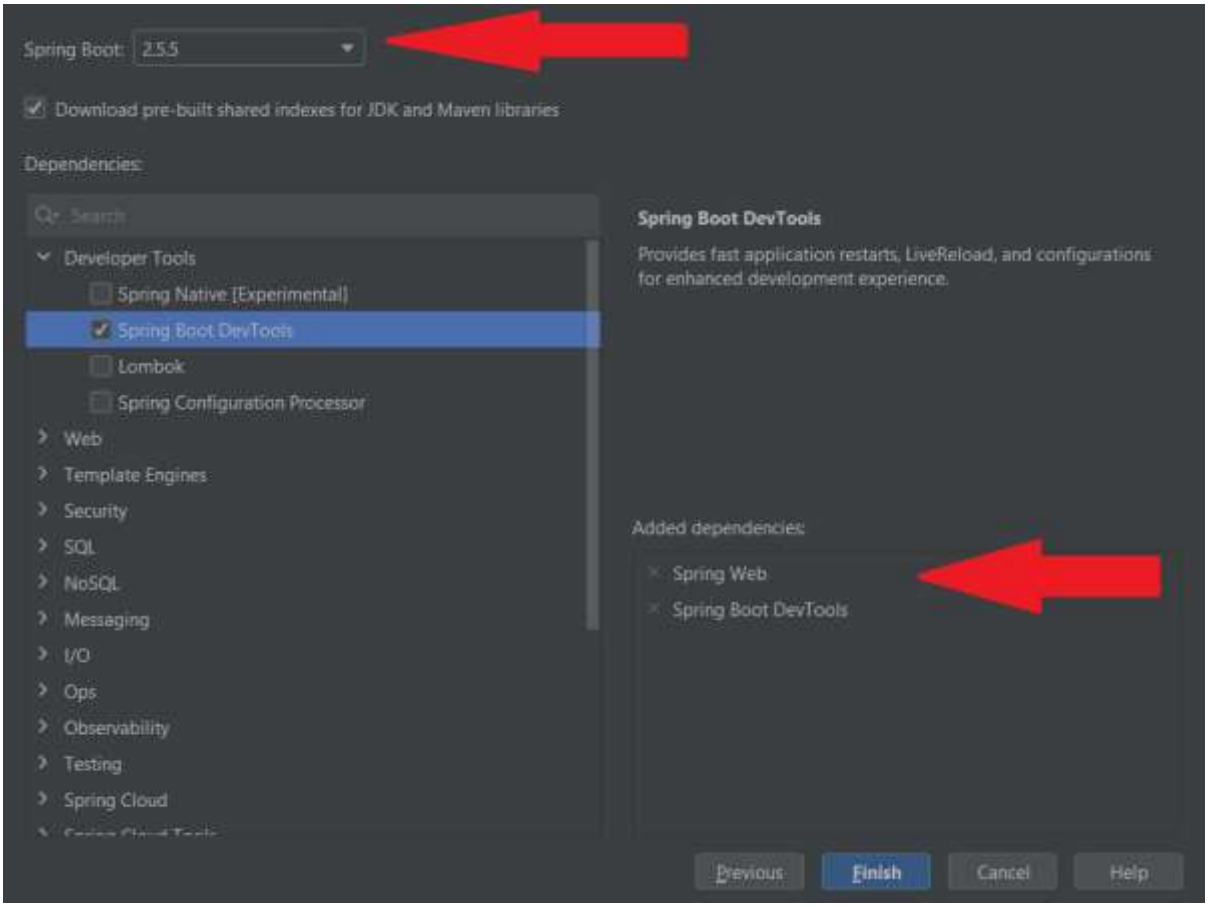
Handson-2

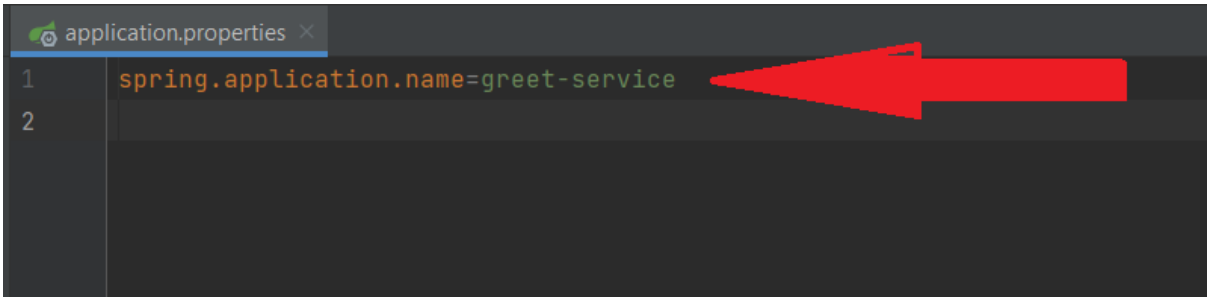
Create a Spring Cloud API Gateway and call one microservice thru the API gateway. Configure a global filter to log each request targeting the microservice using Spring Cloud API Gateway.

