Microservice

using version

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

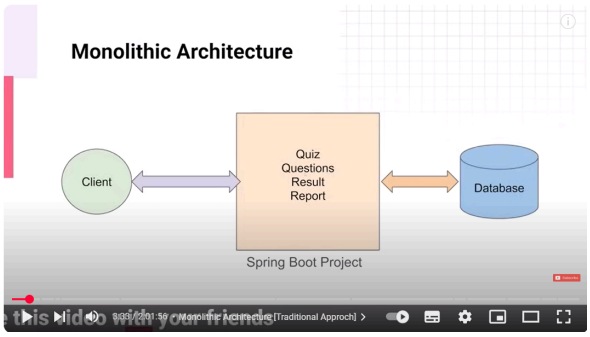
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apac
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</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/>
```

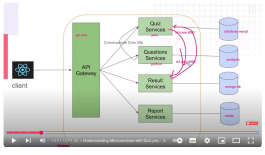
```
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>17</java.version>
    <spring-cloud.version>2022.0.4</spring-cloud.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>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

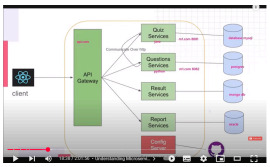
```
        </dependency>
      </dependencies>
    </dependencyManagement>

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

</project>
```

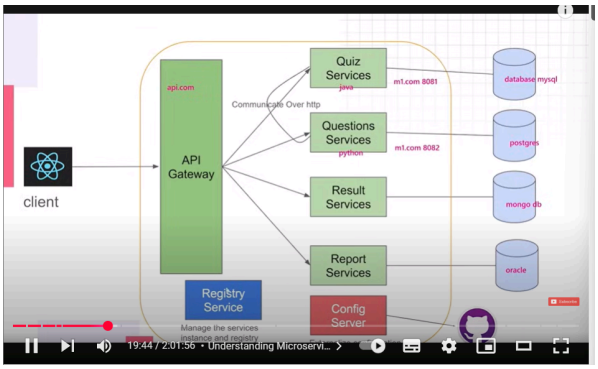
Different Between

- **Monolithic Architecture**
- **microservice architecture**





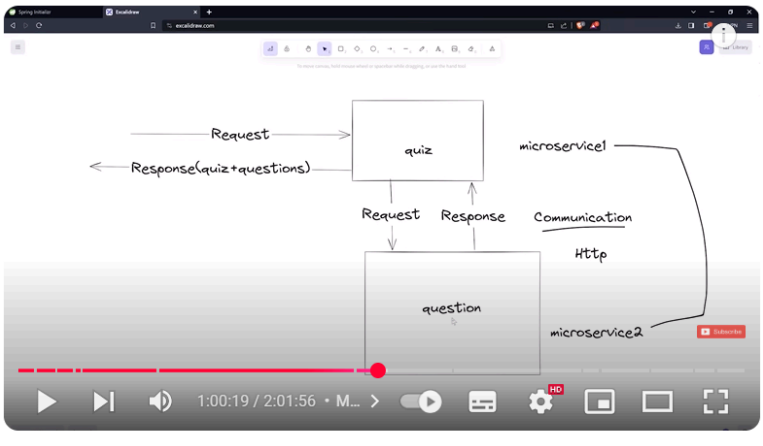




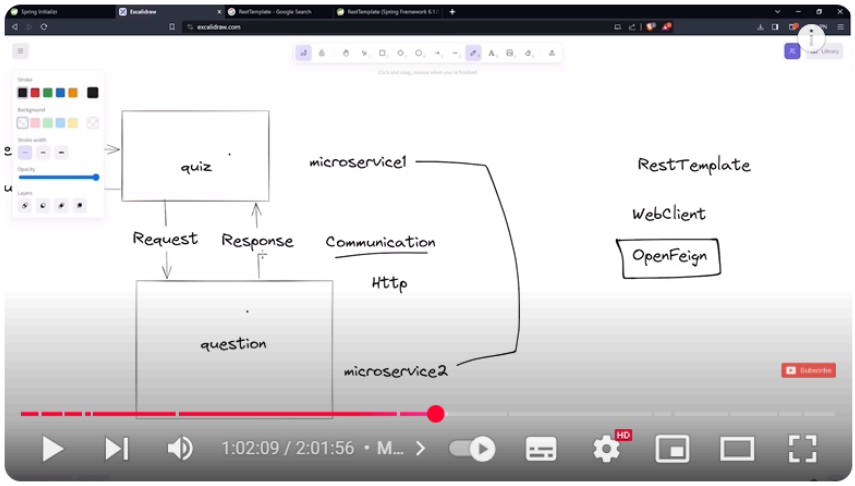
- Api Gateway - Used to manage the URL of the server.
- we have multiple server with different port we want access all server with only one port.

Registry Service - Used to store all service information.

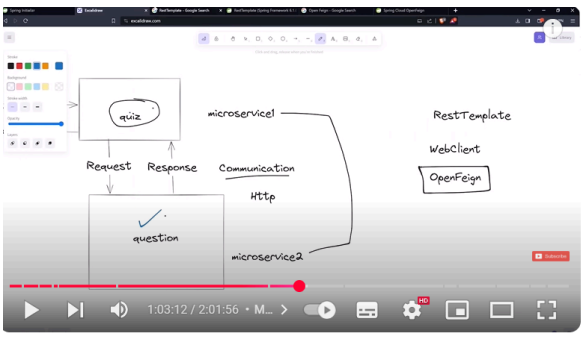
Calling one service to another service using OpenFeign - workflow

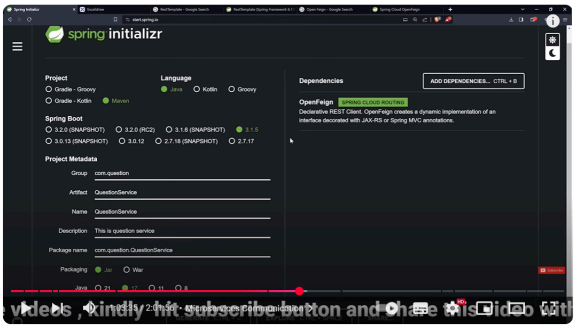


- RestTemplate is deprecated.



