

WEEK 5:

Microservices with API gateway

Create Eureka Discovery Server and register microservices

The screenshot shows the Spring Eureka web interface. The top navigation bar includes the 'spring Eureka' logo and links for 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing system configuration and metrics.

Environment	test	Current time	2025-07-18T12:09:45 +0530
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0
- DS Replicas:** A section with a single entry 'localhost'.
- Instances currently registered with Eureka:** A table with headers 'Application', 'AMIs', 'Availability Zones', and 'Status'. It currently shows 'No instances available'.
- General Info:** A section at the bottom of the main content area.

This screenshot shows the same Spring Eureka web interface, but with a different timestamp and uptime. The browser tabs show 'VTU Online Class - Change Pass', 'Spring Initializr', and 'Eureka'. The 'System Status' table is as follows:

Environment	test	Current time	2025-07-18T11:39:09 +0530
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

The 'DS Replicas' section still shows 'localhost'. The 'Instances currently registered with Eureka' table remains empty with the message 'No instances available'.

Microservices with API gateway

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

◦ Developer Tools > Spring Boot DevTools ◦ 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 connectivity.

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

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

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

Loan Microservices:

Follow similar steps specified for Account Microservice and implement a service API to get loan account details

- Method: GET
- Endpoint: /loans/{number}
- Sample Response. Just a dummy response without any backend 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.

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