Steps.

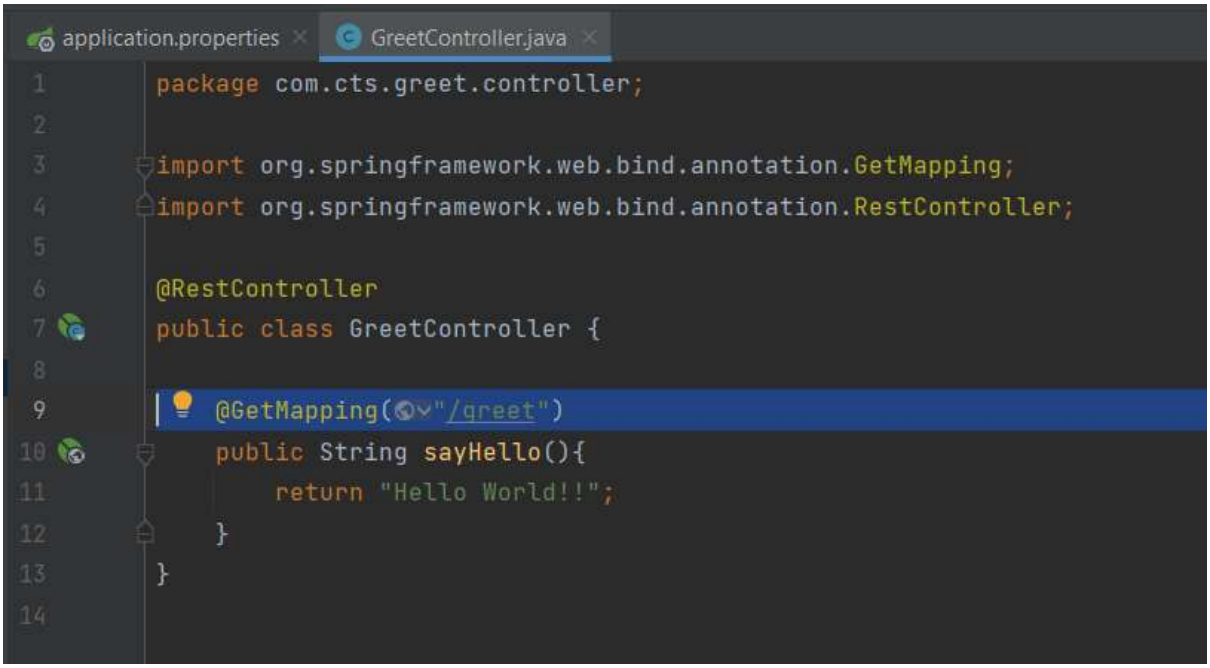
1. Create Simple Microservice **greet-service** that returns "Hello World" using Spring Initializer.
2. Select the latest version of Spring Boot and the dependencies as shown in the image below.



