

Microservice

using version

- java 17
- springboot 3.1.5
- <spring-cloud.version>2022.0.4</spring-cloud.version>

```
<?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/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.gateway</groupId>
  <artifactId>ApiGateway</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ApiGateway</name>
  <description>this is api gateway</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
  </scm>
```

```
<tag/>
<url/>
</scm>
<properties>
    <java.version>17</java.version>
    <spring-cloud.version>2022.0.4</spring-cloud.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>

    <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>

```

```

        </dependency>
    </dependencies>
</dependencyManagement>

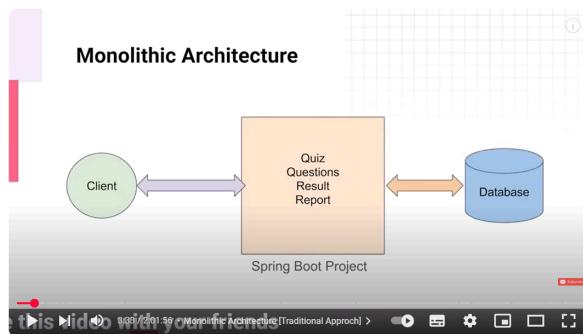
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

</project>

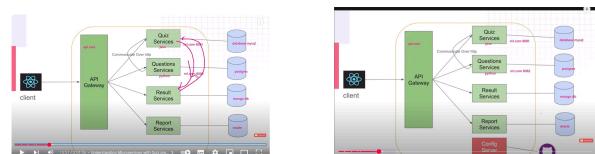
```

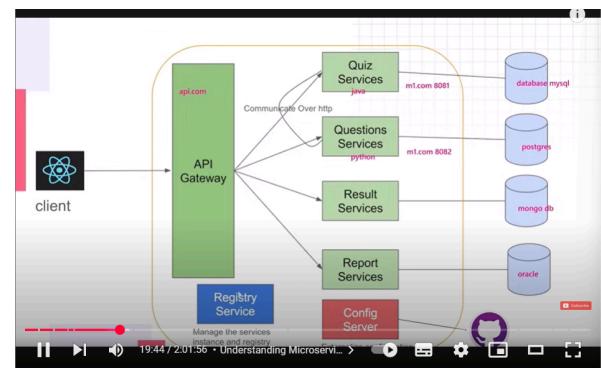
Different Between

- Monolithic Architecture



- microservice architecture

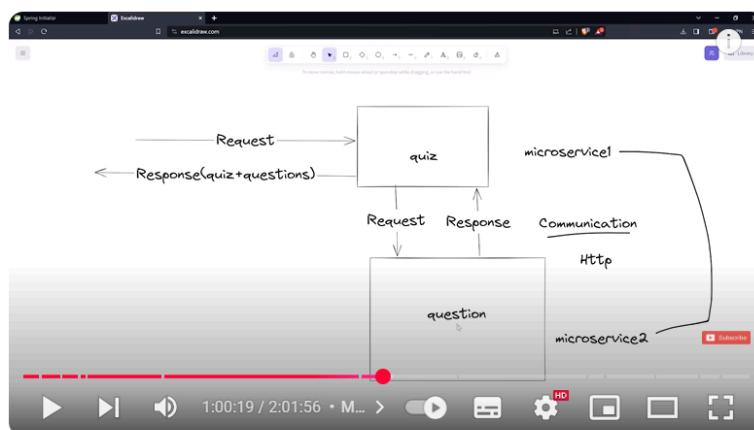




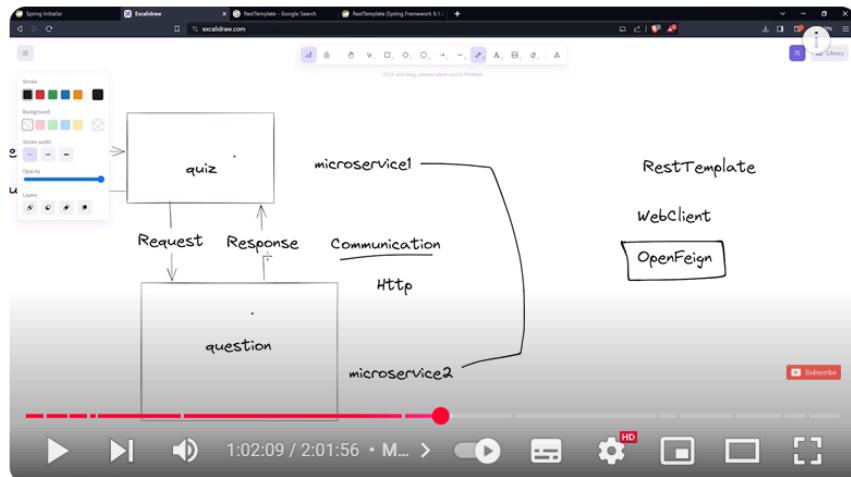
- Api Gateway - Used to manage the URL of the server.
- we have multiple server with different port we want access all server with only one port.

