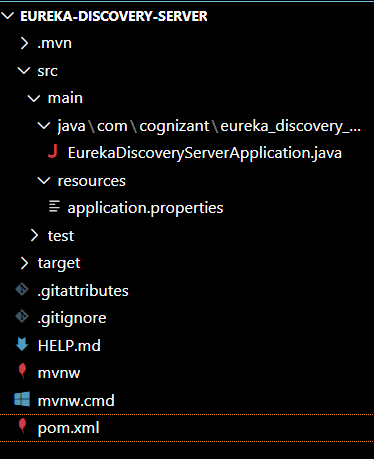
Eureka Discovery Server and register

Eureka Discovery Server acts as a service registry in a microservices architecture. It maintains a directory of all running microservices, allowing them to discover and communicate with each other without hardcoding their network locations.

**Folder Structure:**

****

**Code:**

**Pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.3.4</version>

        <relativePath />

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>eureka-discovery-server</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>eureka-discovery-server</name>

    <description>Eureka Discovery Server</description>

    <properties>

        <java.version>17</java.version>

        <spring-cloud.version>2023.0.2</spring-cloud.version>

    </properties>

    <dependencies>

        <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-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <dependencyManagement>

        <dependencies>

            <dependency>

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

                <artifactId>spring-cloud-dependencies</artifactId>

                <version>${spring-cloud.version}</version>

                <type>pom</type>

                <scope>import</scope>

            </dependency>

        </dependencies>

    </dependencyManagement>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**EurekaDiscoveryServerApplication.java:**

package com.cognizant.eureka\_discovery\_server;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

    public static void main(String[] args) {

        SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

        System.out.println("Eureka Discovery Server is running at http://localhost:8761");

