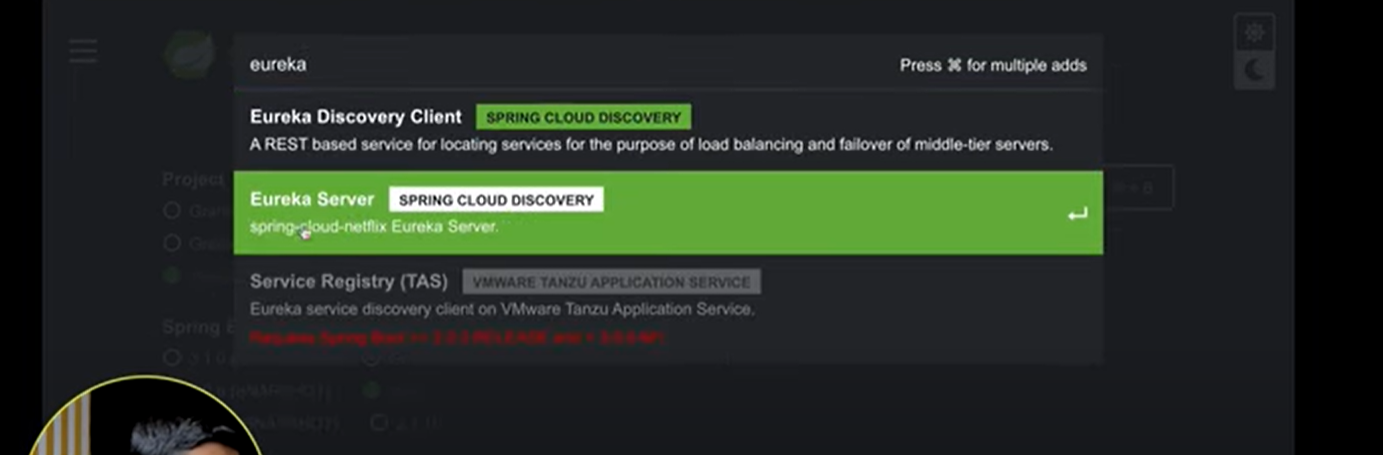
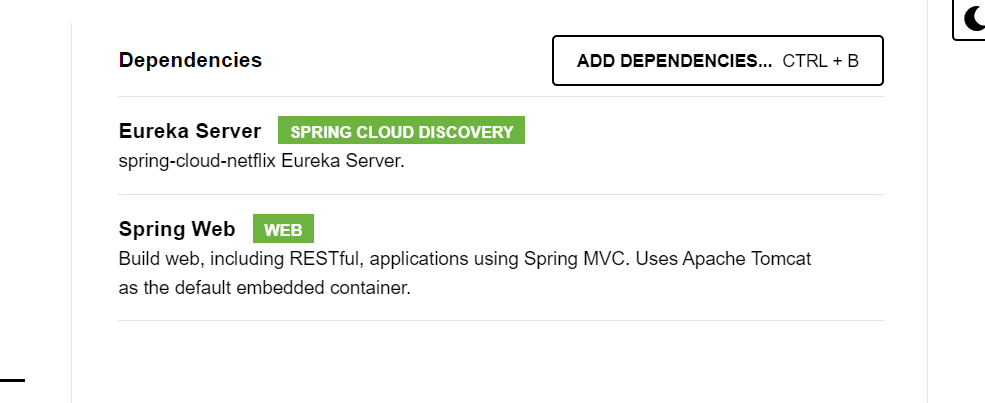


Service Registry :

We use eureka server :



**Also We need Web Dependency :**



Eureka Server is an application that holds the information about all client-service applications. Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

Building a Eureka Server

Eureka Server comes with the bundle of Spring Cloud. For this, we need to develop the Eureka server and run it on the default port 8761.

The @EnableEurekaServer annotation is used to make your Spring Boot application acts as a Eureka Server.

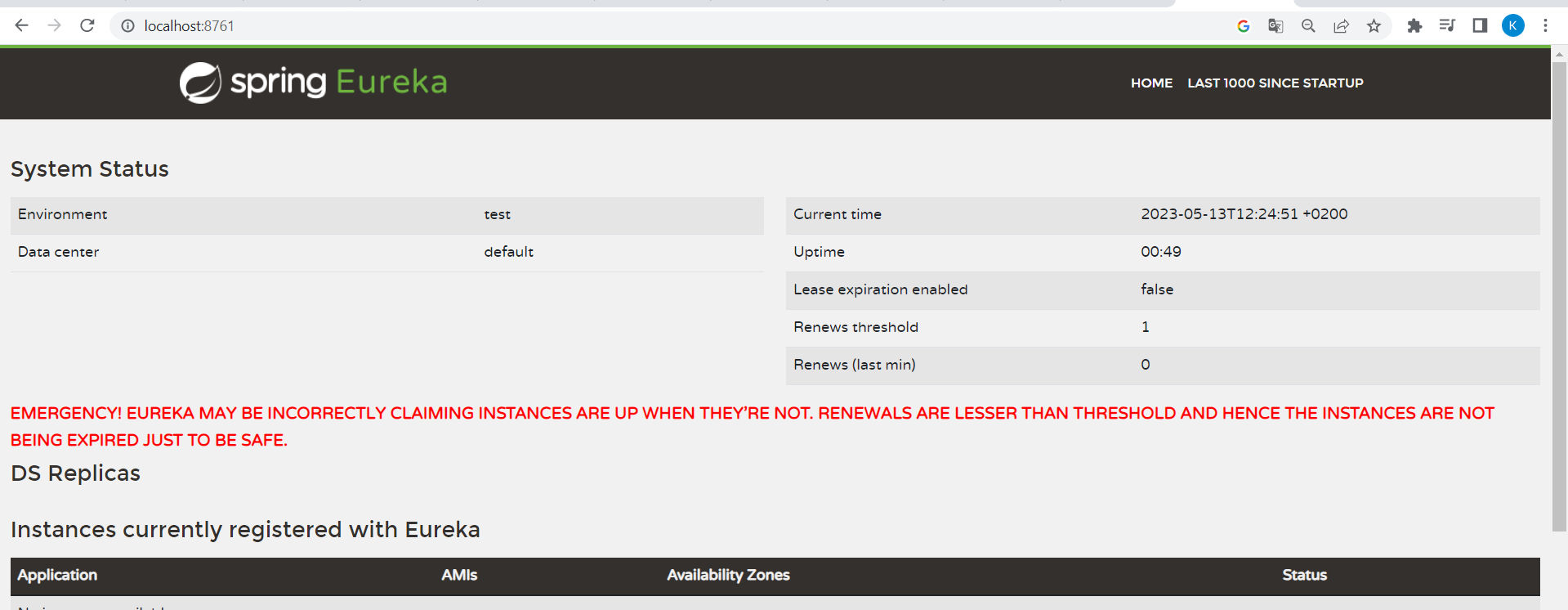
package com.tutobiq.serviceregistry;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class ServiceRegistryApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ServiceRegistryApplication.class, args);  
 }  
  