Registry Service - Used to store all service information.

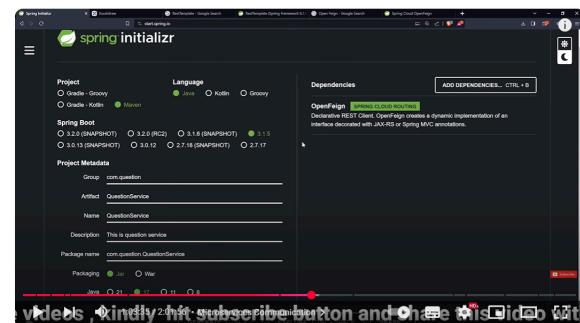
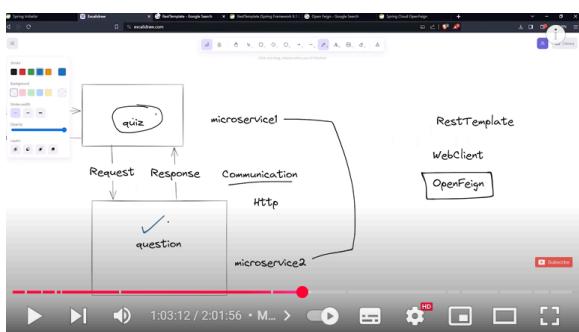
Calling one service to another service using OpenFeign - workflow



- RestTemplate is deprecated.

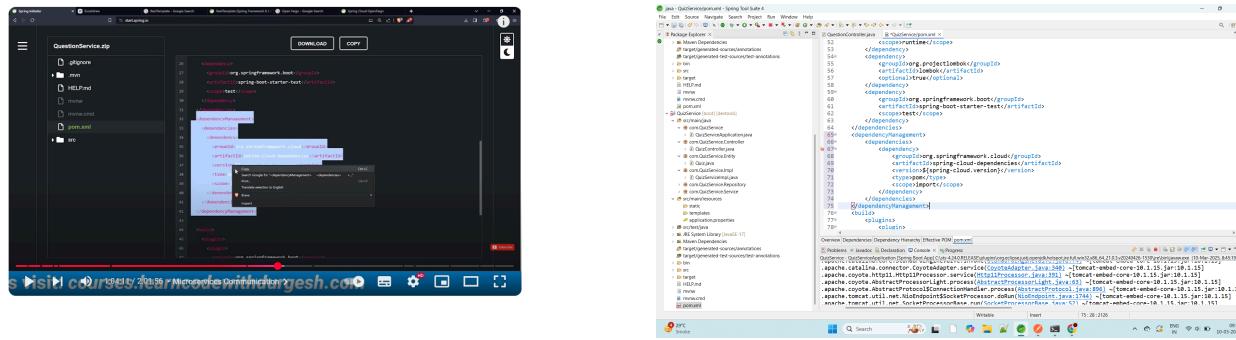


Which service to call, there add the OpenFing library.