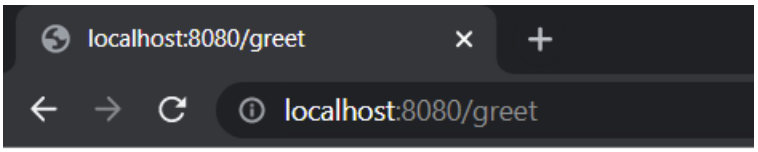
3. Configure the application name in "application.properties" as shown below



4. Create a controller as shown below.

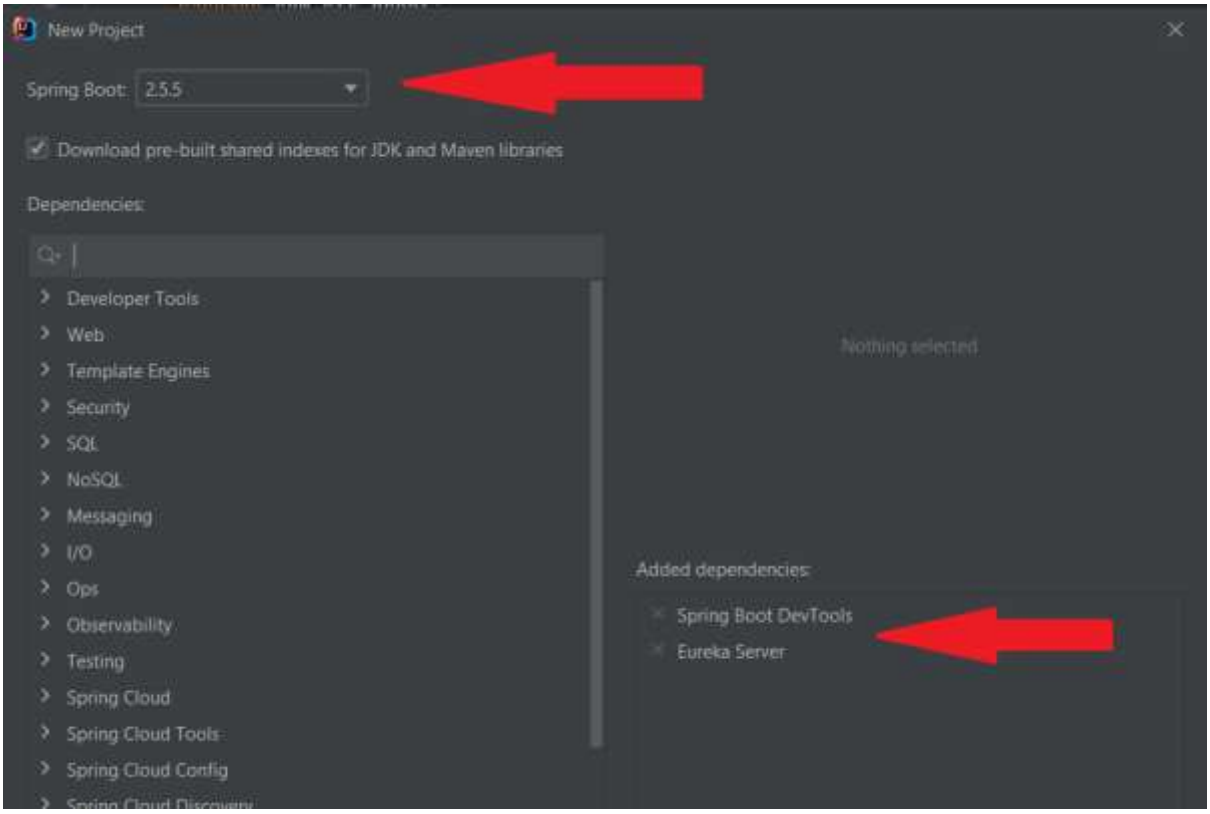


5. Run the microservice and make sure that it is working fine as shown below



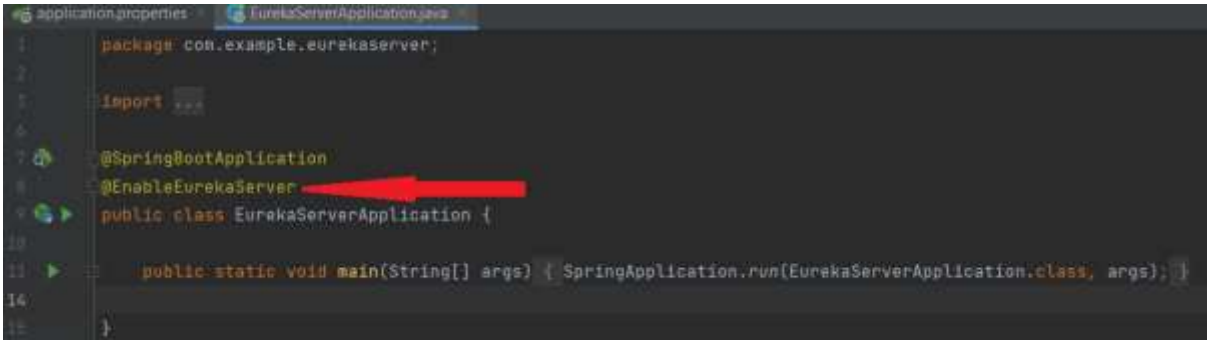
Hello World!!