}

The code for Maven user dependency is shown below −

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
</dependency>

By default, the Eureka Server registers itself into the discovery. You should add the below given configuration into your application.properties file or application.yml file.

server.port=8761  
spring.application.name=service-registry  
  
eureka.client.registerWithEureka = false  
eureka.client.fetchRegistry = false

****

REGISTRATION EUREKA SERVICE

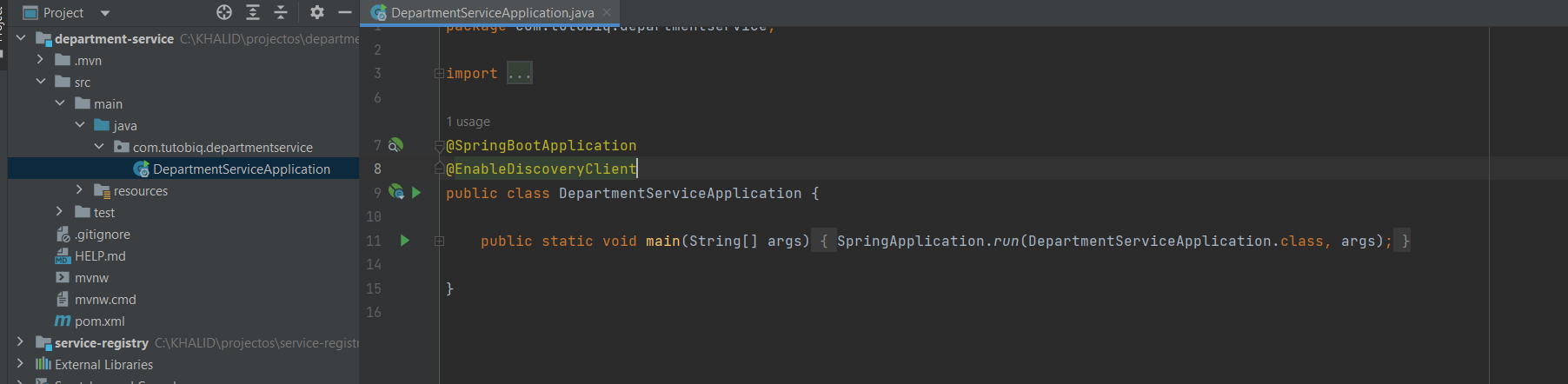
In this chapter, you are going to learn in detail about How to register the Spring Boot Micro service application into the Eureka Server. Before registering the application, please make sure Eureka Server is running on the port 8761 or first build the Eureka Server and run it. For further information on building the Eureka server, you can refer to the previous chapter.

First, you need to add the following dependencies in our build configuration file to register the microservice with the Eureka server.