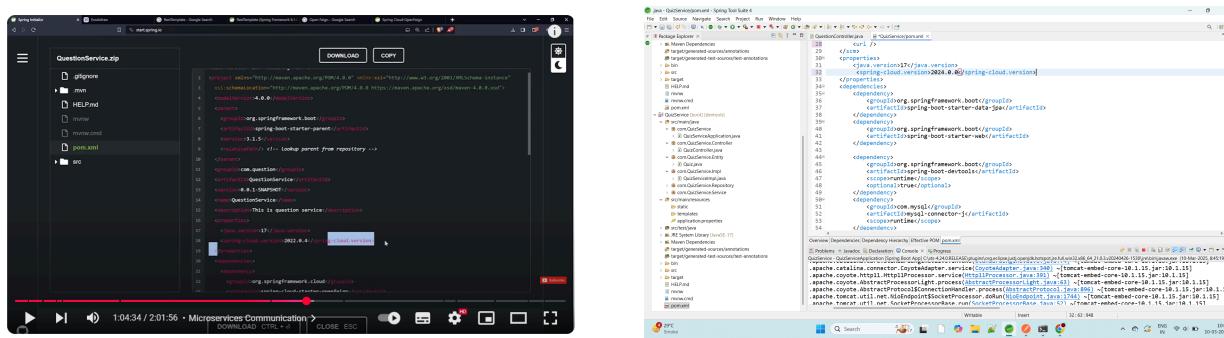


Not put directly openFing dependency

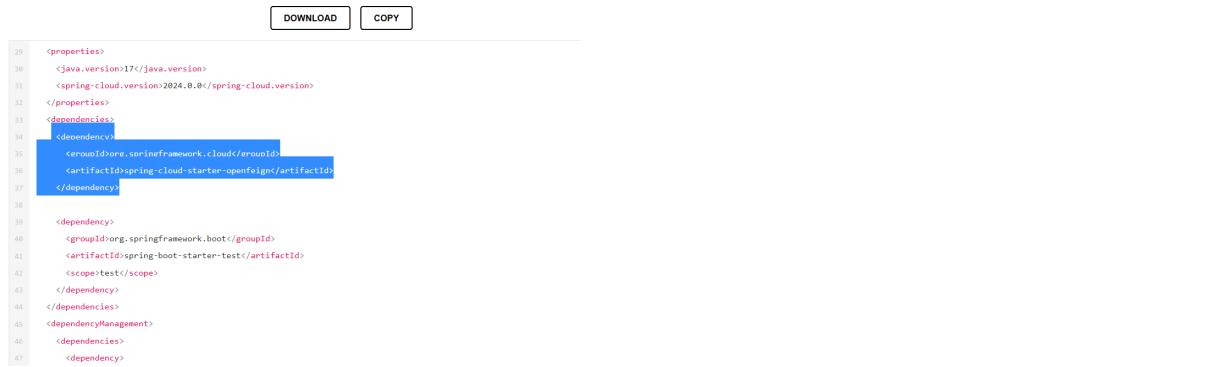
- first open the project (explore) then copy the dependency management
- copy dependency management



- copy property version



- Now copy the dependency and paste



- Create a service and use annotation of `fing`

The screenshot shows the Spring Tool Suite interface with the QuizService project open. The Package Explorer view on the left lists various Java files and resources. The central editor shows the `QuestionController.java` file, which contains code for a Feign client. Below the editor is a terminal window showing the application's shutdown logs. The bottom status bar indicates the date and time as 10-03-2025.

```

1 package com.QuizService.Service;
2
3 import java.util.List;
4
5 import org.springframework.cloud.openfeign.FeignClient;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8
9 import com.QuizService.Entity.Question;
10
11 @FeignClient(url="http://localhost:8082", value="Question-client")
12 public interface QuestionClient {
13     @GetMapping("/question/quiz/{id}")
14     List<Question> getQuestionOfQuiz(@PathVariable Long id);
15 }
16

```

```

18 --- [n(90)-127.0.0.1] o.apache.catalina.core.StandardService : Stopping service [Tomcat]
18 --- [n(90)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[] : Destroying Spring FrameworkServlet 'dispatcherServlet'
18 --- [n(90)-127.0.0.1] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
18 --- [n(90)-127.0.0.1] com.zaxxer.hikari.HikariDataSource : HikariPool-7 - Shutdown initiated...
18 --- [n(90)-127.0.0.1] com.zaxxer.hikari.HikariDataSource : HikariPool-7 - Shutdown completed.

```

- also enable feign clients - its automatic scan all feign client service.