6. Create a second microservice that is acting as the Discovery Server using Spring Cloud Eureka Server.
7. Select the latest version of Spring Boot and the required dependencies as shown below



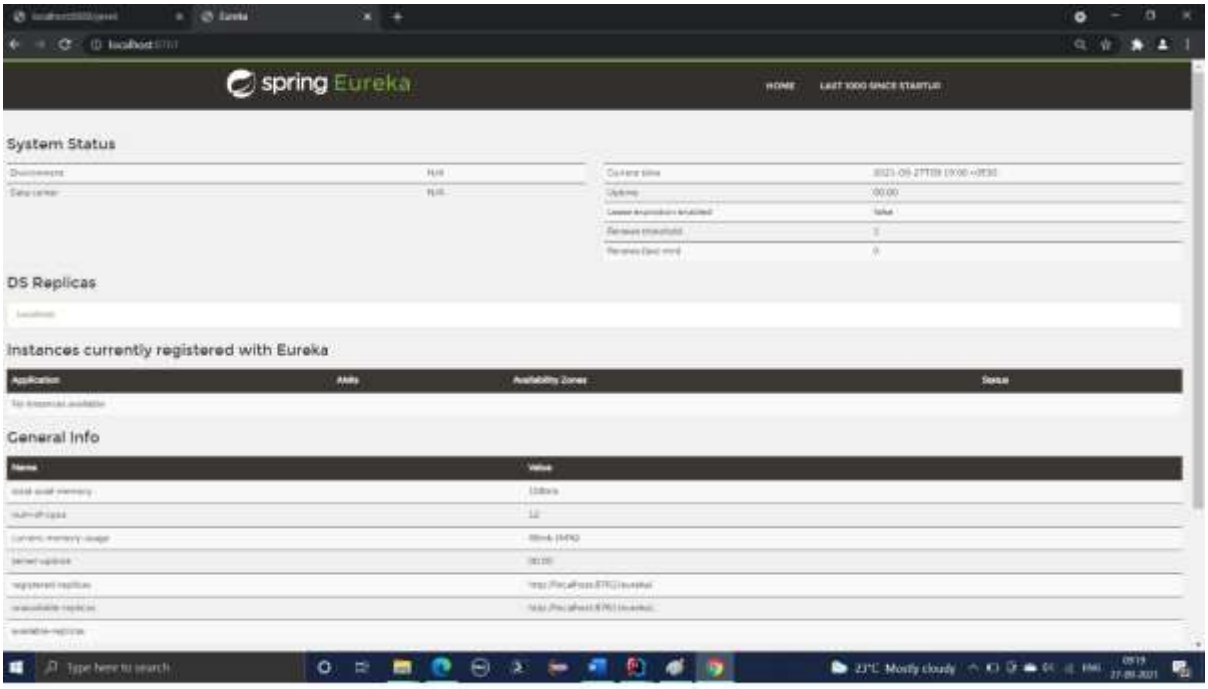
8. Include the following configurations in the "**application.properties**" of the Eureka server