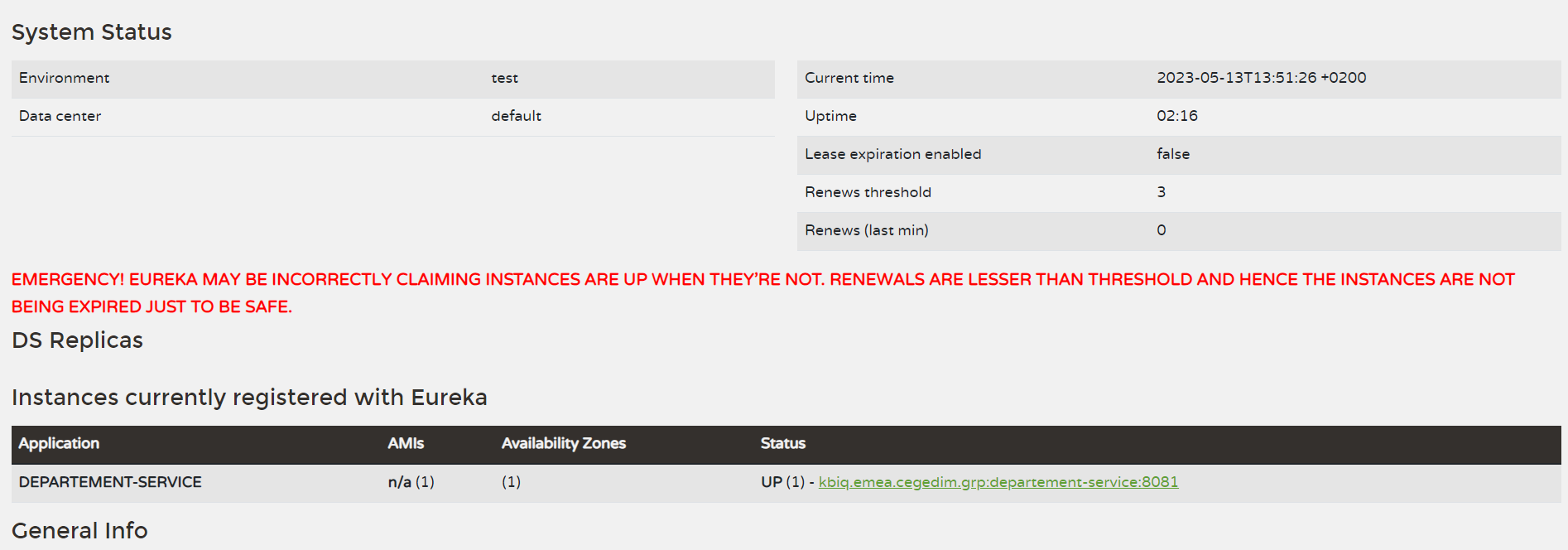
Maven users can add the following dependencies into the **pom.xml** file −

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>

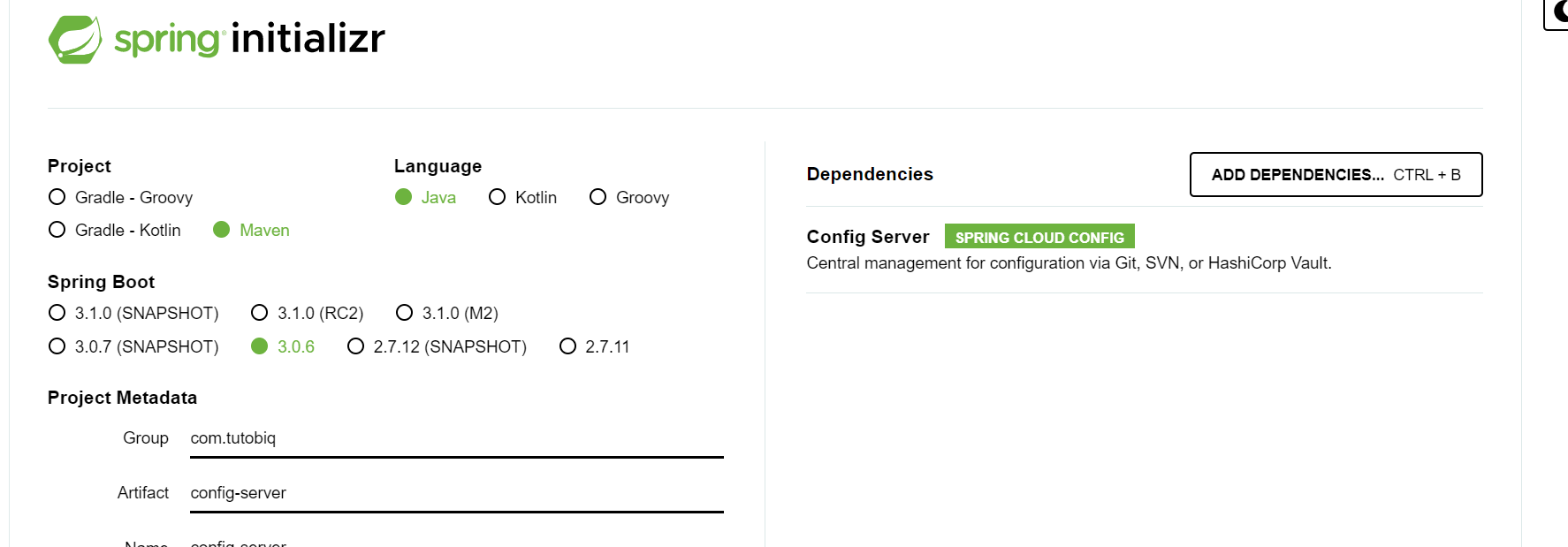
1 \_ Service Department Service :



server.port=8081  
eureka.client.serviceUrl.defaultZone = http://localhost:8761/eureka  
eureka.client.instance.preferIpAddress = true  
spring.application.name = eurekaclient



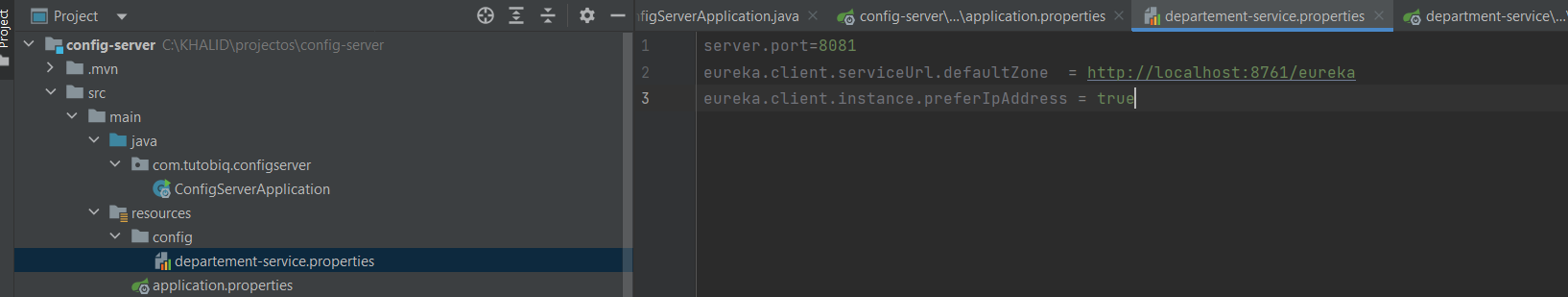
Config Server :