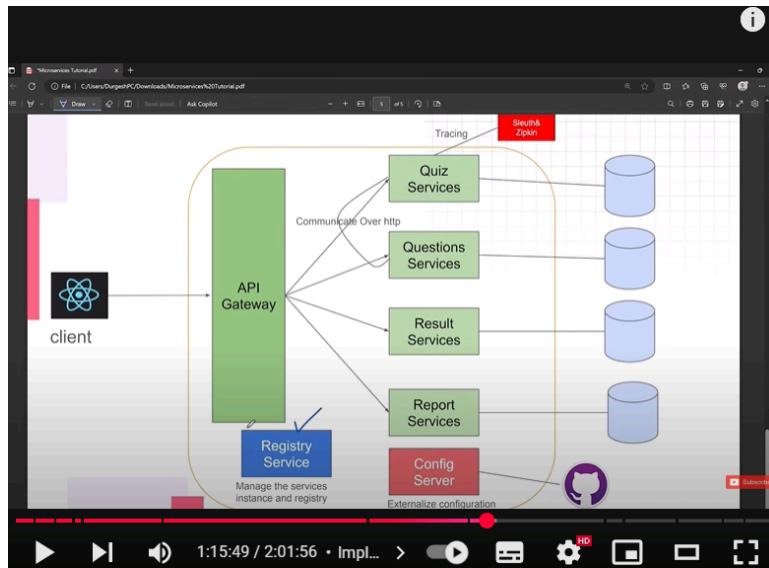
The screenshot shows the Spring Tool Suite interface with the QuizServiceApplication.java file open in the editor. The browser window next to it displays a YouTube video titled "Master Microservices using Spring Boot 3.X | Hindi". The video player shows the progress bar at 1:09:23 / 2:01:56. The bottom status bar indicates the date and time as 10-03-2025.

```

1 package com.QuizService;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableFeignClients
7 public class QuizServiceApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(QuizServiceApplication.class, args);
11     }
12 }
13
14
15
16

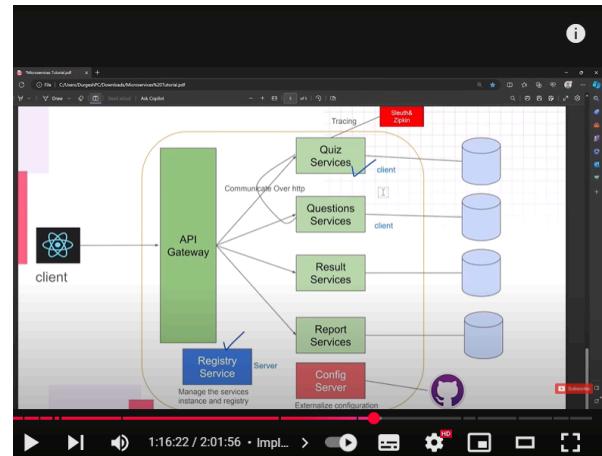
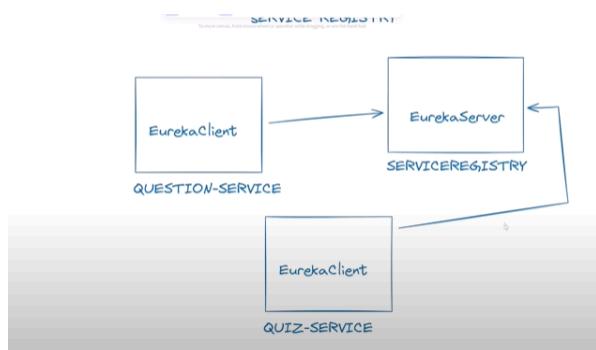
```

Service Registry or Discovery Service



1. all micro services are registered
2. instance will be manage
3. Load balance will be managed.

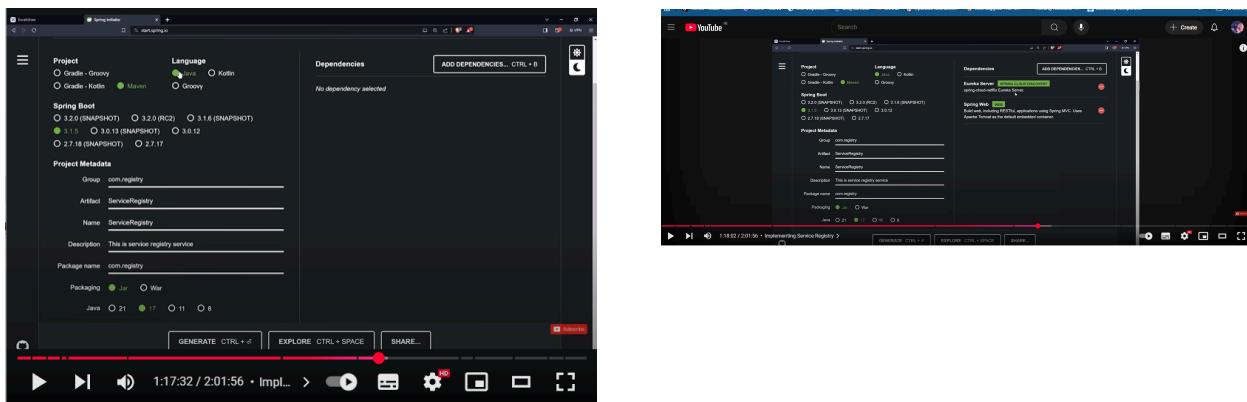
Registry Service works as a service, and all the services work as clients