9. Annotate the main class in Eureka Server with **@EnableEurekaServer** as shown below.

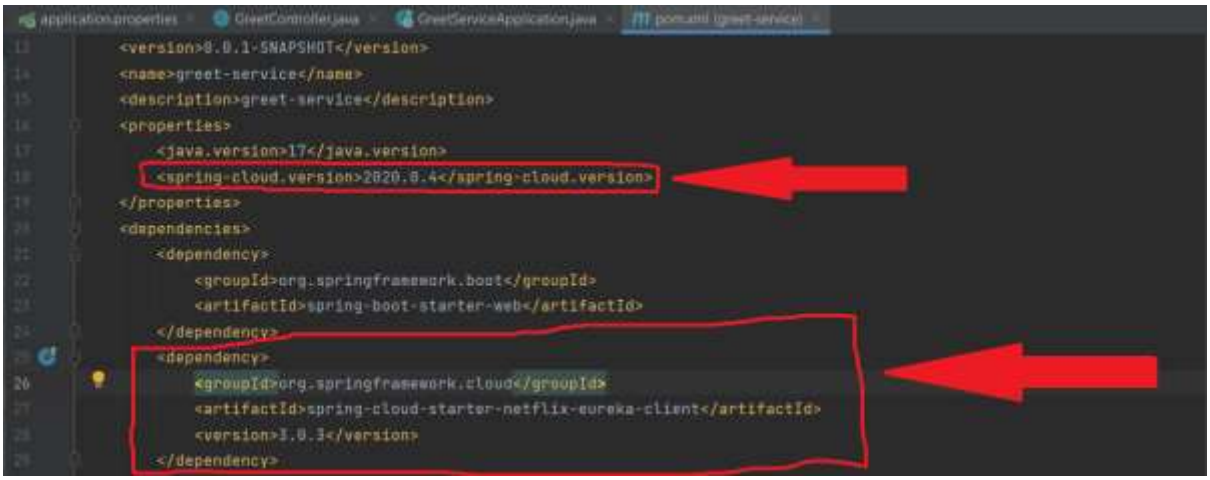


10. Run the Eureka Server and ensure that the server is functioning properly by entering the below URL at any of the browser.

