xác.

DEMO SERVER-SIDE SERVICE DISCOVERY VÀ LOAD BALANCING TRONG KUBERNETES

application.yml trong gateway-service:

```
spring:
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
          lowerCaseServiceId: true
      routes:
        - id: accounts
          uri: lb://accounts
          predicates:
            - Path=/easybank/accounts/api/**
```

pom.xml hoặc build.gradle – Các dependencies cần thiết:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>


<dependency>
```

```
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-fabric8</artifactId>
</dependency>
```

Kubernetes Deployment (2 replicas):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: accounts
spec:
  replicas: 2
  selector:
    matchLabels:
      app: accounts
  template:
    metadata:
      labels:
        app: accounts
    spec:
      containers:
        - name: accounts
```

```
        image: your-dockerhub/accounts:latest
        ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: accounts
spec:
  selector:
    app: accounts
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```