SERVICE DISCOVERY USING SPRING CLOUD NETFLIX EUREKA

Service discovery is the process of locating and identifying services within a network or system. In the context of microservices, it involves the ability of services to find and communicate with each other. Services are often distributed across multiple nodes or containers, and they can be deployed, scaled, or terminated dynamically. This dynamic nature makes it challenging for services to know the network locations (IP addresses and ports) of other services.

Service discovery involves services registering themselves with a central service registry or a discovery server when they start up. This registration typically includes information such as the service name, IP address, and port number, When a service needs to communicate with another service, it queries the service registry to look up the network location of the desired service. The service registry then provides the necessary information for the requesting service to establish a connection.

PREREQUISITES

Java 17, Maven, Spring Boot Application

EUREKA SERVER

1. Create a new Maven module called **"discovery-server"** and add the following dependencies inside pom.xml:

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

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-parent</artifactId>
            <version>2023.0.0</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

2. Inside our discover-server module, create the main application class:

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

```
    }
}
```

Do not forget to enable the Eureka Server in a @SpringBootApplication by annotating it with @EnableEurekaServer!

3. Create a new file called *"application.properties"* inside the resources folder of our discovery-server module and add the following configurations:

```
eureka.instance.hostname=localhost
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
```

5. Run the SpringBootApplication

EUREKA CLIENT

1. Add the following dependencies inside the pom.xml of your client:

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

2. Inside the application.properties file of your client add the following Eureka Client configurations:

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

3. Provide a name for your client application:

```
spring.application.name=client-service
```

3. You can also let Spring Boot choose a random port for your client by setting:

`server.port=0` (This will also allow registering multiple instances of the client)

4. Run the SpringBootApplication of the client

VALIDATION

1. Notice inside the discovery-server logs that your client has been discovered:

```
c.n.e.registry.AbstractInstanceRegistry  : Registered instance
CLIENT-SERVICE/192.168.1.145:client-service:0 with status UP
(replication=false)
```

2. Visit the Eureka dashboard at http://localhost:8761 and see the instance of your client registered inside the Applications tab

3. Now, instead of making calls to your client by using “http://localhost:8080/api” (supposing the client is running on port 8080), you can use “http://client-service/api”, which is more dynamic and maintainable