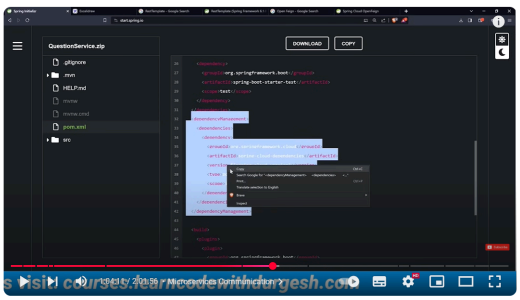
Which service to call, there add the OpenFing library.

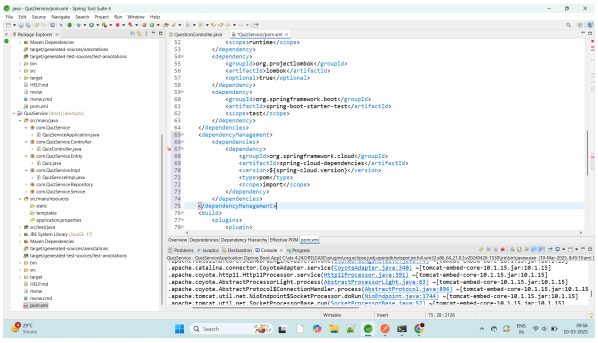




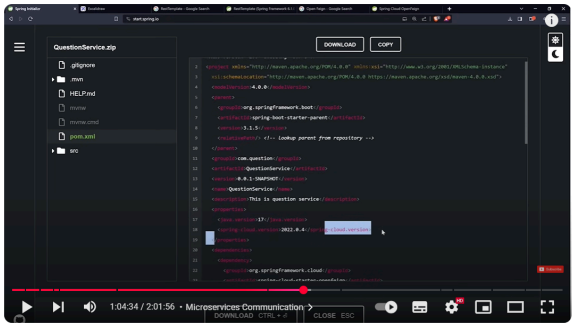
Not put directly openFIng dependency

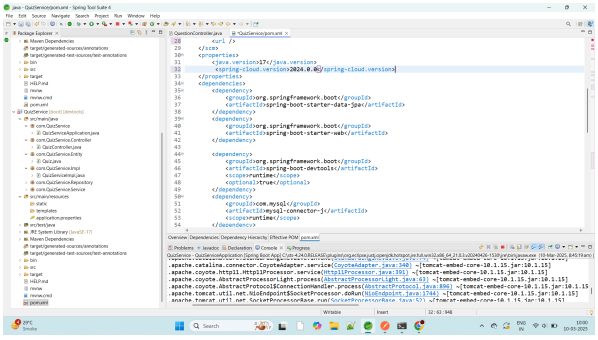
- first open the project (explore) then copy the dependency management
- copy dependency management





