MICROSERVICES

To implement a **microservices architecture** in a project, we'll go step-by-step to create an application involving multiple microservices, including **Eureka Service Registry**, **API Gateway**, and individual services communicating with each other. Below is the complete structure and implementation guide:

Project Overview

Scenario

We'll create a simple **E-commerce platform** with the following services:

1. **Eureka Server**: Service registry.
2. **API Gateway**: Central entry point for all requests.
3. **Product Service**: Handles product data.
4. **Order Service**: Handles orders and communicates with Product Service.
5. **Config Server (Optional)**: Centralized configuration for microservices.
6. **Service Communication**: RestTemplate and Feign for inter-service communication.
7. **Resilience**: Circuit Breakers (using Resilience4j).

1. Create the Eureka Server

Dependencies

Add the following dependencies in the **Eureka Server**'s `pom.xml`:

xml

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

Main Class

Enable Eureka Server with `@EnableEurekaServer`:

java

```
package com.example.eurekaserver;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

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

application.yml

Set up Eureka configuration:

```
yaml

server:
  port: 8761

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
  server:
    enable-self-preservation: false
```

2. Create the API Gateway

Dependencies

Add the following dependencies for Spring Cloud Gateway in the **API Gateway's** `pom.xml`:

```
xml

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

Main Class

Enable Eureka Client:

```
java

package com.example.apigateway;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

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

application.yml

Configure routes for the gateway:

```
yaml
CopyEdit
server:
  port: 8080

spring:
  application:
    name: api-gateway
  cloud:
    gateway:
      routes:
        - id: product-service
          uri: lb://PRODUCT-SERVICE
          predicates:
            - Path=/product/**
        - id: order-service
          uri: lb://ORDER-SERVICE
          predicates:
            - Path=/order/**
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

---

3. Create the Product Service

Dependencies

Add Eureka Client dependency and web dependencies:

```
xml

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

Main Class

Enable Eureka Client:

```
java

package com.example.productservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

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

application.yml

Register Product Service with Eureka:

```
yaml

server:
  port: 8081

spring:
  application:
    name: product-service

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

Controller

Create an endpoint to return product details:

```
java

@RestController
@RequestMapping("/product")
public class ProductController {

    @GetMapping("/{id}")
    public String getProduct(@PathVariable String id) {
        return "Product details for product id: " + id;
    }
}
```

---

4. Create the Order Service

Dependencies

Add dependencies for Eureka, web, and Feign:

```
xml

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

Main Class

Enable Eureka Client and Feign:

```
java
```

```
package com.example.orderservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

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

application.yml

Register Order Service with Eureka:

```
yaml
CopyEdit
server:
  port: 8082

spring:
  application:
    name: order-service

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

Feign Client

Create a Feign client to call Product Service:

```
java

@FeignClient(name = "product-service")
public interface ProductClient {

    @GetMapping("/product/{id}")
    String getProduct(@PathVariable String id);
}
```

Controller

Use Feign to fetch product data for an order:

```
java
CopyEdit
@RestController
@RequestMapping("/order")
public class OrderController {

    private final ProductClient productClient;

    public OrderController(ProductClient productClient) {
        this.productClient = productClient;
    }

    @GetMapping("/{id}")
    public String getOrder(@PathVariable String id) {
        String productDetails = productClient.getProduct(id);
        return "Order details for order id: " + id + ", Product: " +
productDetails;
    }
}
```

---

5. Run and Test

1. Start the **Eureka Server** (`http://localhost:8761`).
2. Start the **API Gateway**, **Product Service**, and **Order Service**.
3. Access endpoints via the gateway:
    - `http://localhost:8080/product/{id}`
    - `http://localhost:8080/order/{id}`

---

6. Optional Enhancements

- **Config Server**: Add a centralized configuration using Spring Cloud Config.

- **Resilience**: Add Circuit Breakers using Resilience4j for fault tolerance.
- **Database Integration**: Use a database like MySQL for persistent storage.
- **Docker & Kubernetes**: Containerize and orchestrate services.

Would you like detailed steps for any enhancement, or help with deployment?