Spring Cloud Configuration Server is a centralized application that manages all the application related configuration properties. In this chapter, you will learn in detail about how to create Spring Cloud Configuration server.

server.port=8088  
spring.profiles.active=native

After i create file configuration for each service to externalsie configration of service



# **Spring Boot - Cloud Configuration Client**

Some applications may need configuration properties that may need a change and developers may need to take them down or restart the application to perform this. However, this might be lead to downtime in production and the need of restarting the application. Spring Cloud Configuration Server lets developers to load the new configuration properties without restarting the application and without any downtime.

So we had to add this config for all project like for download his config withot downtime

dependency into the pom.xml file.

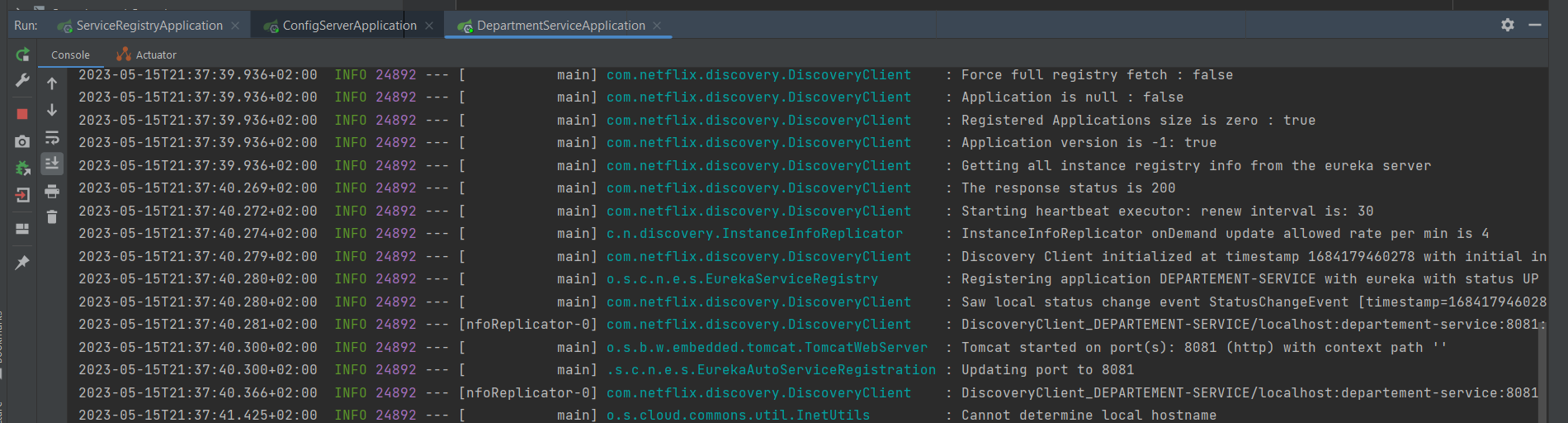
<dependency>

<groupId>org.springframework.cloud</groupId>

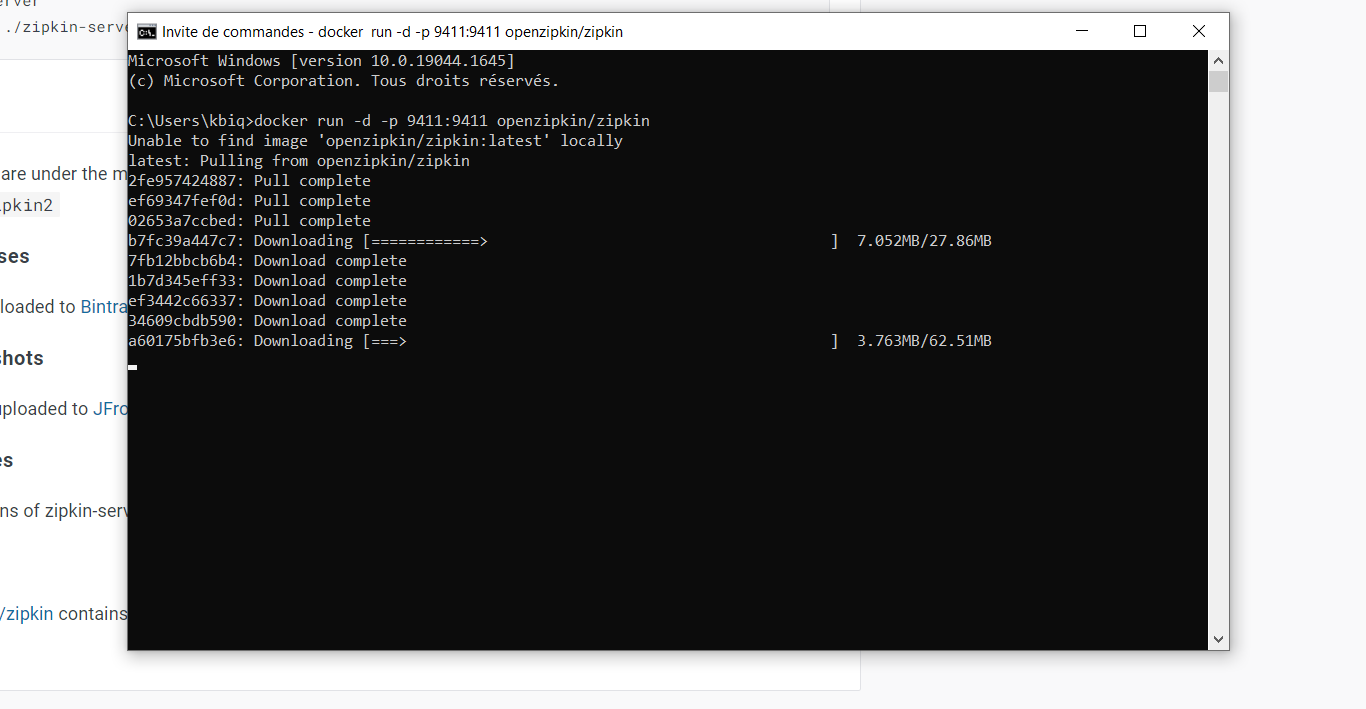
<artifactId>spring-cloud-starter-config</artifactId>