    }

}

**application.properties:**

server.port=8761

# Eureka does not register itself

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

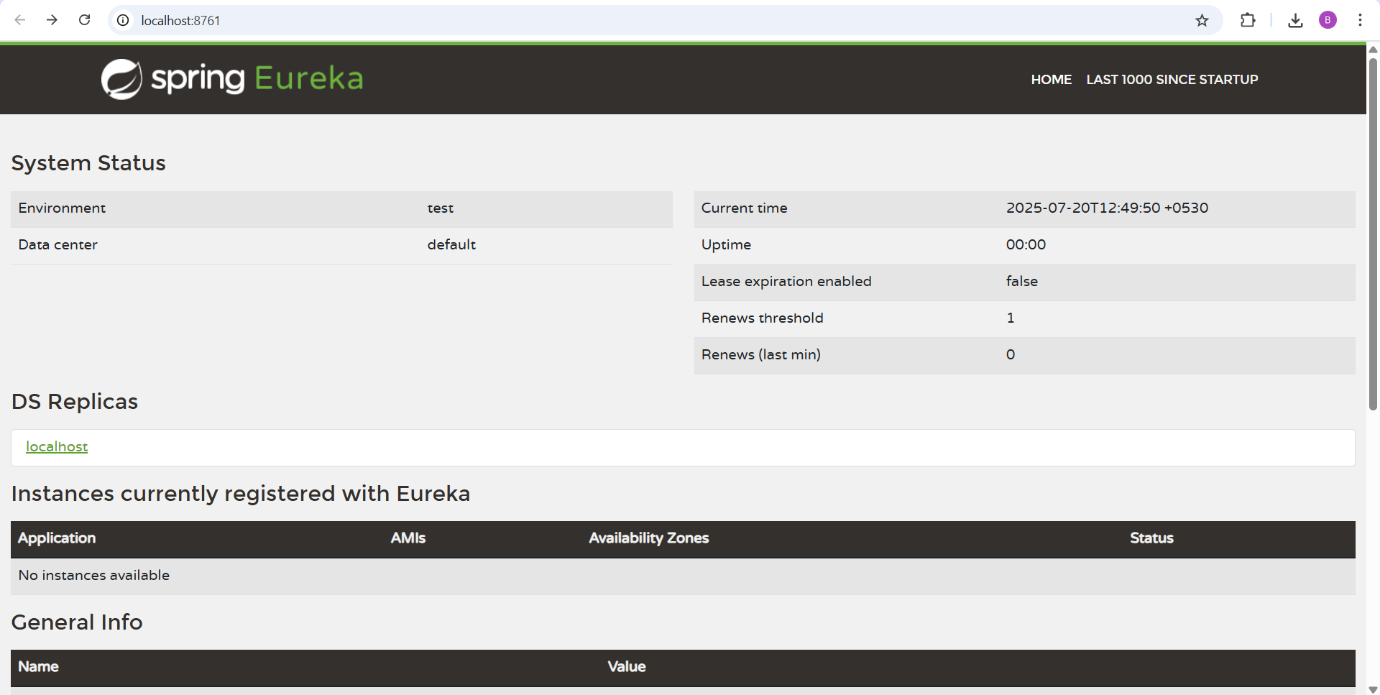
# Optional: to avoid extra logs

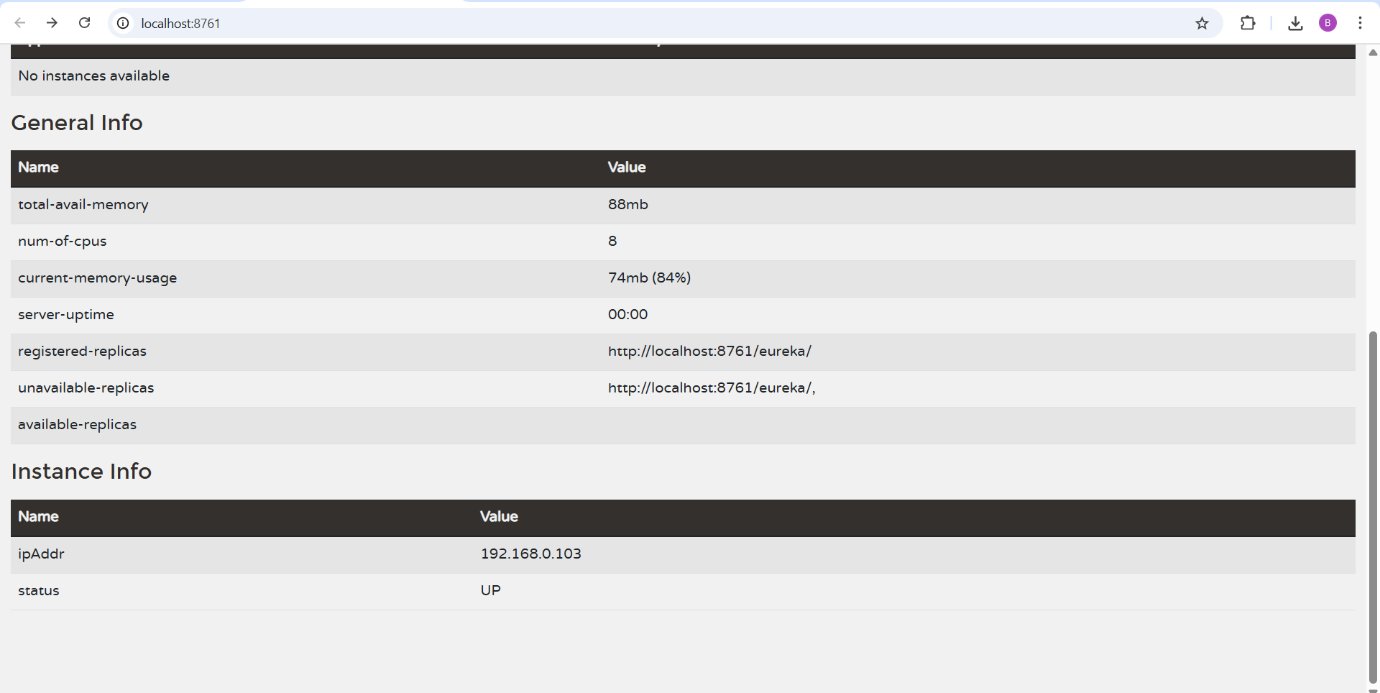
logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