- Disable the serverRegistry as a client
- Enable the all server as a client

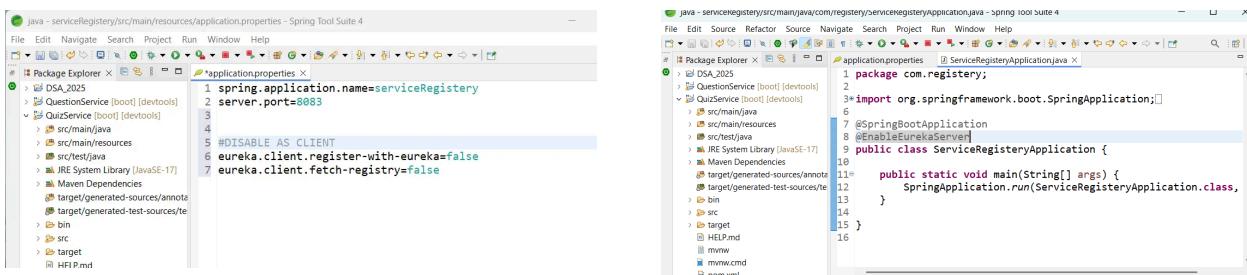
Implementation

- create project
- add dependency



Registry Service -server

- Disable the serviceRegistry as a client



check Eureka server are start or not

The screenshot shows the Spring Eureka dashboard with the following details:

- System Status:**

Environment	test	Current time	2025-03-10T12:22:41 +0530
Data center	default	Uptime	00:05
		Lease expiration enabled	false
		Renews threshold	3
		Renews (last min)	1
- DS Replicas:**

Instances currently registered with Eureka

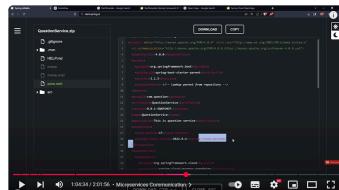
Application	AMIs	Availability Zones	Status
QUIZSERVICE	n/a (1)	(1)	UP (1) - localhost:QuizService:8081
- General Info:**

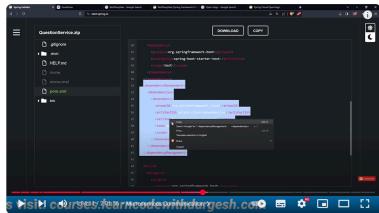
Name	Value
total-avail-memory	71mb
num-of-cpus	8

Registry Service - Client

Enable the other Service as a client

- First Search dependency and open project and copy all things and then dependency
 - i. past dependencymanager
 - ii. past the version
 - iii. add a Dependency eureka client in the other server





4th step

- New Spring Boot version does not need to add an annotation in the client to enable service as a client only dependency add.
- only set the Eureka server URL in the client

java - QuizService/src/main/resources/application.properties - Spring Tool Suite 4

```

File Edit Navigate Search Project Run Window Help
File Edit Navigate Search Project Run Window Help
Package Explorer application.properties application.properties
DSA_2025
  QuestionService [boot] [devtools]
  QuizService [boot] [devtools]
    src/main/java
      com.QuizService
        QuizServiceApplication.java
        QuizServiceController
        QuizServiceEntity
        QuizServiceImpl
        QuizServiceRepository
        QuizServiceService
    src/main/resources
      static
      templates
      application.properties
    src/test/java
      JRE System Library [JavaSE-17]
      Maven Dependencies
      target/generated-sources/annotations
      target/generated-test-sources/test-annotations
    bin
    src
    target
      HELP_md
      mvnw
      mvnw.cmd
      pom.xml
  serviceRegistry [boot]
    src/main/java
      com.registry
        ServiceRegistryApplication.java
    src/main/resources
      static
      templates
      application.properties
    src/test/java
      JRE System Library [JavaSE-17]
      Maven Dependencies