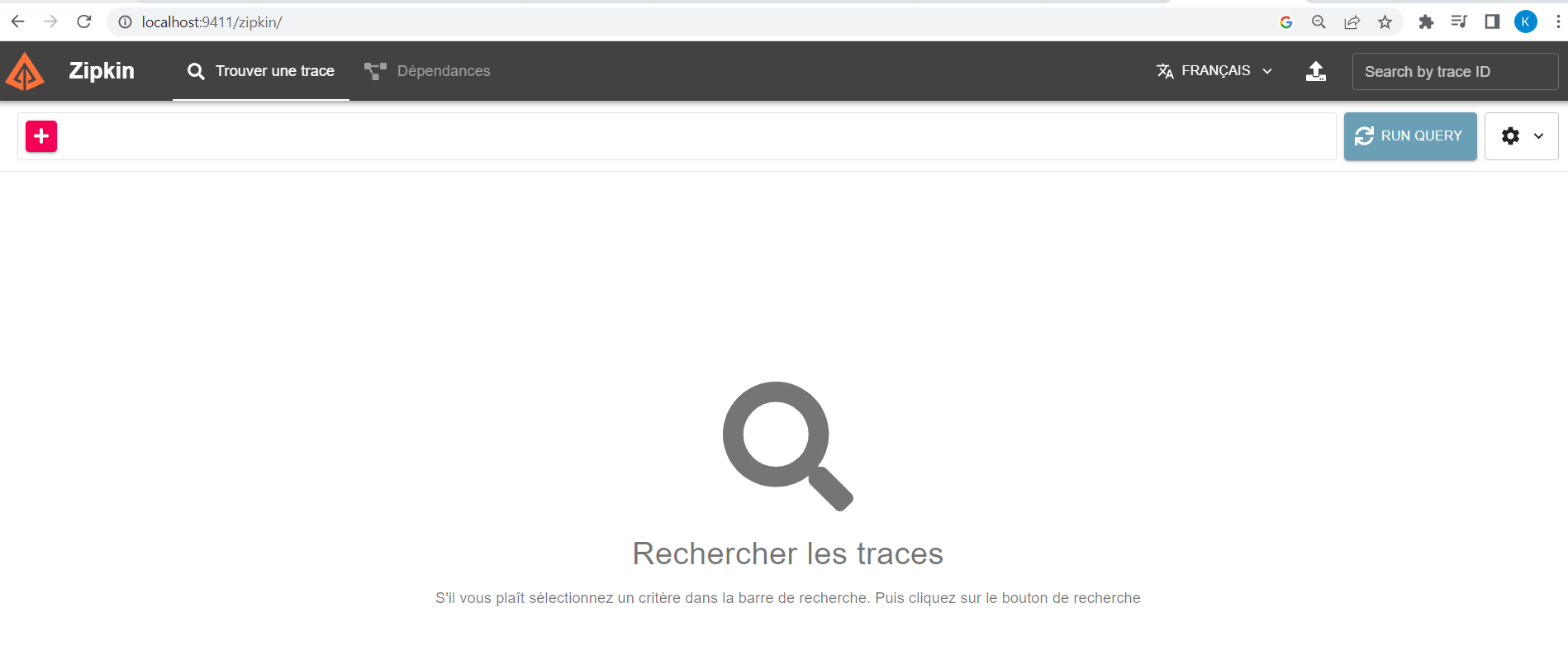
</dependency>

NEW CONFIG IS DOWNLOADED



Le serveur à télécharger l’application ses propres depuis config server :





