Microservices with API gateway

Creating Microservices for account and loan

In this hands on exercises, we will create two microservices for a bank. One microservice for handing accounts and one for handling loans.

Each microservice will be a specific independent Spring RESTful Webservice maven project having it's own pom.xml. The only difference is that, instead of having both account and loan as a single application, it is split into two different applications. These webservices will be a simple service without any backend connectivity.

Follow steps below to implement the two microservices:

**Account Microservice**

* Create folder with employee id in D: drive
* Create folder named 'microservices' in the new folder created in previous step. This folder will contain all the sample projects that we will create for learning microservices.
* Open https://start.spring.io/ in browser
* Enter form field values as specified below:
  + Group: com.cognizant
  + Artifact: account

* Select the following modules o Developer Tools > Spring Boot DevTools o Web > Spring Web
* Click generate and download the zip file
* Extract 'account' folder from the zip and place this folder in the 'microservices' folder created earlier
* Open command prompt in account folder and build using mvn clean package command
* Import this project in Eclipse and implement a controller method for getting account details based on account number. Refer specification below:
  + Method: GET
  + Endpoint: /accounts/{number}
  + Sample Response. Just a dummy response without any backend o connectivity.

{ number: "00987987973432", type: "savings", balance: 234343 }

* Launch by running the application class and test the service in browser

**Loan Microservice**

* Follow similar steps specified for Account Microservice and implement a service API to get loan account details
  + Method: GET o Endpoint: /loans/{number}
  + Sample Response. Just a dummy response without any backend o connectivity.

{ number: "H00987987972342", type: "car", loan: 400000, emi: 3258, tenure: 18 }

* Launching this application by having account service already running
* This launch will fail with error that the bind address is already in use
* The reason is that each one of the service is launched with default port number as 8080. Account service is already using this port and it is not available for loan service.
* Include "server.port" property with value 8081 and try launching the application
* Test the service with 8081 port

Now we have two microservices running on different ports.

**NOTE**: The console window of Eclipse will have both the service console running. To switch between different consoles use the monitor icon within the console view.

Program:

AccountServiceApplication.java

|  |
| --- |
| package com.cognizant.account\_service;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication public class AccountServiceApplication { public static void main(String[] *args*) {  SpringApplication.run(AccountServiceApplication.class, *args*);  }  } |

AccountController.java

|  |
| --- |
| package com.cognizant.account\_service.controller;  import org.springframework.web.bind.annotation.**\***; import java.util.Map; import java.util.HashMap;    @RestController  @RequestMapping("/accounts") public class AccountController {    @GetMapping("/{number}")  public Map<String, Object> getAccount(@PathVariable String *number*) { Map<String, Object> account = new HashMap<>(); account.put("number", *number*); account.put("type", "savings"); account.put("balance", 45000); account.put("branch", "Chennai"); return account;  }  } |

pom.xml (for account services)

<?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>3.1.5</version>  <relativePath/>  </parent>    <groupId>com.cognizant</groupId>  <artifactId>account-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>account-service</name>  <description>Microservice for account</description>    <properties>  <java.version>21</java.version>  <spring-cloud.version>2022.0.5</spring-cloud.version>  </properties>    <dependencies>  *<!-- Spring Web -->*  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>    *<!-- Eureka Discovery Client -->*  <dependency>  <groupId>org.springframework.cloud</groupId> <artifactId>spring-cloud-starter-netflix-eurekaclient</artifactId>  </dependency>    *<!-- DevTools (Optional) -->*  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>    *<!-- Testing -->*  <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>

Loan Services

LoanServiceApplication.java

|  |
| --- |
| package com.cognizant.loan\_service;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication public class LoanServiceApplication { public static void main(String[] *args*) {  SpringApplication.run(LoanServiceApplication.class, *args*);  }  } |

LoanController.java

|  |
| --- |
| package com.cognizant.loan\_service.controller;  import org.springframework.web.bind.annotation.**\***;  import java.util.Map;    @RestController  @RequestMapping("/loans") public class LoanController {    @GetMapping("/{customerId}") public Map<String, String> getLoanDetails(@PathVariable String *customerId*)  {  return Map.of(  "customerId", *customerId*,  "loanNumber", "LN987654321",  "type", "Home Loan",  "status", "Approved");  }  } |

Pom.xml (for loan Services)