- copy property version



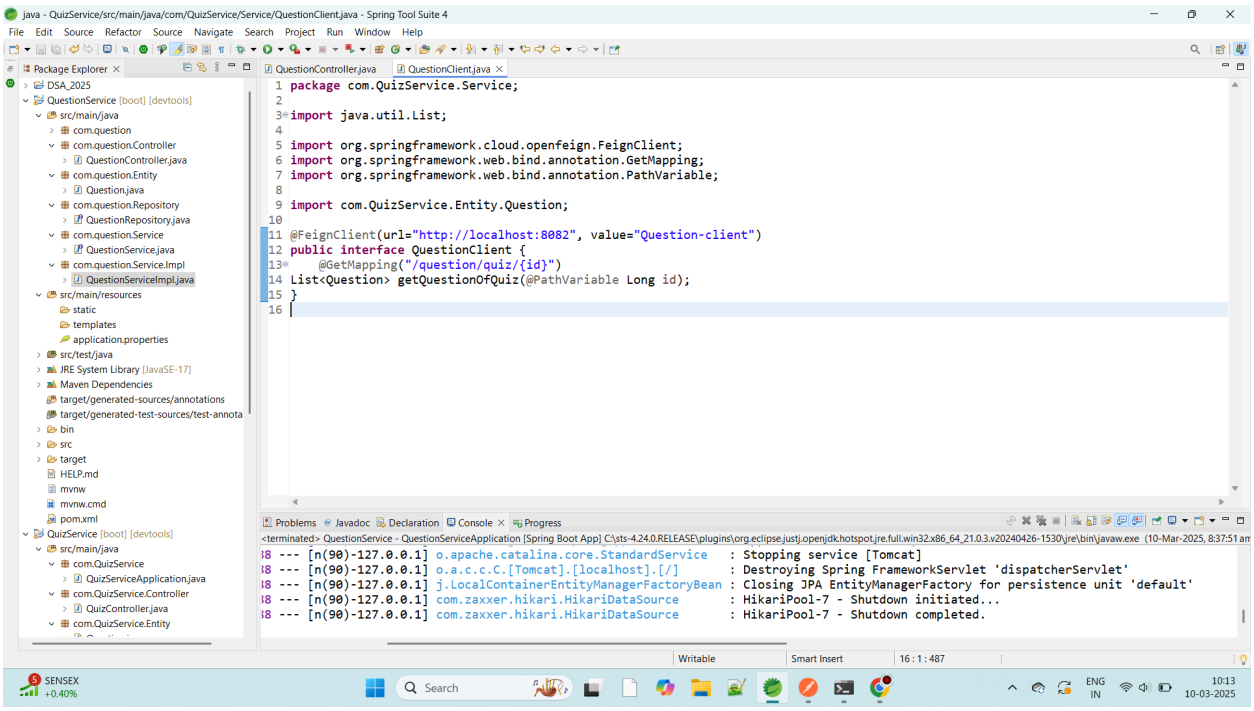


- Now copy the dependency and paste

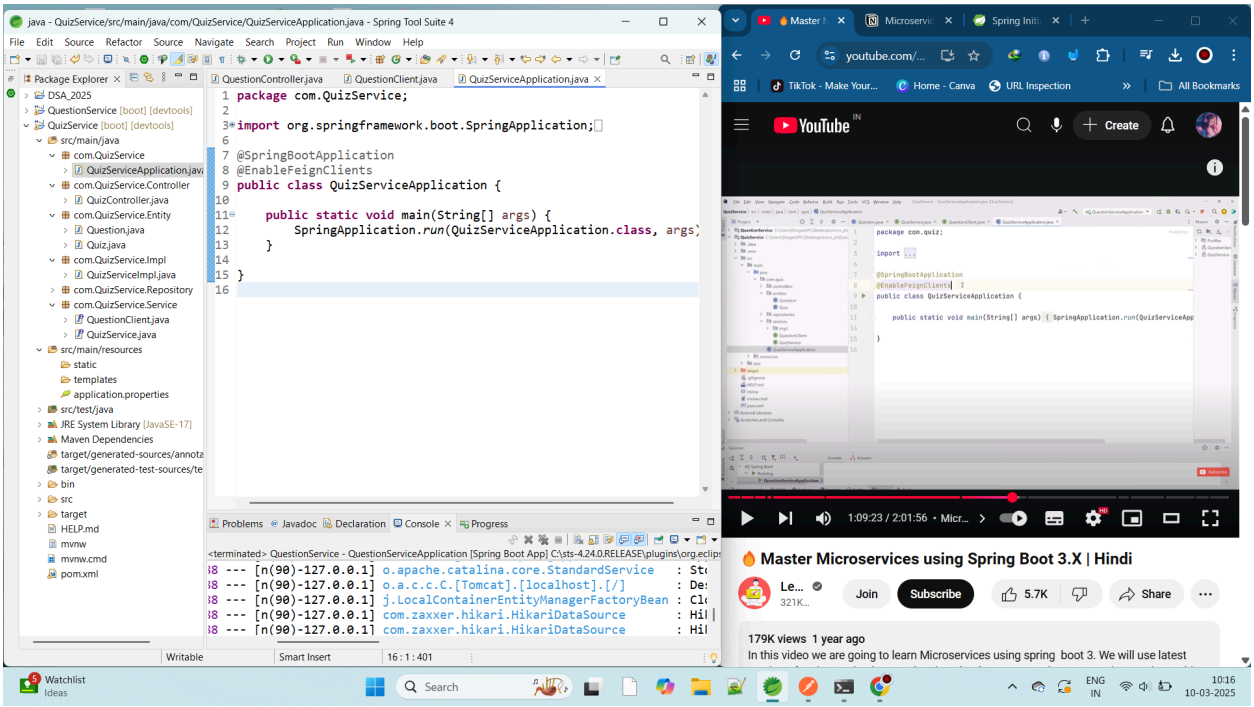
DOWNLOAD COPY

```
<properties>
  <java.version>17</java.version>
  <spring-cloud.version>2024.0.0</spring-cloud.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
  </dependency>

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

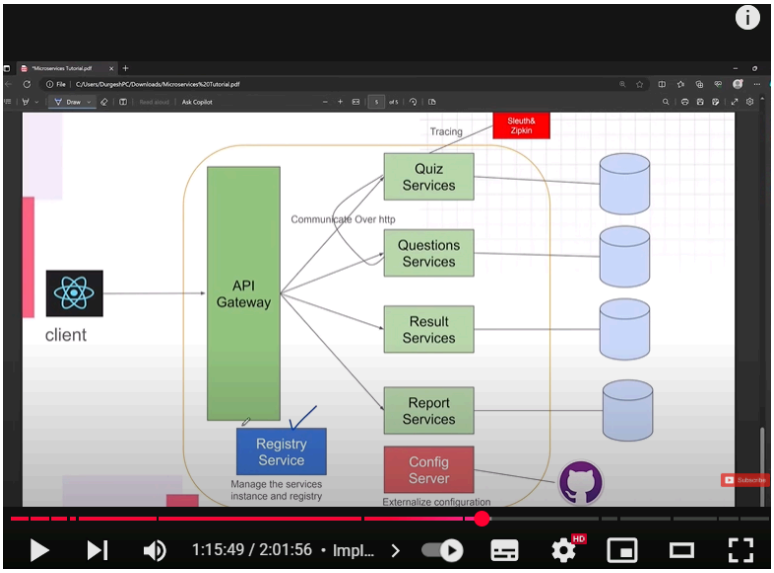
- Create a service and use annotation of fing



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

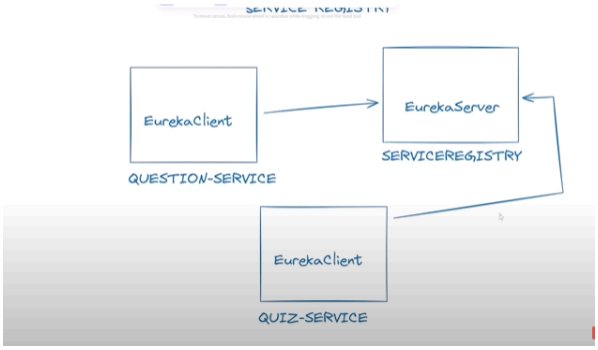


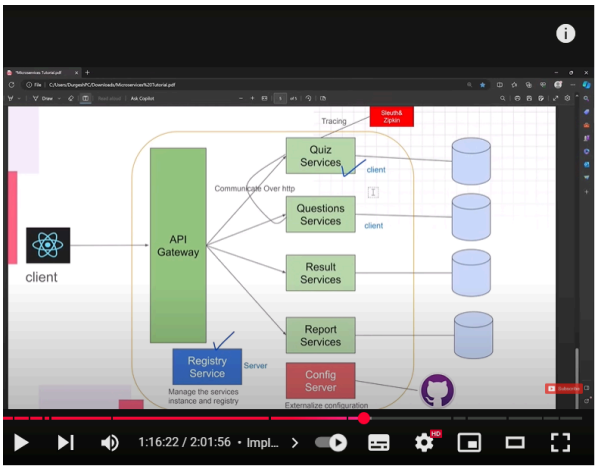