Dans le projet

**<dependency>**

**<groupId>io.micrometer</groupId>**

**<artifactId>micrometer-tracing-bridge-brave</artifactId>**

**</dependency>**

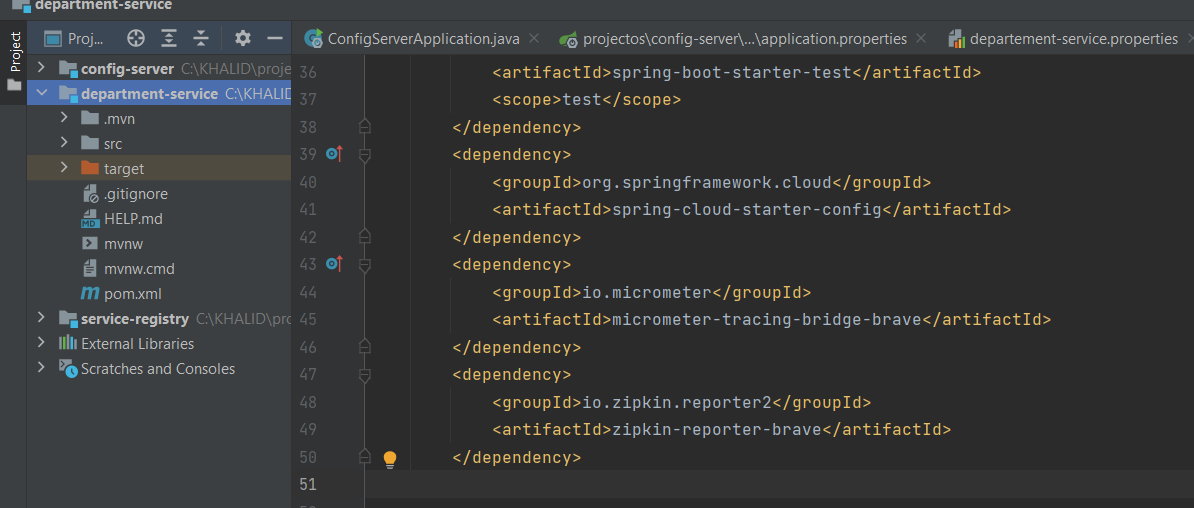
**<dependency>**

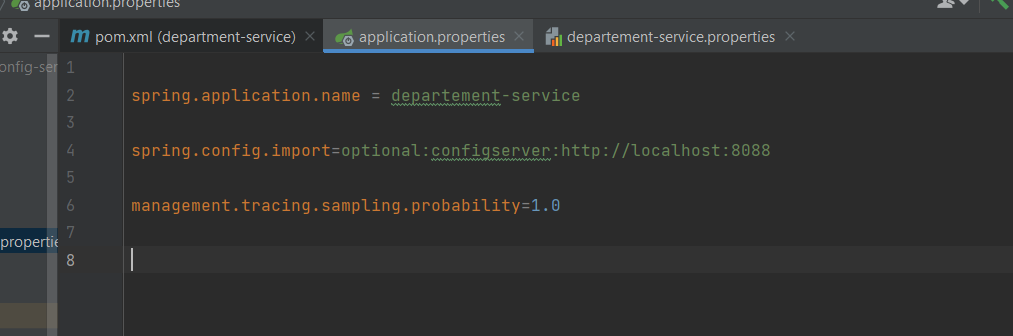
**<groupId>io.zipkin.reporter2</groupId>**

**<artifactId>zipkin-reporter-brave</artifactId>**

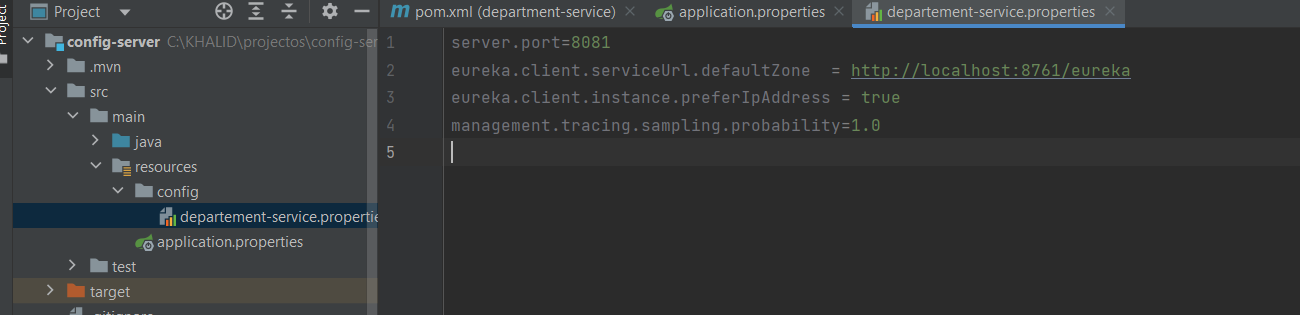
**</dependency>**

Je vais ajouter cette dépendance au client :