```

```

1 spring.application.name=QuizService
2 server.port=8081
3 spring.datasource.username=root
4 spring.datasource.password=root
5 spring.datasource.url=jdbc:mysql://localhost:3306/micro
6
7 spring.jpa.hibernate.ddl-auto=update
8 spring.jpa.show-sql=true
9
10 eureka.client.service-url.defaultZone=http://localhost:8083/eureka/
11

.
.
.

registry.ServiceRegistryApplication : Starting ServiceRegistryApplication using Java 21.0.3 with PID 12016 (D:\Language\)
registry.ServiceRegistryApplication : No active profile set, falling back to 1 default profile: "default"
.s.cloud.context.scope.GenericScope : BeanFactory id=a9d2be50-eb69-37aa-8284-7bd85bccfabd
.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8083 (http)
.apache.catalina.core.StandardService : Starting service [Tomcat]
.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.15]
.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
.s.s.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1584 ms
.n.d.provider.DiscoveryJerseyProvider : Using JSON encoding codec LegacyJacksonJson
.n.d.provider.DiscoveryJerseyProvider : Using JSON decoding codec LegacyJacksonJson

```

35°C Sunny 12:23 10-03-2025

- Run all the client server Eureka Auto Handle all that instance.

The screenshot shows a browser window with multiple tabs open. The active tab is 'localhost:8083' displaying the Spring Eureka dashboard. The dashboard has sections for 'System Status', 'DS Replicas', and 'General Info'. In 'System Status', there are environment details like 'Environment: test', 'Data center: default', and various uptime metrics. A red warning message at the top states: 'EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.' The 'DS Replicas' section lists 'QUESTIONSERVICE' and 'QUIZSERVICE' with their respective statuses and URLs. The 'General Info' section shows a Windows taskbar with various icons and system information.

Load Balancing or Auto Balancing

- means we have multiple the same service when one service goes down, then automatically call another service.
- When 2 or more of the same service are running, then Eureka also handles that.
- Eureka automatic handles all things, only put the dependency and one-time configure.

Note: Only put dependency and configure the name of eureka which function id calling. Auto manage all things

DS Replicas

[localhost](#)

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
QUESTION-SERVICE	n/a (2)	(2)	UP (2) - host.docker.internal:QUESTION-SERVICE:8082, host.docker.internal:QUESTION-SERVICE:9092
QUIZ-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:QUIZ-SERVICE:8081

General Info

Name	Value
total-avail-memory	96mb
num-of-cpus	6
current-memory-usage	68mb (70%)
server-uptime	00:10
registered-replicas	http://localhost:8761/eureka/
available-replicas	http://localhost:8761/eureka/
available-replicas	1:39:50 / 2:01:56 • Load Balancing with Example >

Add the dependency

- in the Quiz service because the quiz service calls the question service.
- whom is calling there? Put dependency there

```

<developerConnection>
  <tag>
  </tag>
</developerConnection>

<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-loadbalancer</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

```

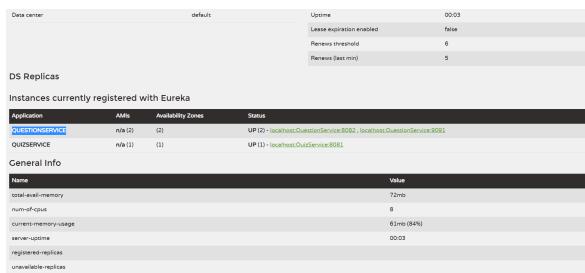
DOWNLOAD CTRL + d **CLOSE ESC**

Dependencies **ADD DEPENDENCIES... CTRL + B**

Cloud LoadBalancer **SPRING CLOUD ROUTING**
Client-side load-balancing with Spring Cloud LoadBalancer.

copy name of service from Eureka

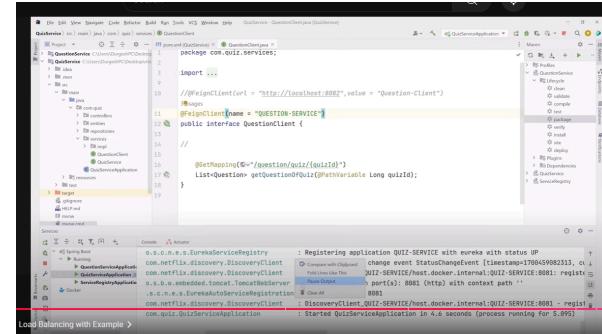
- we have multiple the same service so we not need do call with the URL.
- multiple same service like question



- Two Same Question service run but in different ports

Past where calling the service

- Now, if we have multiple the same service, if one service is down, then another will be run and automatically called with the Eureka name.
- Now Port does not matter.