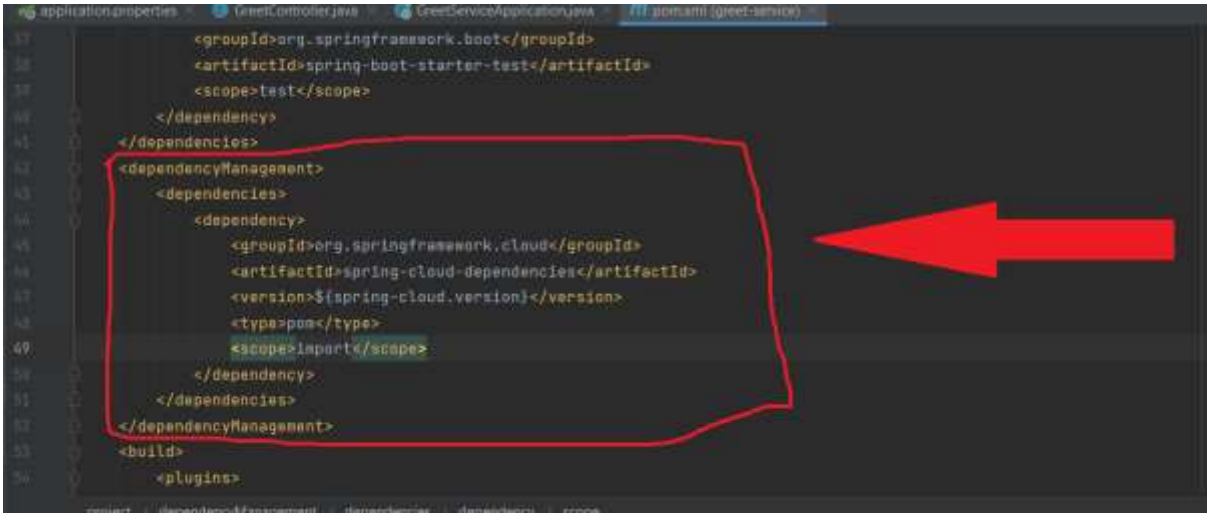
    http://localhost:8761



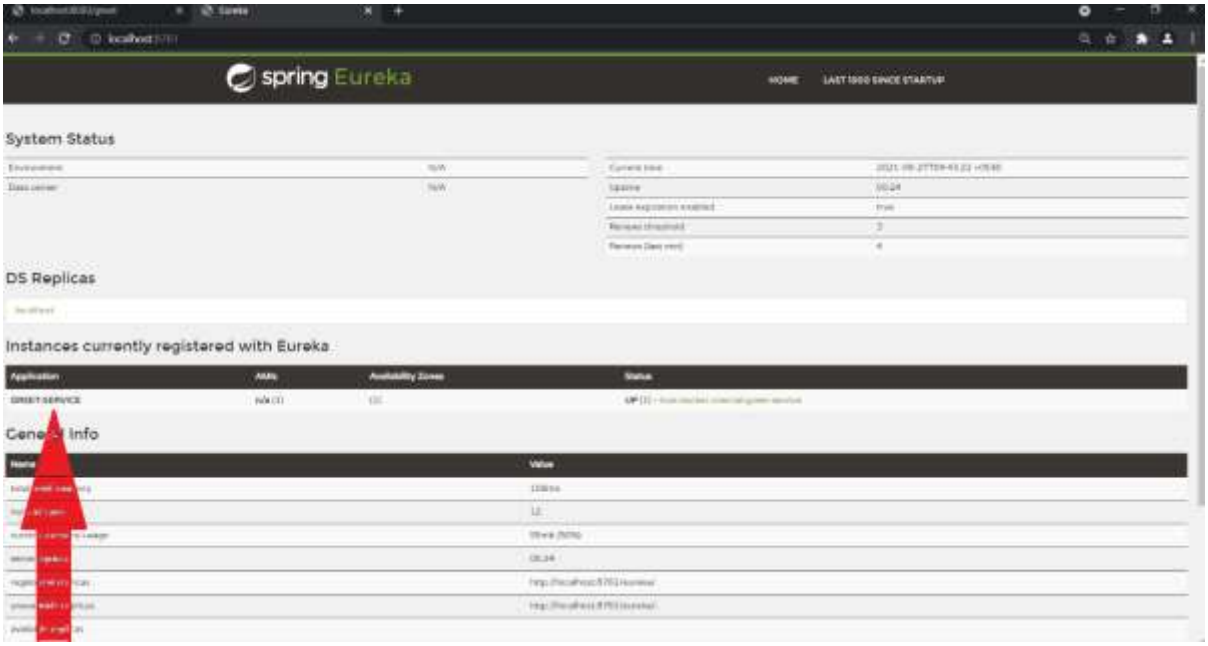
11. Open the pom.xml of greet-service microservice and add the eureka client dependencies as shown in the image below.
12. Copy the spring cloud version from pom.xml of eureka-server and paste it at the appropriate location in the pom.xml of greet-service as shown below.