Service Registry or Discovery Service



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

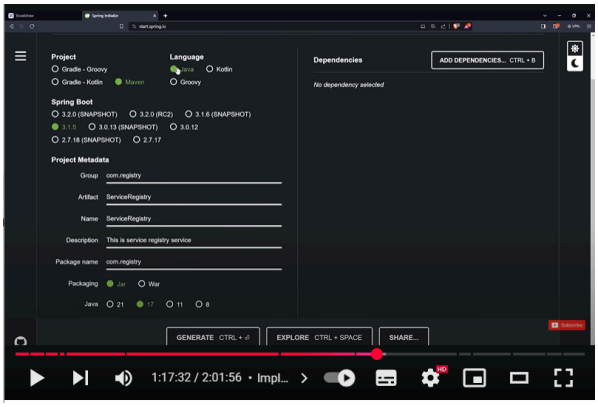


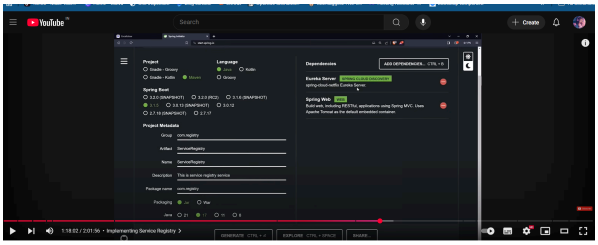


- **Disable the serverRegistery as a client**
- **Enable the all server as a client**

Implementation

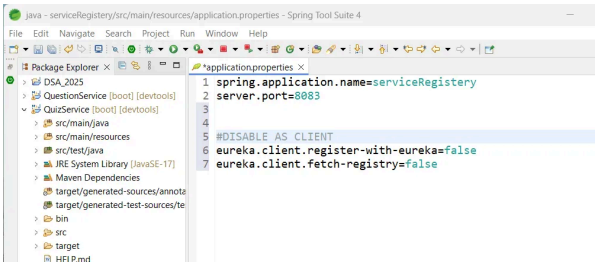
- create project
- add dependency

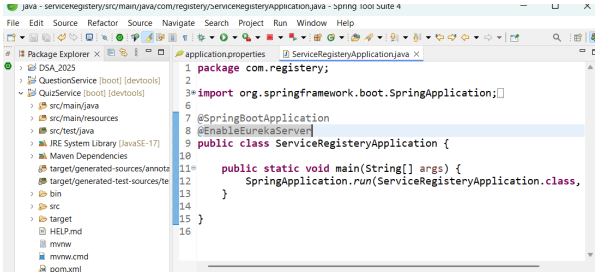




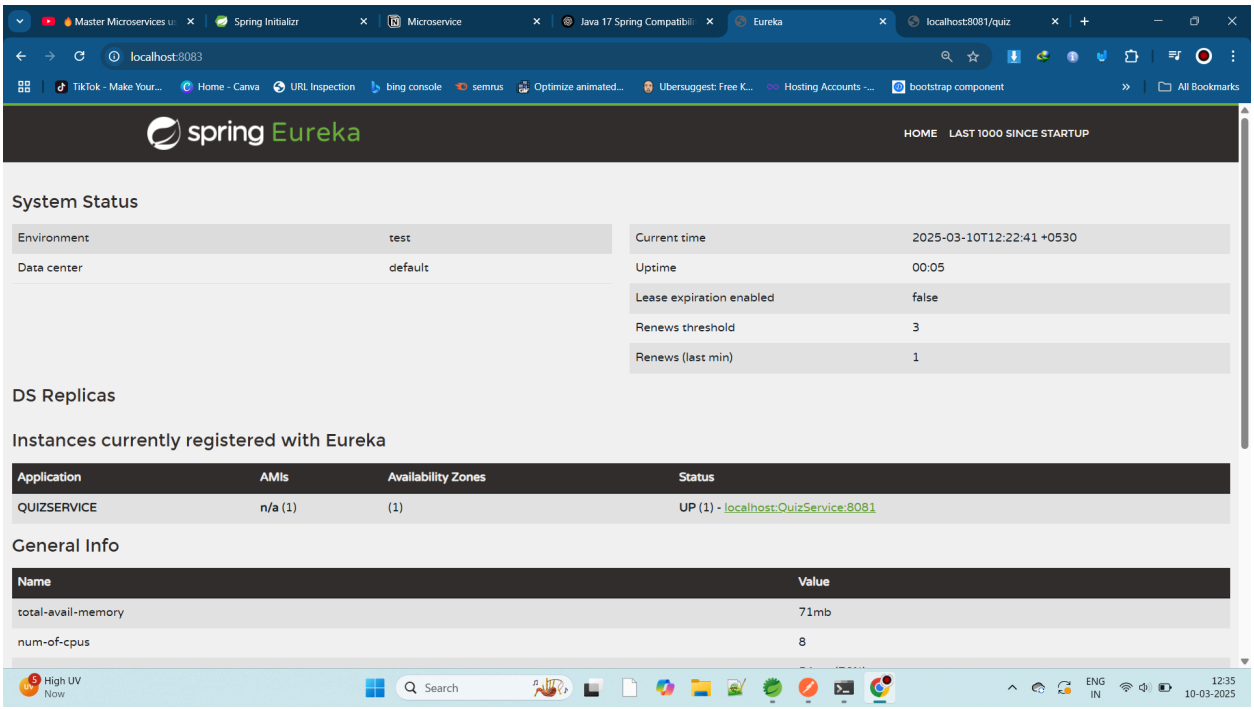
Registry Service -server

- Disable the serviceRegistry as a client





check Eureka server are start or not



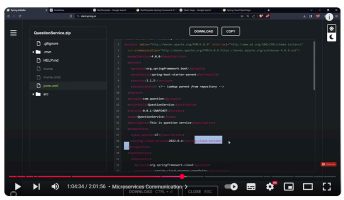