**My url**: http://localhost:8761/

**Output:**

****

****

**Register services**

Registering **Accounts- service**:  
**pom.xml:**<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.3.4</version>  <!--  Downgraded for Eureka compatibility -->

        <relativePath />

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>account</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>account</name>

    <description>Account Service - Microservice</description>

    <properties>

        <java.version>17</java.version>

        <spring-cloud.version>2023.0.2</spring-cloud.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <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-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <dependencyManagement>

        <dependencies>

            <dependency>

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

                <artifactId>spring-cloud-dependencies</artifactId>

                <version>${spring-cloud.version}</version>

                <type>pom</type>

                <scope>import</scope>

            </dependency>

        </dependencies>

    </dependencyManagement>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**AccountController.java:**  
package com.cognizant.account.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AccountController {

    @GetMapping("/accounts/{number}")

    public Map<String, Object> getAccountDetails(@PathVariable String number) {

        Map<String, Object> response = new HashMap<>();

        response.put("number", number);

        response.put("type", "savings");

        response.put("balance", 234343);

        return response;

    }

}

**AccountApplication.java:**  
package com.cognizant.account;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient // Enables Eureka client registration

public class AccountApplication {

    public static void main(String[] args) {

        SpringApplication.run(AccountApplication.class, args);

        System.out.println("Account Service is running on port 8080 and registered with Eureka...");

    }

}

**Application.properties:**server.port=8080

spring.application.name=account-service

# Eureka Registration

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

Similarly, changes will be done to Loan too and get registered.

**Changes:**

1. Adding Eureka Client Dependency in pom.xml:

<dependency>

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

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

</dependency>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-dependencies</artifactId>

<version>2023.0.2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

1. Add Eureka URL in application.properties:

Eureka Registration

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

1. Enable Eureka Client in Main Class:

package com.cognizant.loan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient // Registers this service with Eureka

public class LoanApplication {

public static void main(String[] args) {

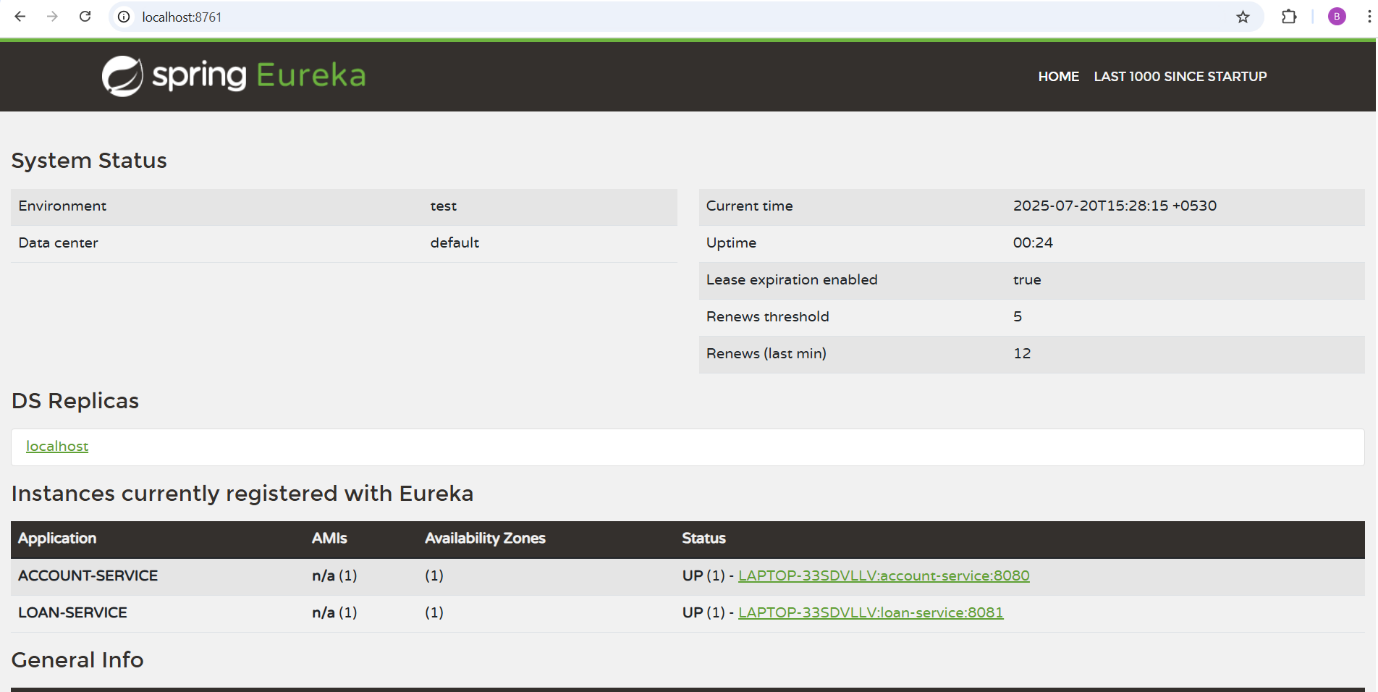
SpringApplication.run(LoanApplication.class, args);