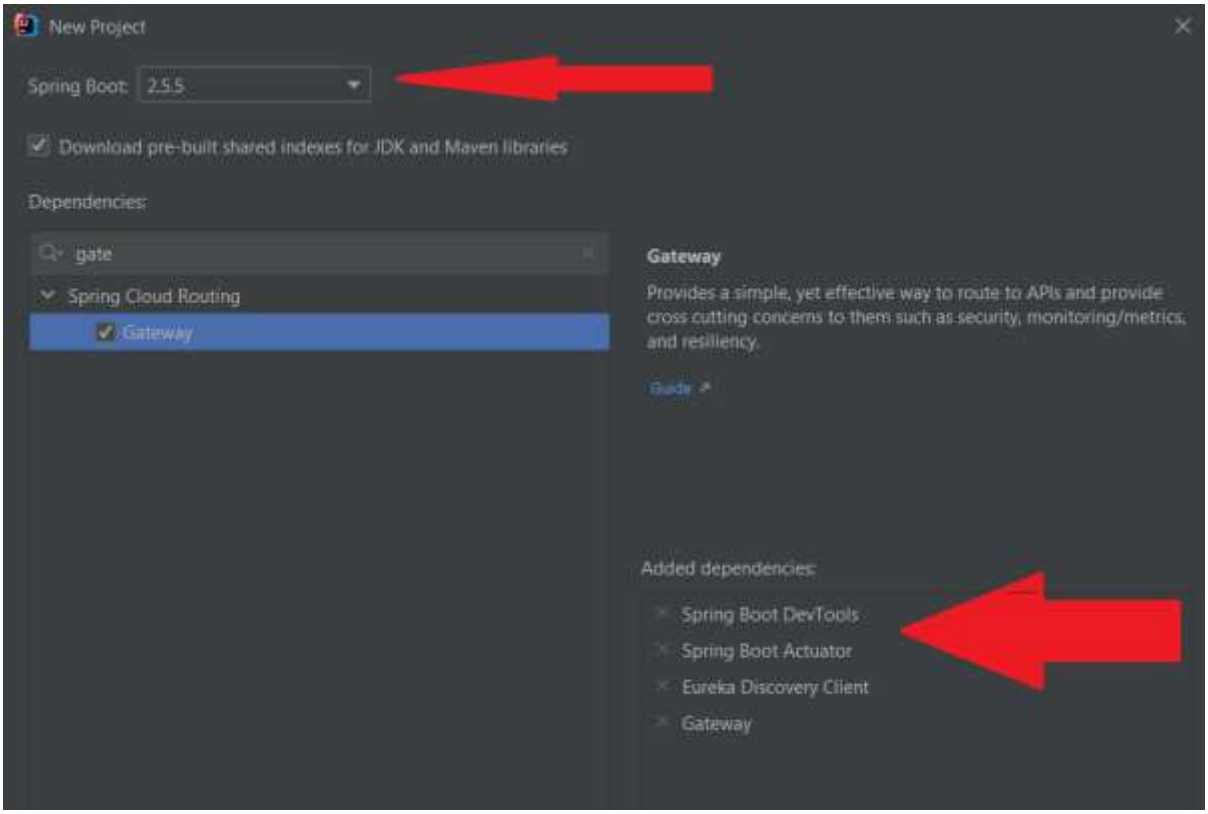
13. Copy the "dependency-management" session in the pom.xml of the eureka-server and paste it immediately after the dependencies session of the pom.xml of the greet-service as shown below.



14. Restart greet-service and refresh the http://localhost:8761 to see whether the name of the greet-service figuring in the eureka-server console as shown below.