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>loan-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

|  |
| --- |
| <packaging>jar</packaging>    <name>loan-service</name>  <description>Loan microservice for Eureka demo</description>  <properties>  <java.version>21</java.version>  <spring-cloud.version>2022.0.5</spring-cloud.version>  </properties>    *<!--* ✅ *Fixed parent section -->*  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>3.1.3</version>  <relativePath/>  </parent>    <dependencies>  *<!-- Web -->*  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>    *<!-- Eureka Client -->*  <dependency>  <groupId>org.springframework.cloud</groupId> <artifactId>spring-cloud-starter-netflix-eurekaclient</artifactId>  </dependency>    *<!-- Optional: Devtools -->*  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  </dependency>    *<!-- Optional: Testing -->*  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </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.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<release>${java.version}</release>

</configuration>

</plugin>

</plugins>

</build>

</project>

Eureka Discovery Server:

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*); }  } |

pom.xml (for eureka server)

<?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>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId> <artifactId>eureka-discovery-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

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

<properties>

<java.version>21</java.version>

<spring-cloud.version>2022.0.5</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-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</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>

API-gateway

ApiGatewayApplication.java

|  |
| --- |
| package com.cognizant.api\_gateway;    import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  @SpringBootApplication @EnableDiscoveryClient public class ApiGatewayApplication {  public static void main(String[] *args*) {  SpringApplication.run(ApiGatewayApplication.class, *args*);  }  } |

LogFilter.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.cloud.gateway.filter.GatewayFilterChain; import org.springframework.stereotype.Component; import org.springframework.web.server.ServerWebExchange; import reactor.core.publisher.Mono;    @Component public class LogFilter implements GlobalFilter {    private static final Logger logger =  LoggerFactory.getLogger(LogFilter.class);    @Override public Mono<Void> filter(ServerWebExchange *exchange*, GatewayFilterChain *chain*) { logger.info("Request received for path: {}", *exchange*.getRequest().getPath()); return *chain*.filter(*exchange*);  }  } |

Pom.xml (for API-gateway)

<?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>3.1.5</version>  <relativePath/>  </parent>    <groupId>com.cognizant</groupId>  <artifactId>api-gateway</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>api-gateway</name>    <properties>  <java.version>21</java.version>  <spring-cloud.version>2022.0.5</spring-cloud.version>  </properties>    <dependencies>  *<!-- Spring Cloud Gateway -->*  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-gateway</artifactId>  </dependency>  *<!-- Eureka Discovery Client -->*  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId> </dependency>  *<!-- DevTools (Optional) -->*  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  *<!-- Test -->*  <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>