System.out.println("Loan Service is running on port 8081 and registered with Eureka.");

}

}

**Final Output:**

****

In this task, I:

1. Created a Eureka Discovery Server running on port 8761 to serve as a central registry.
2. Configured the Account and Loan microservices as Eureka clients, enabling them to automatically register themselves with the Eureka server.
3. Verified that the services appear on the Eureka dashboard, making them discoverable for future inter-service communication or routing via an API Gateway.

This approach helps achieve loose coupling, dynamic service discovery, and scalability in distributed systems.

**API Gateway with Global Filter**

**ApiGatewayApplication.java:**

package com.cognizant.api\_gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ApiGatewayApplication {

    public static void main(String[] args) {

        SpringApplication.run(ApiGatewayApplication.class, args);

        System.out.println(" API Gateway running at http://localhost:8765");

    }

}

**LoggingFilter.java:**package com.cognizant.api\_gateway.filter;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

@Order(1)

public class LoggingFilter implements GlobalFilter {

    private static final Logger logger = LoggerFactory.getLogger(LoggingFilter.class);

    @Override

    public Mono<Void> filter(ServerWebExchange exchange,

            org.springframework.cloud.gateway.filter.GatewayFilterChain chain) {

        logger.info("Request Path: {}", exchange.getRequest().getPath());

        return chain.filter(exchange)

                .then(Mono.fromRunnable(

                        () -> logger.info("Response Status Code: {}", exchange.getResponse().getStatusCode())));

    }

}

**Pom.xml:**<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.3.4</version>

        <relativePath/>

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>api-gateway</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>api-gateway</name>

    <description>API Gateway using Spring Cloud Gateway</description>

    <properties>

        <java.version>17</java.version>

        <spring-cloud.version>2023.0.2</spring-cloud.version>

    </properties>

    <dependencies>

        <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>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <dependencyManagement>

        <dependencies>

            <dependency>

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

                <artifactId>spring-cloud-dependencies</artifactId>

                <version>${spring-cloud.version}</version>

                <type>pom</type>

                <scope>import</scope>

            </dependency>

        </dependencies>

    </dependencyManagement>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**Application.properties:**server.port=8765