Api GateWay

Problem

- We have multiple URL, when we want to call then call on that

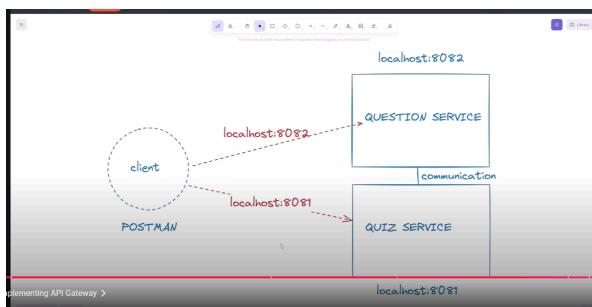
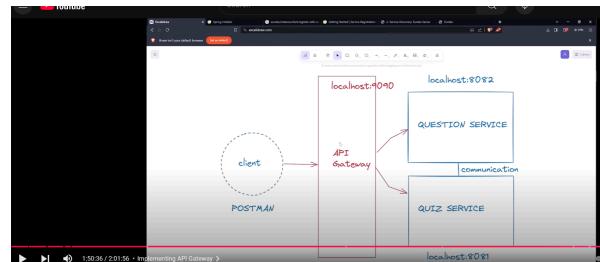
solution

particular URL.

eg: The Question service has all questions. and the quiz service has all quizzes. we know that the quiz service also calls the question service. but if we want to all questions then we need to call question service but that in other port number.

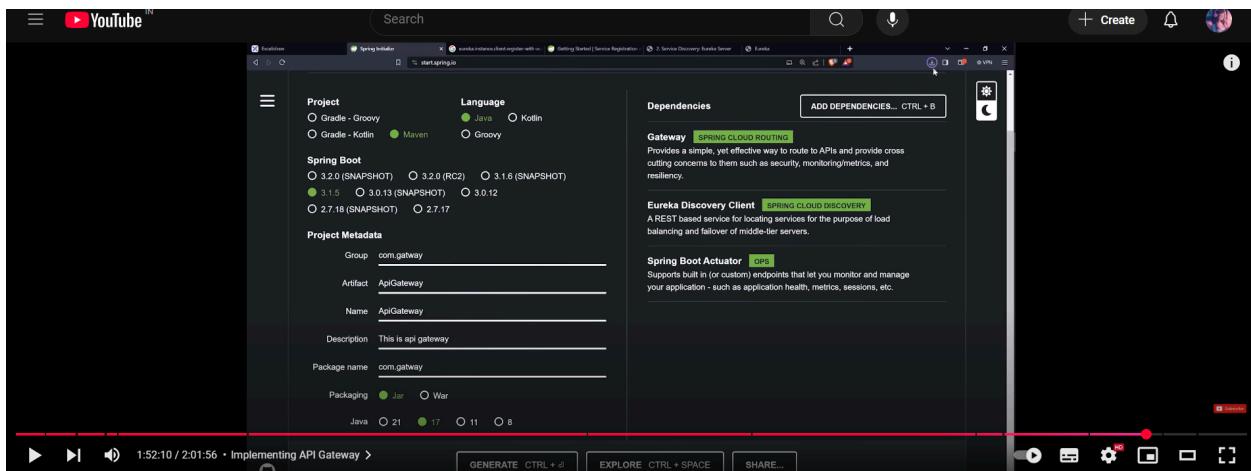
- So if we have multiple services, then it's difficult to handle.