Registry Service - Client

Enable the other Service as a client

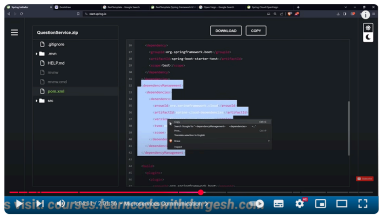
- First Search dependency and open project and copy all things and then dependency

i. past dependencymanager

II. past the version

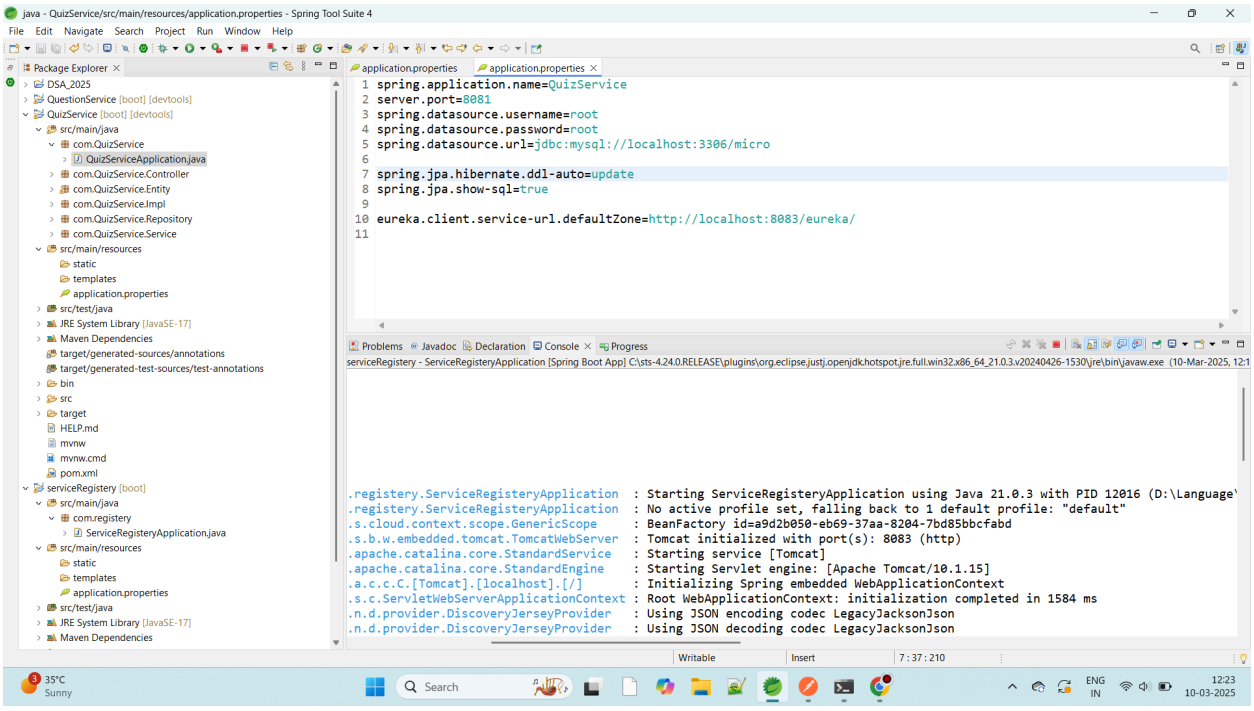


iii. add a Dependency eureka client in the other server

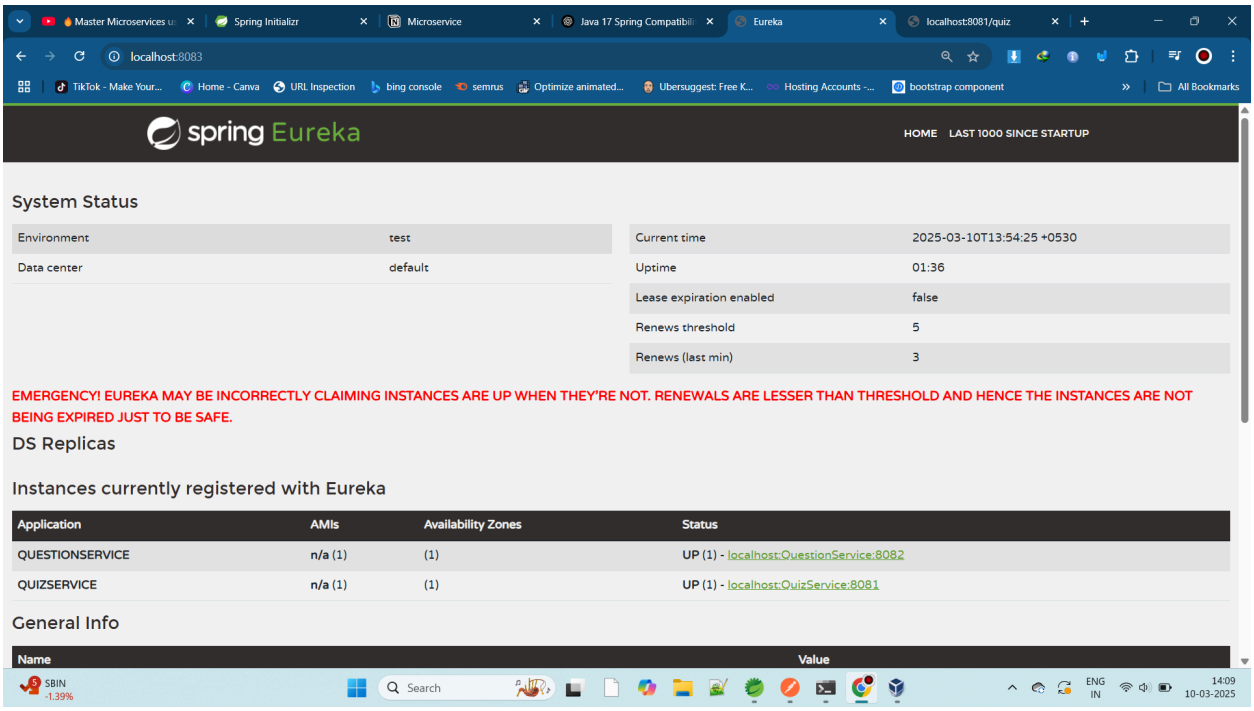


4th step

- 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



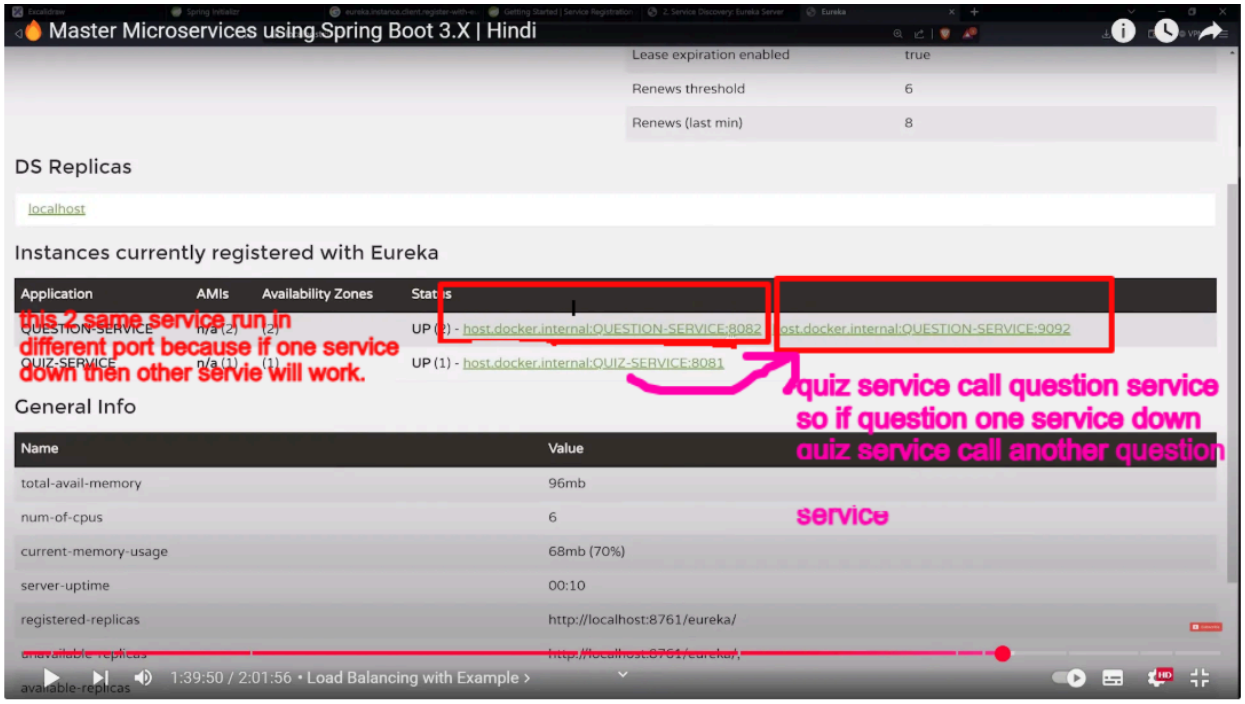
- Run all the client server Eureka Auto Handle all that instance.



Load Balancing or Auto Balancing