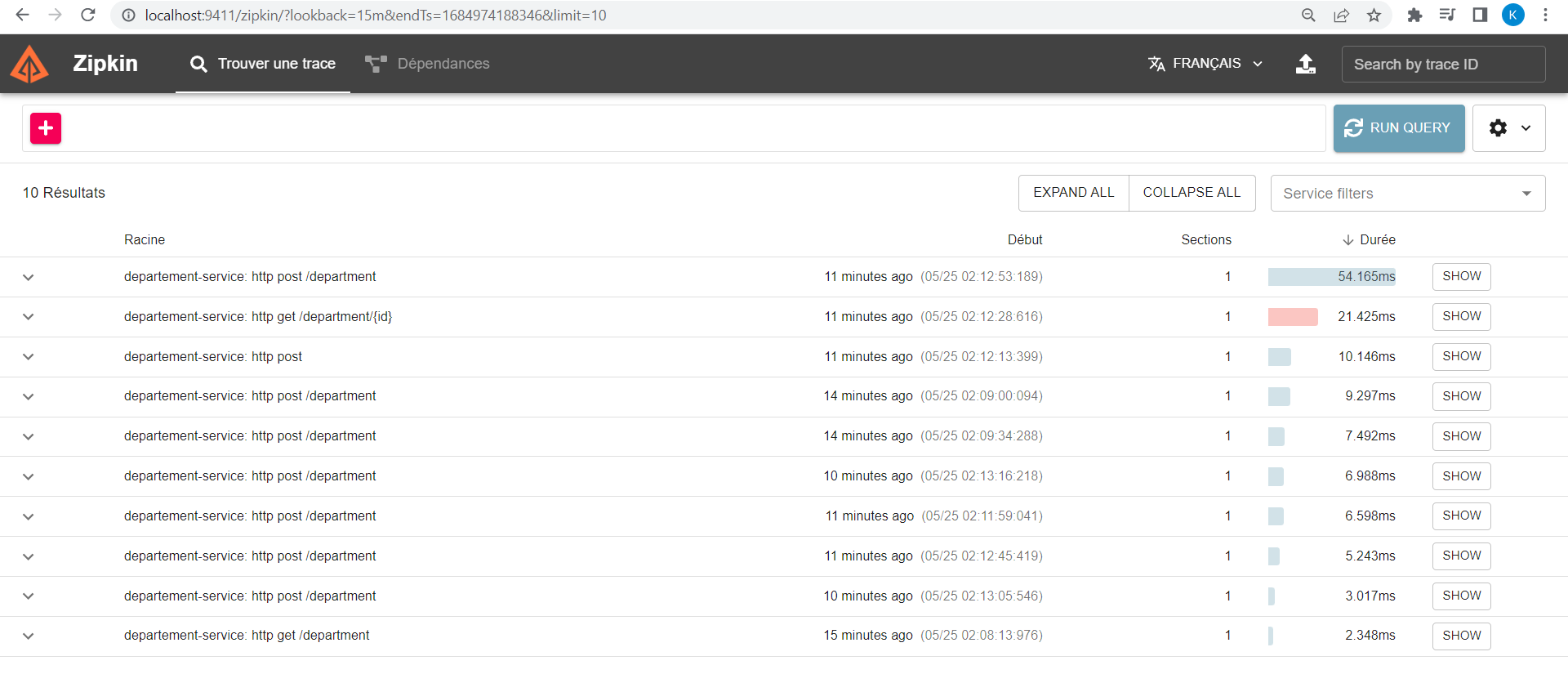




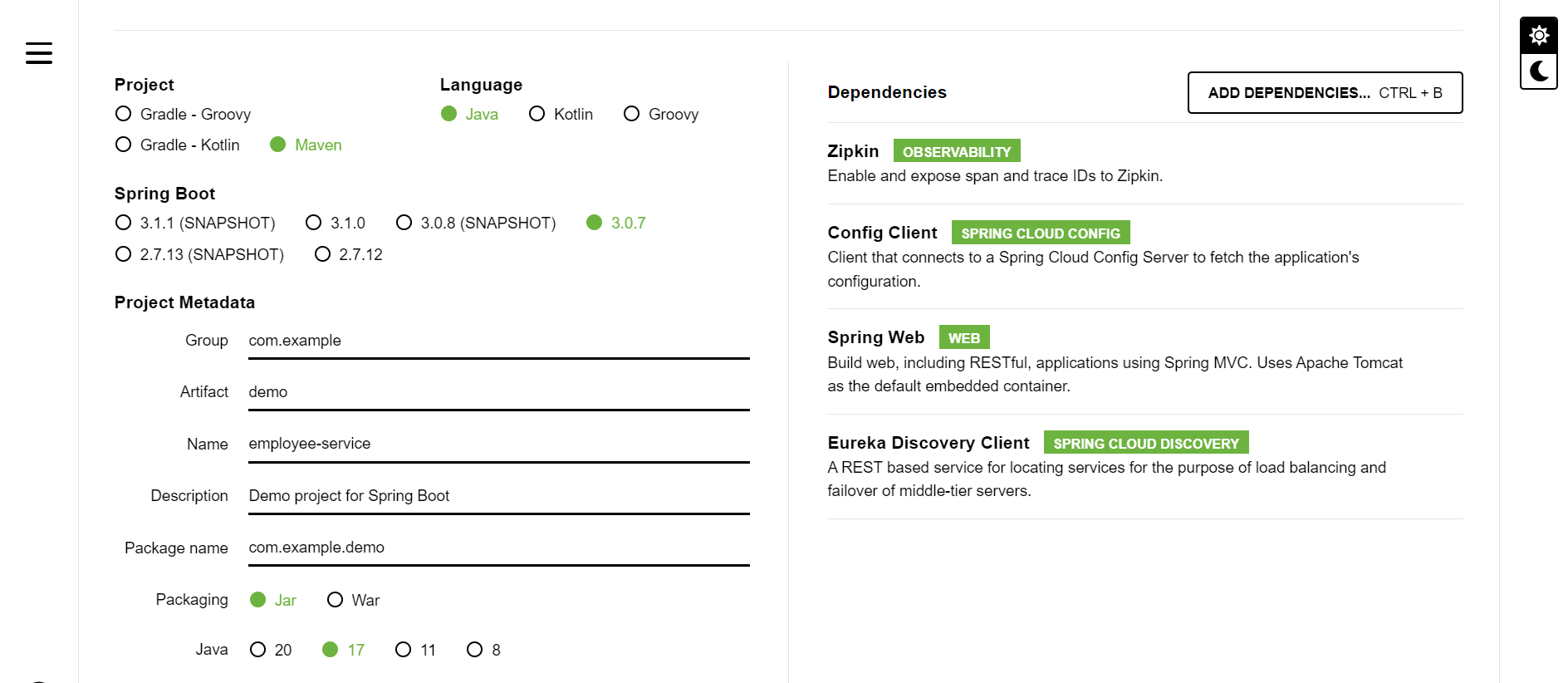
Après je vais le porter sur le serveur de centralisation de configuration.



