Service Discovery and Eureka Configuration

Table of Contents

1. Introduction to Service Discovery

Service Discovery is a crucial component in microservices architecture. It's the process of automatically detecting devices and services on a network. In the context of microservices, it helps services find and communicate with each other without hard-coding hostname and port information.

2. Key Concepts in Service Discovery

- **Registry**: A database of available services, their instances, and their locations.
- **Service Registration**: The process by which a service registers itself with the registry.
- **Service Discovery**: The process of finding the location of a service instance.
- **Health Checking**: Monitoring the health and availability of service instances.

3. Introduction to Eureka

Eureka is Netflix's service discovery server and client. It allows services to find and communicate with each other without hard-coding the hostname and port. Each Eureka client can simultaneously act as a server, to replicate its status to a configured peer.

4. Eureka Architecture

Eureka follows a client-server architecture:

- **Eureka Server**: Holds the registry of services.
- **Eureka Client**: Registers with Eureka Server and periodically sends heartbeats.

5. Configuring Eureka Server

To set up a Eureka Server:

1. Add the Eureka Server dependency to your project.
2. Annotate your main class with `@EnableEurekaServer`.
3. Configure the application properties:

```
server:
  port: 8761

eureka:
  client:
    registerWithEureka: false
    fetchRegistry: false

spring:
  application:
    name: eureka-server
```

6. Configuring Eureka Clients

To configure a service as a Eureka Client:

1. Add the Eureka Client dependency.
2. Annotate your main class with `@EnableDiscoveryClient`.
3. Configure the application properties:

```
spring:
  application:
    name: my-service

eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
```

7. Best Practices and Considerations

- Use meaningful names for your services.
- Implement proper health check endpoints.
- Consider security for your Eureka server.
- Plan for high availability by running multiple Eureka servers.
- Regularly monitor the Eureka dashboard.

8. Troubleshooting Common Issues

- **Service not registering**: Check network connectivity and Eureka client configuration.
- **Service shows as DOWN**: Verify the health check endpoint.
- **High CPU usage**: Ensure proper configuration of heartbeat intervals.
- **Eureka server not starting**: Check for port conflicts.