spring.application.name=api-gateway

#  Eureka Client

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

#  Routes for Account Service

spring.cloud.gateway.routes[0].id=account-service

spring.cloud.gateway.routes[0].uri=lb://ACCOUNT-SERVICE

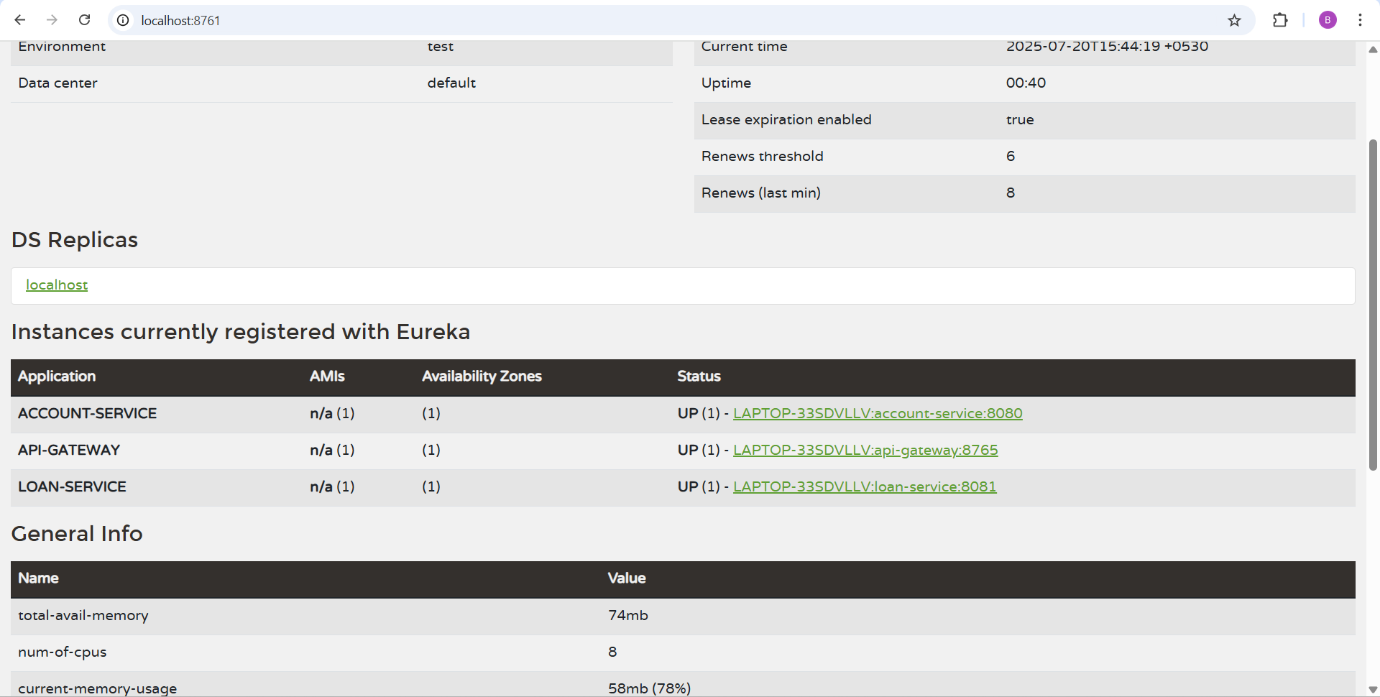
spring.cloud.gateway.routes[0].predicates[0]=Path=/accounts/\*\*

#  Routes for Loan Service

spring.cloud.gateway.routes[1].id=loan-service

spring.cloud.gateway.routes[1].uri=lb://LOAN-SERVICE

spring.cloud.gateway.routes[1].predicates[0]=Path=/loans/\*\*

**Output:**

You can see AP-Gateway is also present in the registered list.

I included an API Gateway in addition to the Eureka Discovery Server and microservices because:

Single Entry Point – Instead of calling each microservice (Account or Loan) separately using different URLs and ports, the API Gateway provides one unified endpoint for all client requests.

Example:

* Without Gateway → http://localhost:8080/accounts/1234 & http://localhost:8081/loans/1234
* With Gateway → http://localhost:8765/accounts/1234 & http://localhost:8765/loans/123

Accessing loans-service via API Gateway: (url: 8765 is api gateway’s port) 