- Api Gateway is also a services



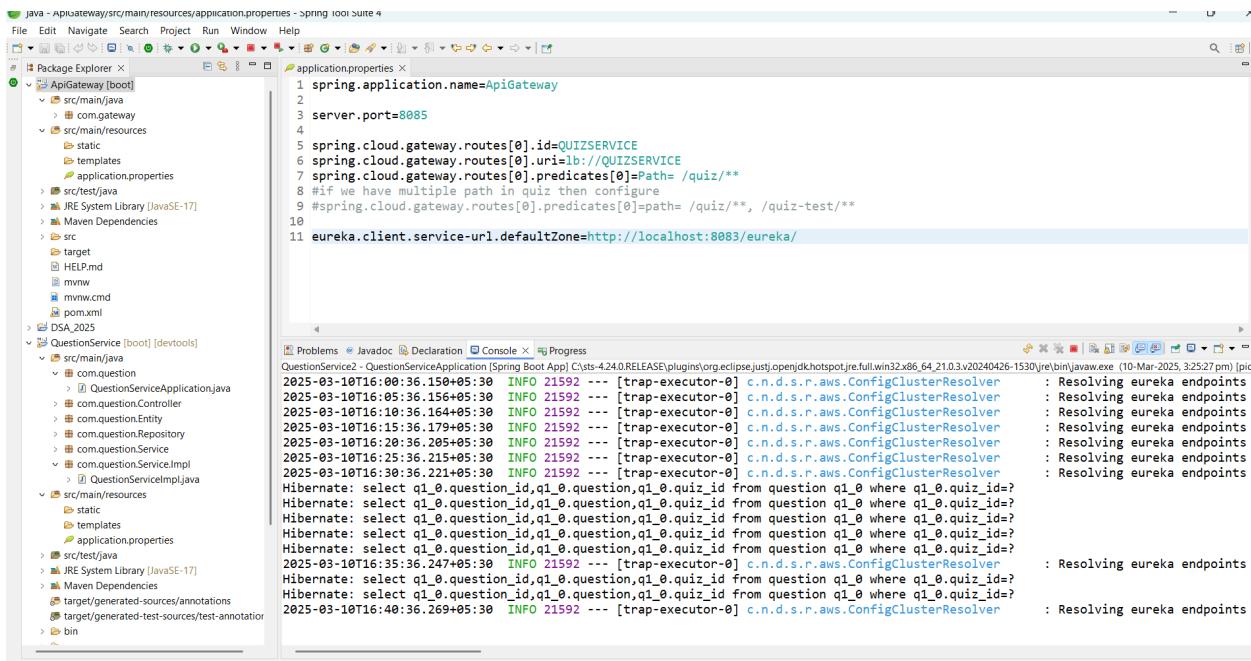
Implementation

- In the Api Gateway only URL configuration, which service runs when
- **creating a project**



- In Gateway dependency, remove (- MVC) other not supported.
- dependency

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



```

spring.application.name=ApiGateway
server.port=8085

spring.cloud.gateway.routes[0].id=QUIZSERVICE
spring.cloud.gateway.routes[0].uri=lb://QUIZSERVICE
spring.cloud.gateway.routes[0].predicates[0]=path= /quiz/**
#if we have multiple path in quiz then configure
#spring.cloud.gateway.routes[0].predicates[0]=path= /quiz/
,/quiz-test/

```

```

1 spring.application.name=ApiGateway
2
3 server.port=8085
4
5 spring.cloud.gateway.routes[0].id=QUIZSERVICE
6 spring.cloud.gateway.routes[0].uri=lb://QUIZSERVICE
7 spring.cloud.gateway.routes[0].predicates[0]=Path= /quiz/**
8 #if we have multiple path in quiz then configure
9 #spring.cloud.gateway.routes[0].predicates[0]=path= /quiz/**, /quiz-test/**
10
11
12 spring.cloud.gateway.routes[1].id=QUESTIONSERVICE
13 spring.cloud.gateway.routes[1].uri=lb://QUESTIONSERVICE
14 spring.cloud.gateway.routes[1].predicates[0]=Path= /question/**
15
16 eureka.client.service-url.defaultZone=http://localhost:8083/eureka/

```

The screenshot shows the Eclipse IDE interface with the 'application.properties' file open in the central editor. The file content is as follows:

```

1 spring.application.name=ApiGateway
2
3 server.port=8085
4
5 spring.cloud.gateway.routes[0].id=QUIZSERVICE
6 spring.cloud.gateway.routes[0].uri=lb://QUIZSERVICE
7 spring.cloud.gateway.routes[0].predicates[0]=Path= /quiz/**
8 #if we have multiple path in quiz then configure
9 #spring.cloud.gateway.routes[0].predicates[0]=path= /quiz/**, /quiz-test/**
10
11
12 spring.cloud.gateway.routes[1].id=QUESTIONSERVICE
13 spring.cloud.gateway.routes[1].uri=lb://QUESTIONSERVICE
14 spring.cloud.gateway.routes[1].predicates[0]=Path= /question/**
15
16 eureka.client.service-url.defaultZone=http://localhost:8083/eureka/

```

The left sidebar shows the 'Package Explorer' with the 'src/main/resources/application.properties' file selected. The bottom status bar shows system information like temperature (38°C), battery level, and network status.

Remaining

- sleuth & zipkin
- configure server