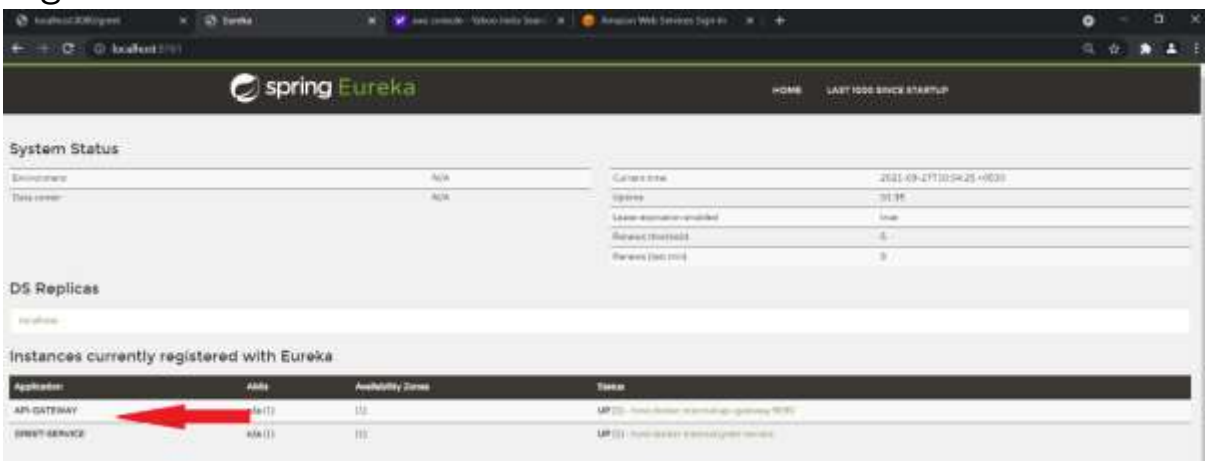


15. Create another microservices “api-gateway”
16. Select the latest version of Spring Boot and add the required dependencies as shown below.



17. Make the necessary configurations in the "**application.properties**" file as shown below.

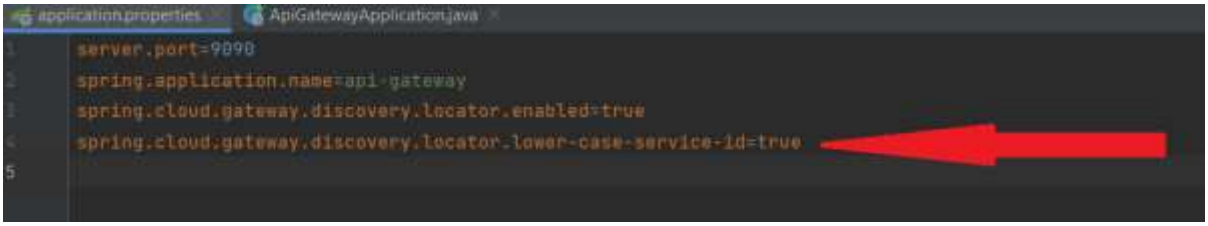
18. Run the api-gateway service and check if it is getting registered with the eureka-server.



19. Try to access the following URL from any of the browser and see if you are able to access the greet-service thru the api-gateway.

http://localhost:9090/GREET-SERVICE//greet

20. Include the following configuration in the " application.properties" of the api-gateway to specify the service name in lower case.

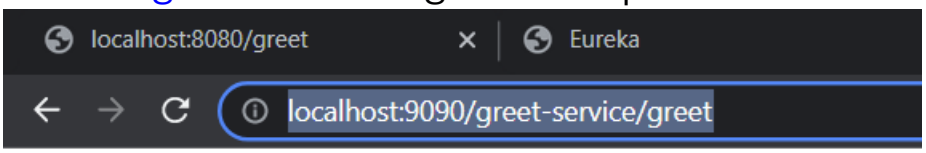


21. Implement a global filter which logs all incoming requests.
    a. Create a LogFilter class as shown below.

application.properties | LogFilter.java | ApiGatewayApplication.java

```java
package com.cts.gateway.filters;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.cloud.gateway.filter.GatewayFilterChain;
import org.springframework.cloud.gateway.filter.GlobalFilter;
import org.springframework.stereotype.Component;
import org.springframework.web.server.ServerWebExchange;
import reactor.core.publisher.Mono;
@Component
public class LogFilter implements GlobalFilter {
    Logger logger= LoggerFactory.getLogger(LogFilter.class);
    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
        logger.info("====>Request URL {}",exchange.getRequest().getURI());
        return chain.filter(exchange);
    }
}
```

22. Try to access the url http://localhost:9090/greet-service/greet. You will get the output as shown below.



Hello World!!

23.