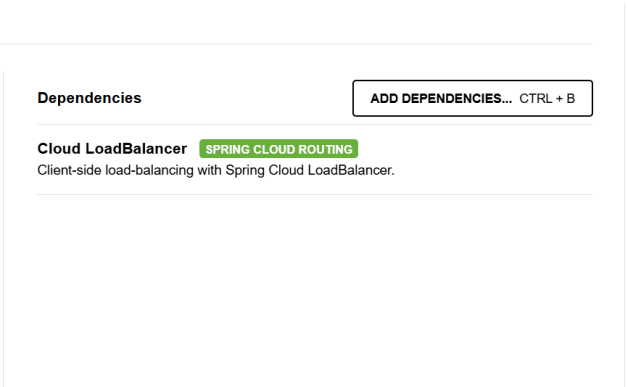
- means we have multiple the same service when one service goes down, then automatically call another service.
- When 2 or more of the same service are running, then Eureka also handles that.
- Eureka automatic handles all things, only put the dependency and one-time configure.

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



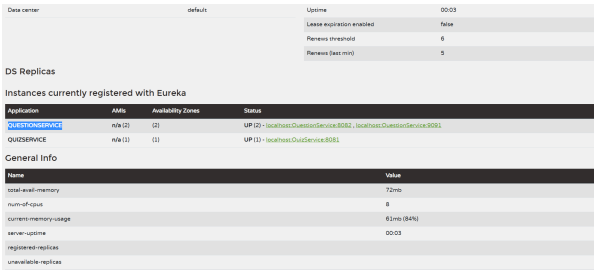
Add the dependency

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



copy name of service from Eureka

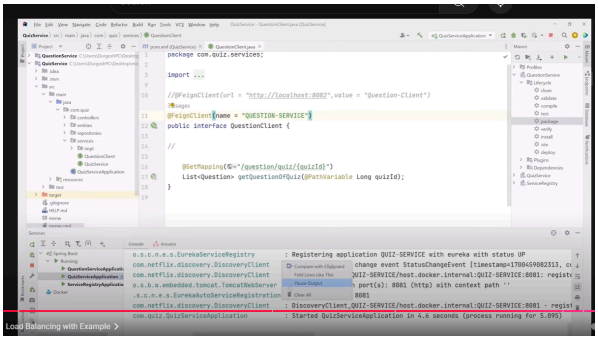
- we have multiple the same service so we not need do call with the URL.
