QUẢN LÝ CẤU HÌNH TRONG MICROSERVICES

1. Giới thiệu

Trong kiến trúc Microservices, mỗi dịch vụ thường có cấu hình riêng, bao gồm thông tin kết nối cơ sở dữ liệu, thông tin xác thực, tham số môi trường, v.v. Việc quản lý các cấu hình này một cách hiệu quả là một thách thức quan trọng nhằm đảm bảo hệ thống hoạt động ổn định và có thể dễ dàng điều chỉnh khi cần thiết.

Báo cáo này sẽ tập trung vào phương pháp Quản lý Cấu hình trong Microservices, đặc biệt là cách sử dụng **Spring Cloud Config Server** để quản lý cấu hình tập trung, dựa trên nội dung từ khóa học "Master Microservices with Spring, Docker, Kubernetes" trên Udemy.

2. Tầm quan trọng của Quản lý Cấu hình

Trong hệ thống Microservices, cấu hình có thể thay đổi tùy theo môi trường triển khai (dev, staging, production). Nếu không có phương pháp quản lý cấu hình hiệu quả, sẽ xảy ra các vấn đề như:

- **Mất đồng nhất cấu hình** giữa các môi trường.
- **Khó kiểm soát thay đổi cấu hình** trong hệ thống lớn.
- **Nguy cơ bảo mật** do lưu trữ thông tin nhạy cảm trực tiếp trong mã nguồn.
- **Khó bảo trì** khi phải cập nhật từng dịch vụ riêng lẻ.

Một giải pháp quản lý cấu hình tập trung giúp giải quyết các vấn đề trên, đồng thời cải thiện hiệu suất và độ tin cậy của hệ thống.

3. Giới thiệu Spring Cloud Config Server

Spring Cloud Config Server là một công cụ trong hệ sinh thái Spring Cloud giúp quản lý cấu hình tập trung cho các microservices. Các dịch vụ có thể truy xuất và cập nhật cấu hình một cách linh hoạt mà không cần tái khởi động hoặc triển khai lại ứng dụng.

3.1. Kiến trúc Spring Cloud Config Server

Spring Cloud Config Server hoạt động theo mô hình **client-server**, trong đó:

- **Config Server**: Chứa tất cả các tệp cấu hình và cung cấp API để các microservices lấy thông tin cấu hình.
- **Config Clients**: Các microservices truy xuất dữ liệu từ Config Server để sử dụng trong ứng dụng.
- **Backend Storage**: Config Server có thể lấy dữ liệu từ nhiều nguồn như Git, cơ sở dữ liệu hoặc bộ nhớ cục bộ.

3.2. Lợi ích của Spring Cloud Config Server

- **Quản lý cấu hình tập trung**: Giúp kiểm soát và duy trì cấu hình của tất cả microservices một cách dễ dàng.
- **Tách biệt cấu hình khỏi mã nguồn**: Hạn chế lưu thông tin nhạy cảm trong mã nguồn, tăng cường bảo mật.
- **Dễ dàng cập nhật và áp dụng thay đổi**: Cho phép cập nhật cấu hình mà không cần triển khai lại ứng dụng.
- **Hỗ trợ nhiều môi trường**: Cấu hình có thể được tùy chỉnh cho từng môi trường triển khai khác nhau.

4. Cách triển khai Spring Cloud Config Server

4.1. Cài đặt Config Server

Để thiết lập một Config Server, ta cần tạo một ứng dụng Spring Boot và khai báo dependencies trong pom.xml:

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```

Sau đó, thêm annotation @EnableConfigServer vào lớp MainApplication.java:

```
@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(ConfigServerApplication.class, args);
    }
}
```

Cấu hình application.yml cho Config Server để lấy dữ liệu từ Git:

```
server:
  port: 8888
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/example/config-repo
```

4.2. Cấu hình Client để sử dụng Config Server

Trong các microservices client, ta cần khai báo dependencies tương ứng:

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

Sau đó, cấu hình bootstrap.yml để kết nối đến Config Server:

```
spring:
  application:
    name: orderservice
  cloud:
```

```
  config:
    uri: http://localhost:8888
```

Client có thể lấy thông tin cấu hình thông qua HTTP API:

```
curl http://localhost:8888/orderservice/default
```

5. Quản lý bảo mật trong Config Server

Một vấn đề quan trọng khi sử dụng Config Server là bảo mật thông tin nhạy cảm như mật khẩu database, API keys, v.v. Các phương pháp bảo vệ thông tin cấu hình bao gồm:

- **Mã hóa dữ liệu**: Spring Cloud Config hỗ trợ mã hóa thông tin bằng symmetric/asymmetric key.
- **Sử dụng Vault**: Có thể tích hợp HashiCorp Vault để quản lý bí mật một cách an toàn.
- **Hạn chế quyền truy cập**: Chỉ cho phép các microservices được ủy quyền mới có thể lấy thông tin cấu hình.

Ví dụ mã hóa dữ liệu với Spring Cloud Config:

```
curl -X POST http://localhost:8888/encrypt -d 'mypassword'
```

6. Xử lý thay đổi cấu hình động

Spring Cloud Config hỗ trợ cập nhật cấu hình mà không cần khởi động lại ứng dụng bằng cách kết hợp với **Spring Cloud Bus**.

1. Thêm dependency spring-cloud-starter-bus-amqp vào dự án.
2. Cấu hình RabbitMQ làm message broker.
3. Khi cập nhật cấu hình, gửi sự kiện thông báo đến các microservices:

```
curl -X POST http://localhost:8888/actuator/bus-refresh
```

4. Các microservices tự động nhận cấu hình mới và cập nhật mà không cần khởi động lại.