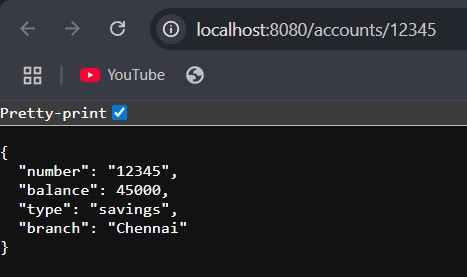
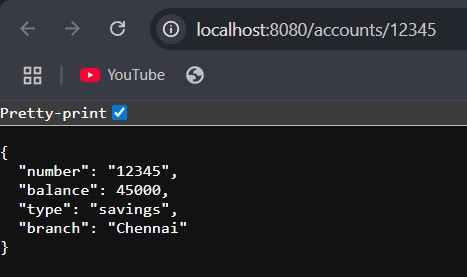
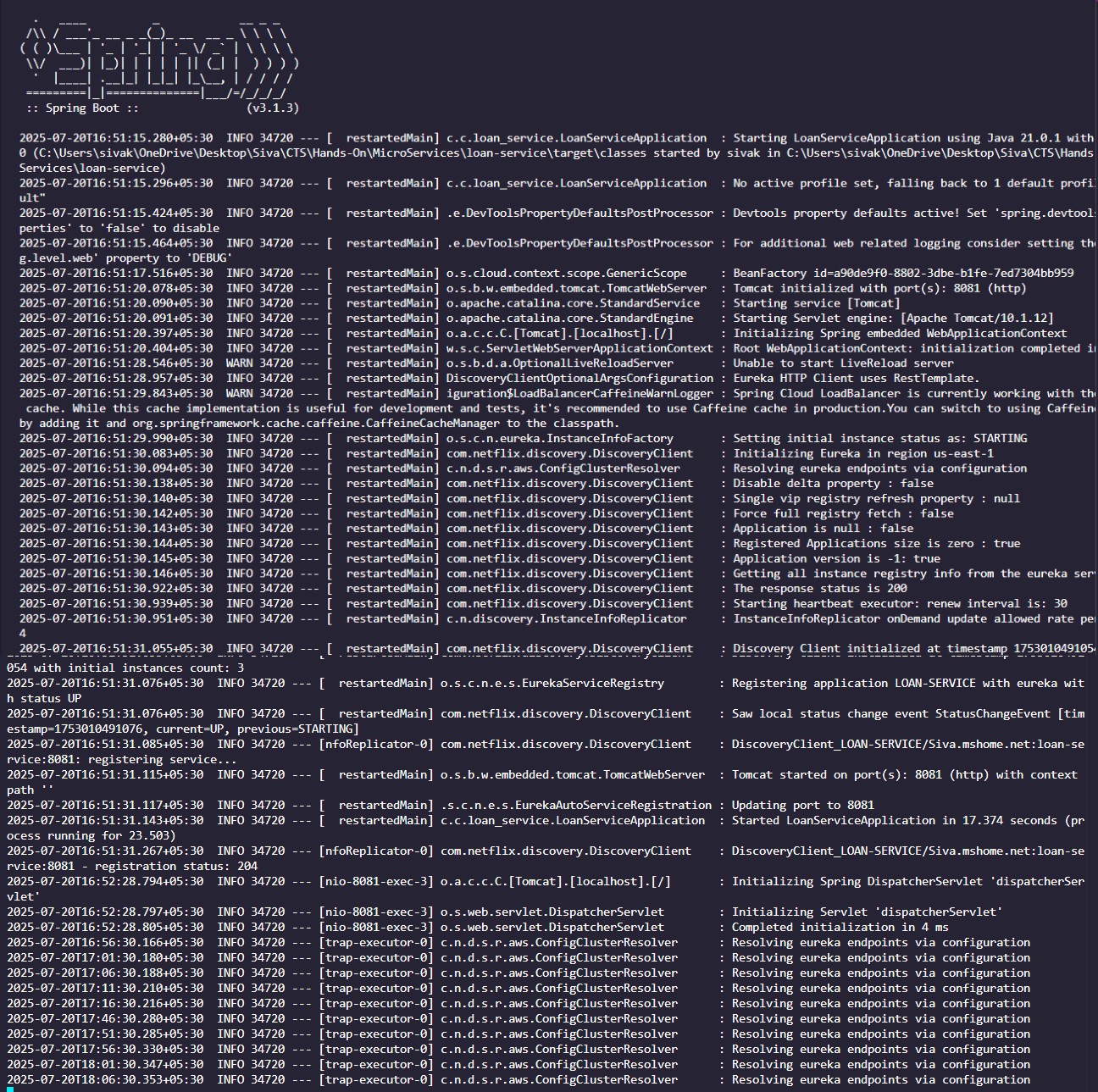
</build>

</project>

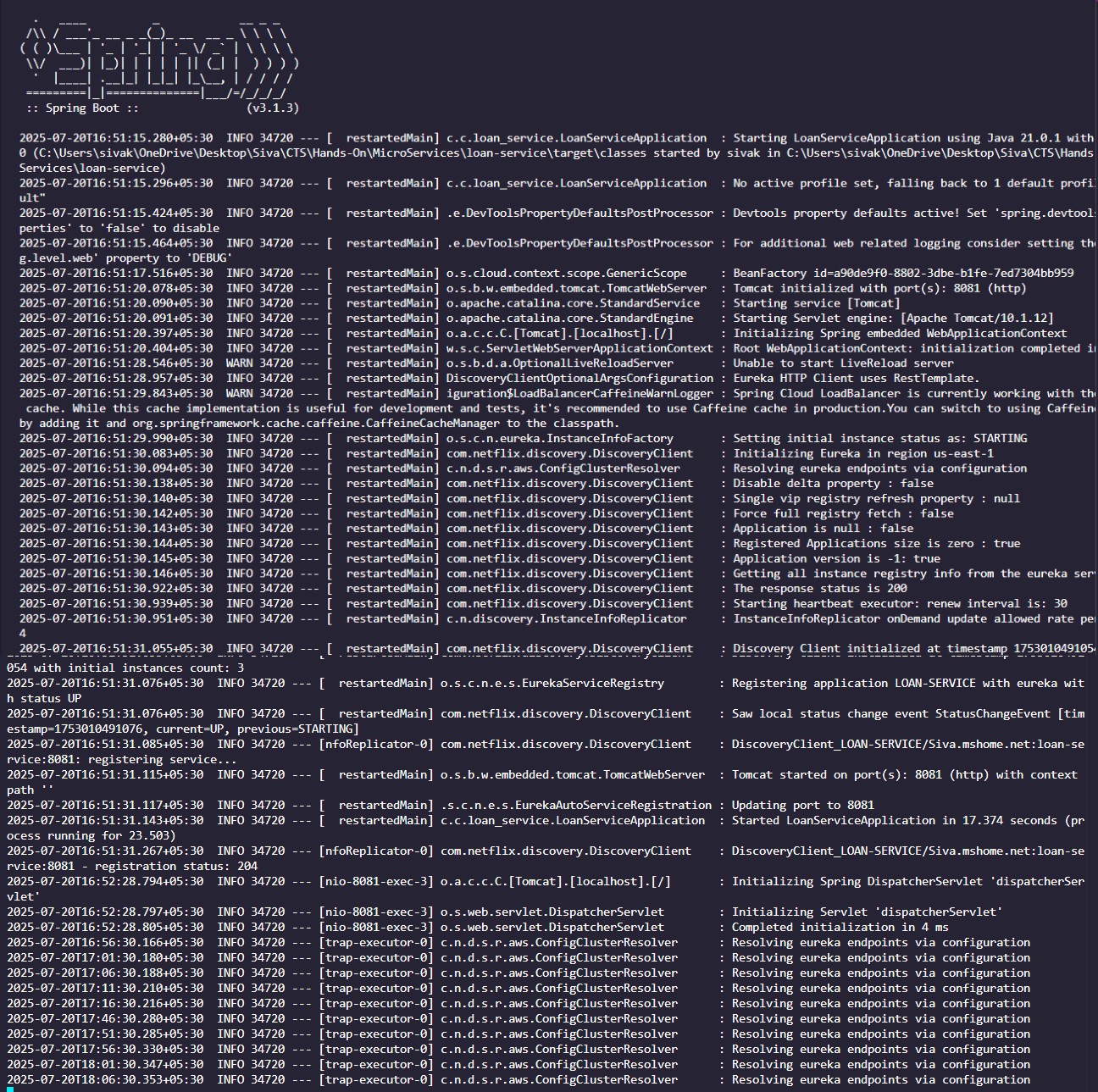
Output:

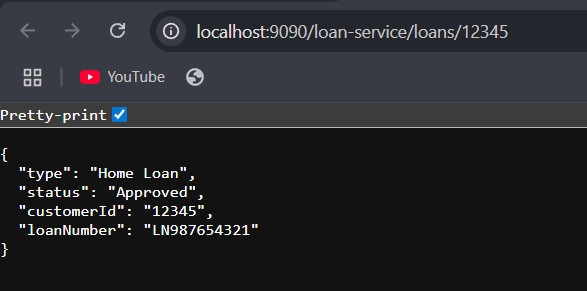
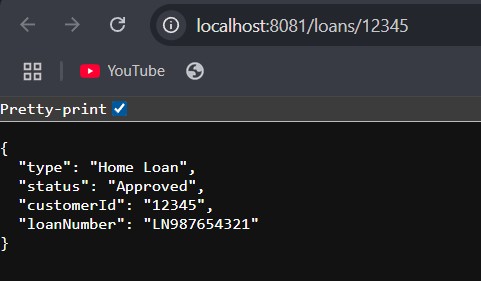
Account

Service:

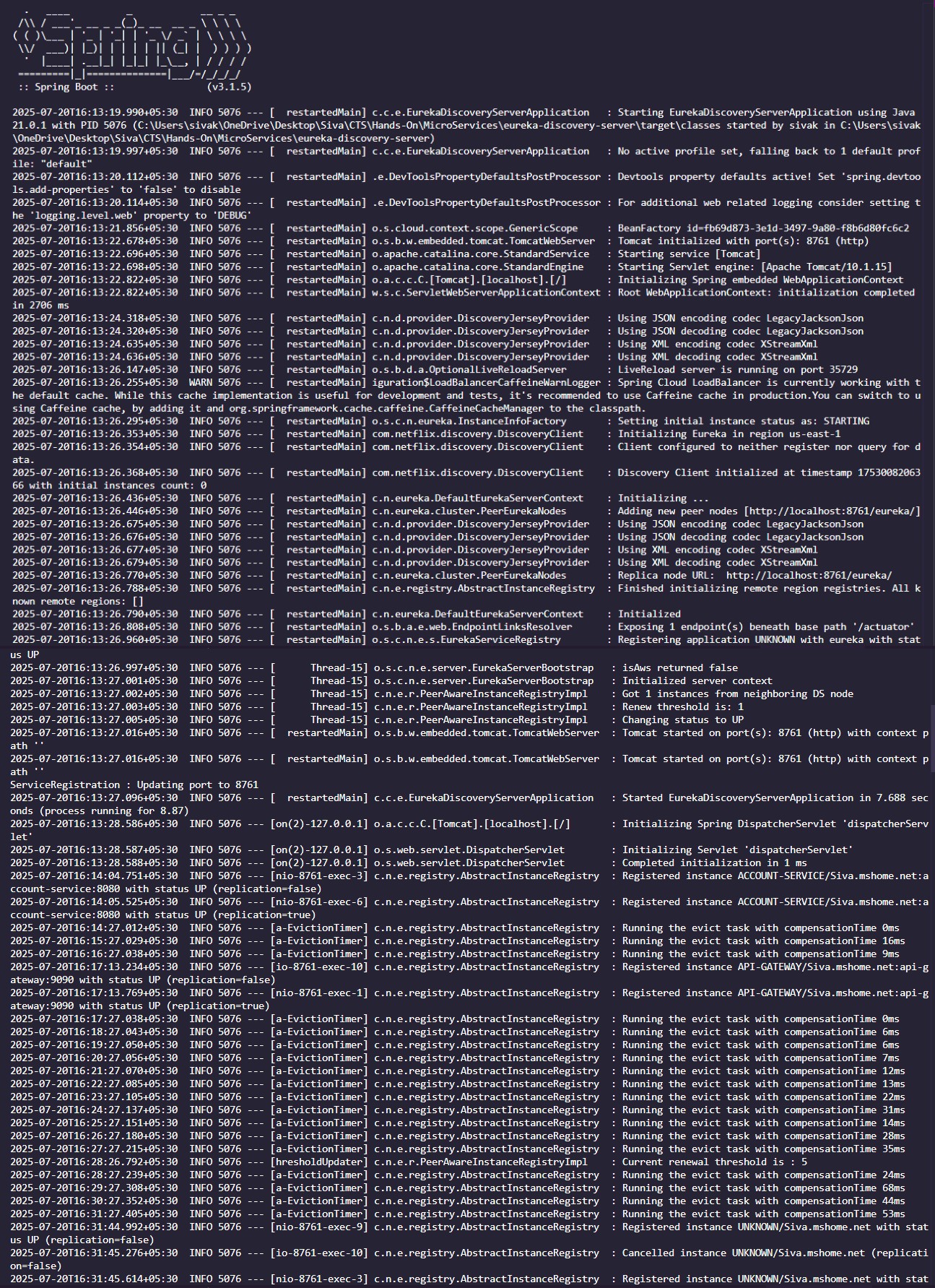


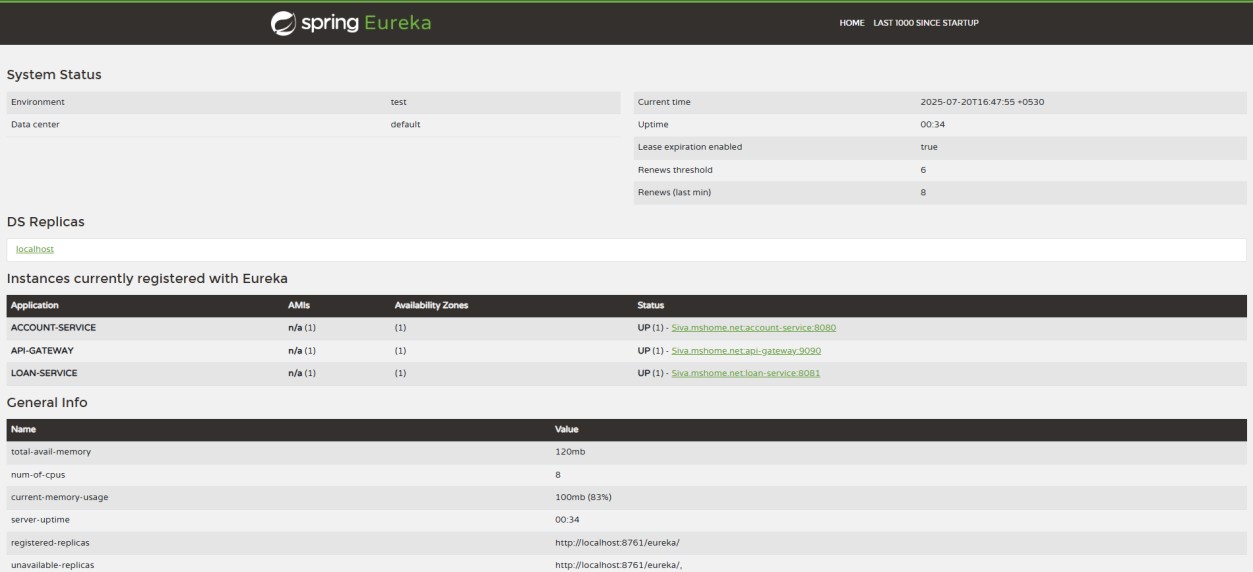
Loan Service:





Eureka Server





API Gateway