- multiple same service like question



- 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

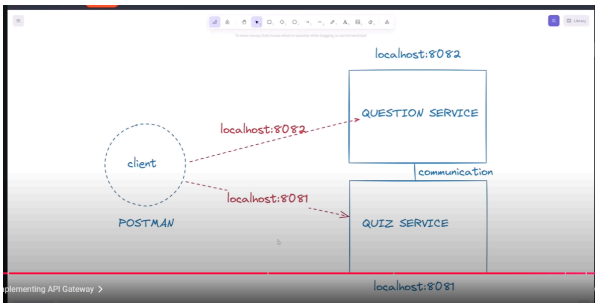
- We have multiple URL, when we want to call then call on that

solution

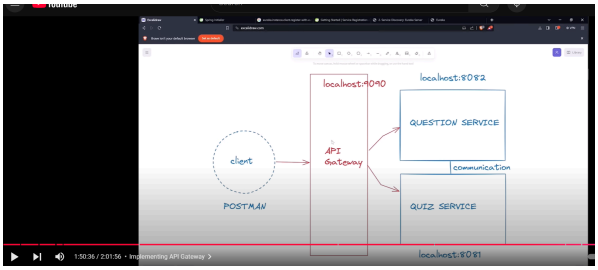
particular URL.

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.



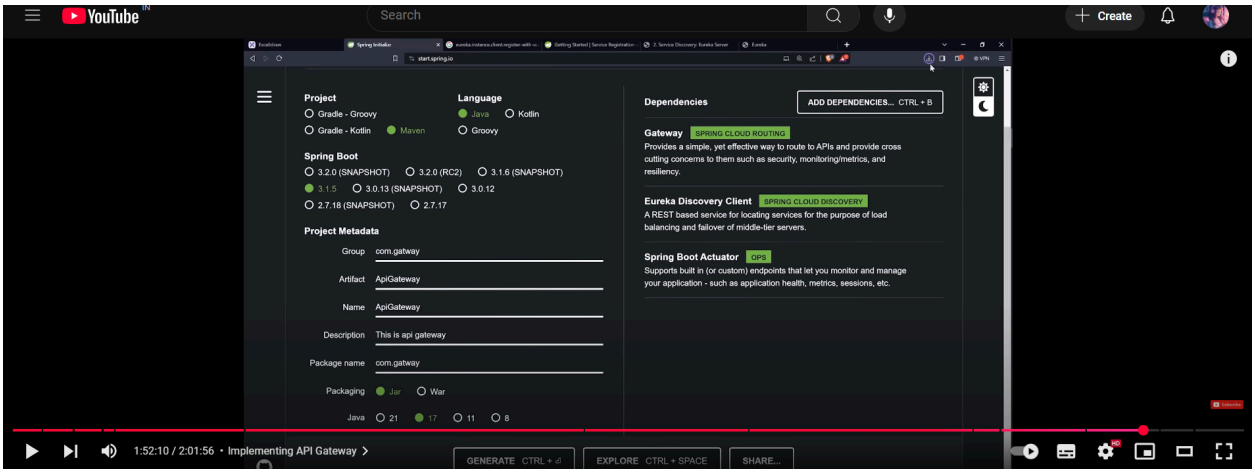
- Api Gateway is also a services



Implementation

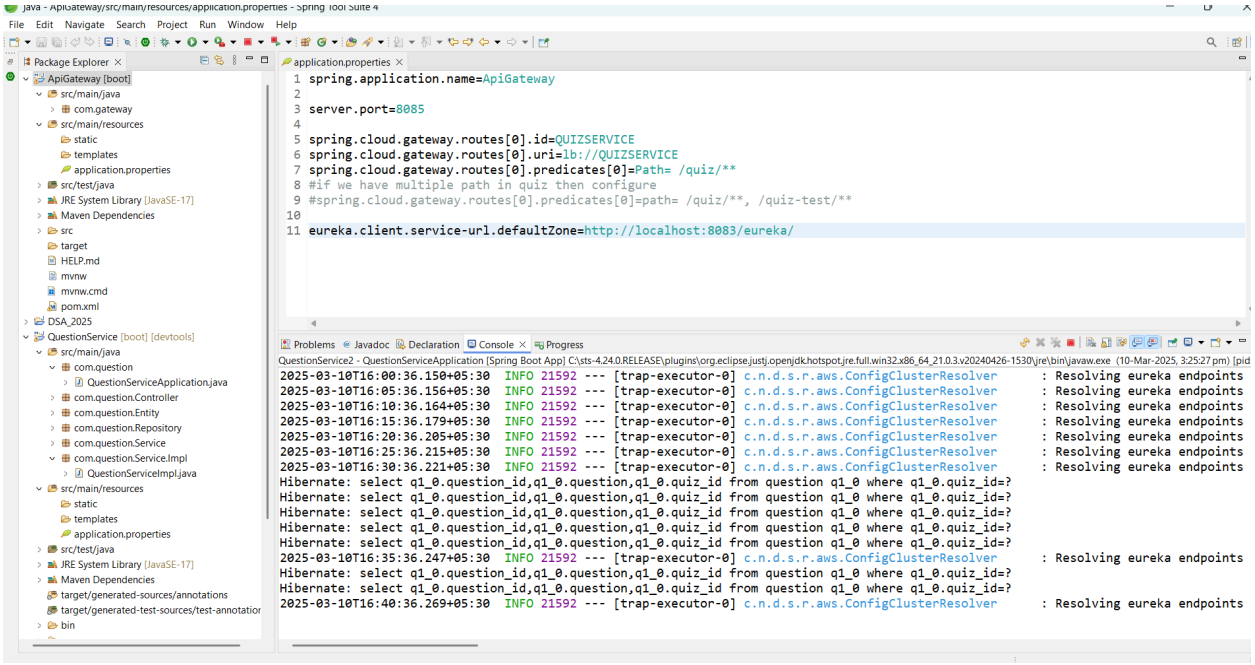
- In the Api Gateway only URL configuration, which service runs when

- **creating a project**



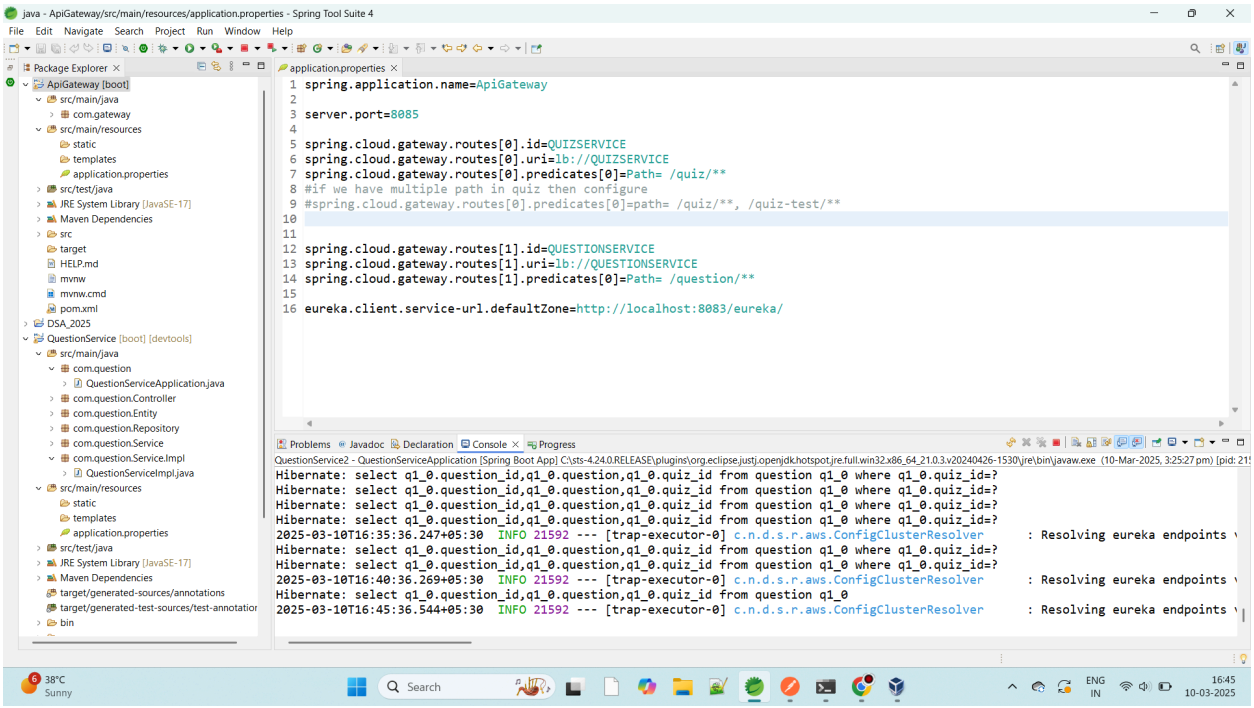
- In Gateway dependency, remove **(- MVC)** other not supported.
- dependency

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



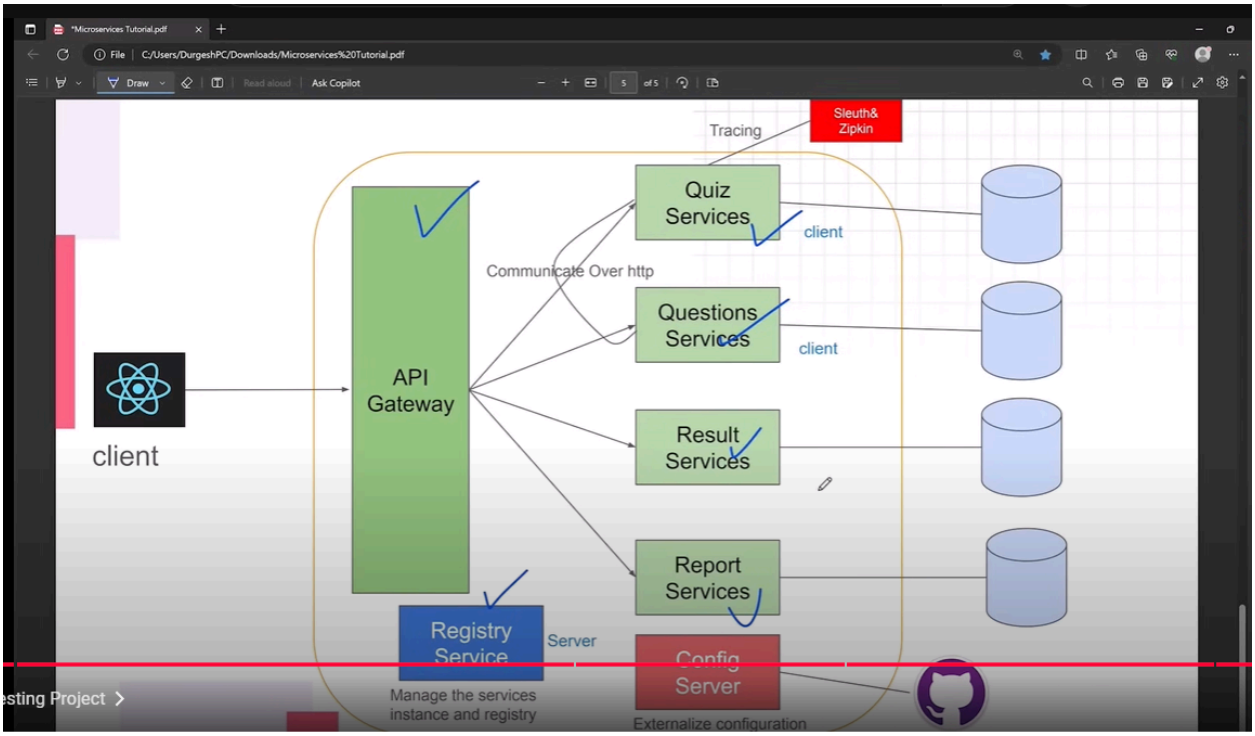
spring.application.name=ApiGateway
server.port=8085

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/



Remaining

- sleuth & zipkin
- configure server


Tracing
Sleuth& Zipkin
Quiz Services
client
Communicate Over http
Questions Services
client
API Gateway
client
Result Services
Report Services
Registry Service
Server
Config Server
Manage the services instance and registry
Externalize configuration