IF YOU SEE NOW AFTER CALLING DOING SOME RECORD

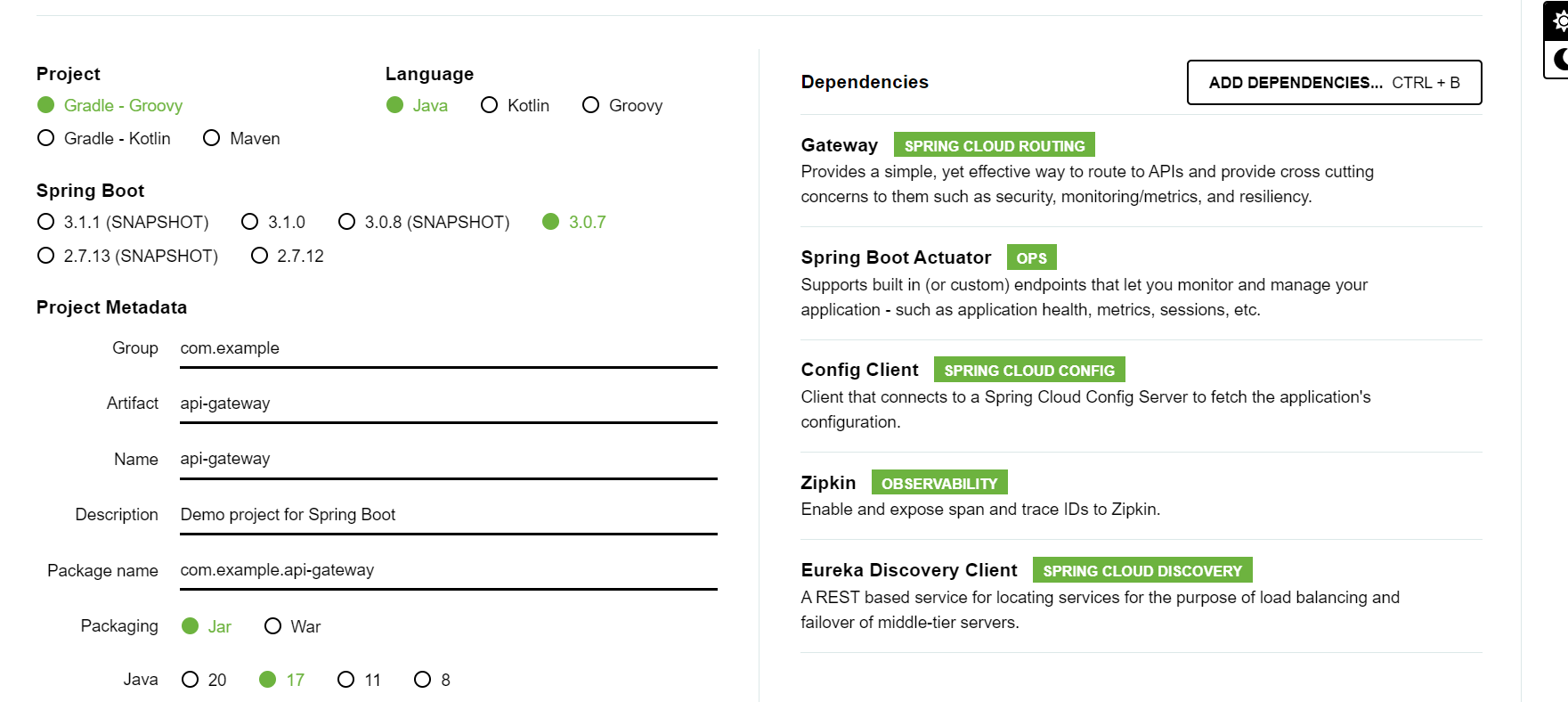


WE HAD TO CREATE ANOTHER EMPLOYEE SERVICE LIKE THIS :

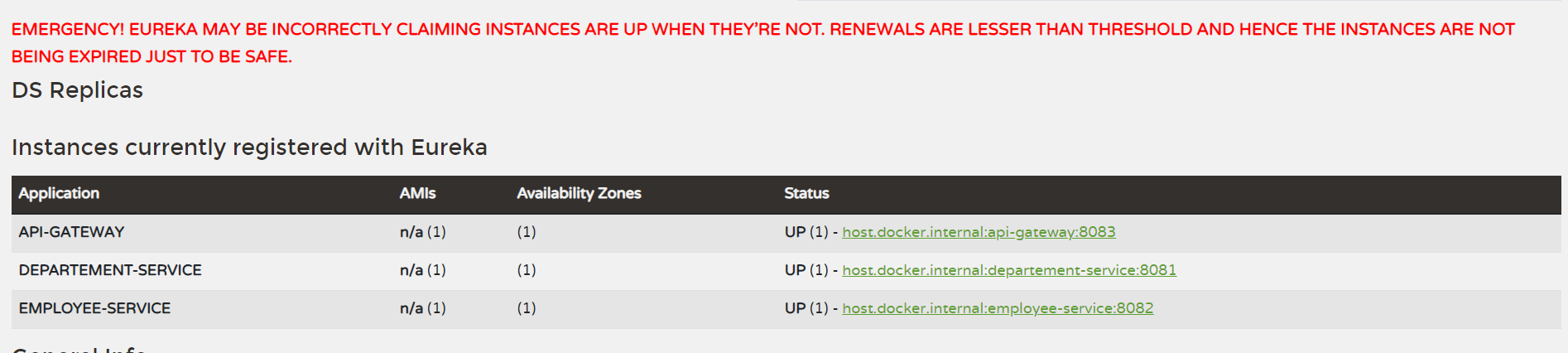


API GATEWAY

\*\*



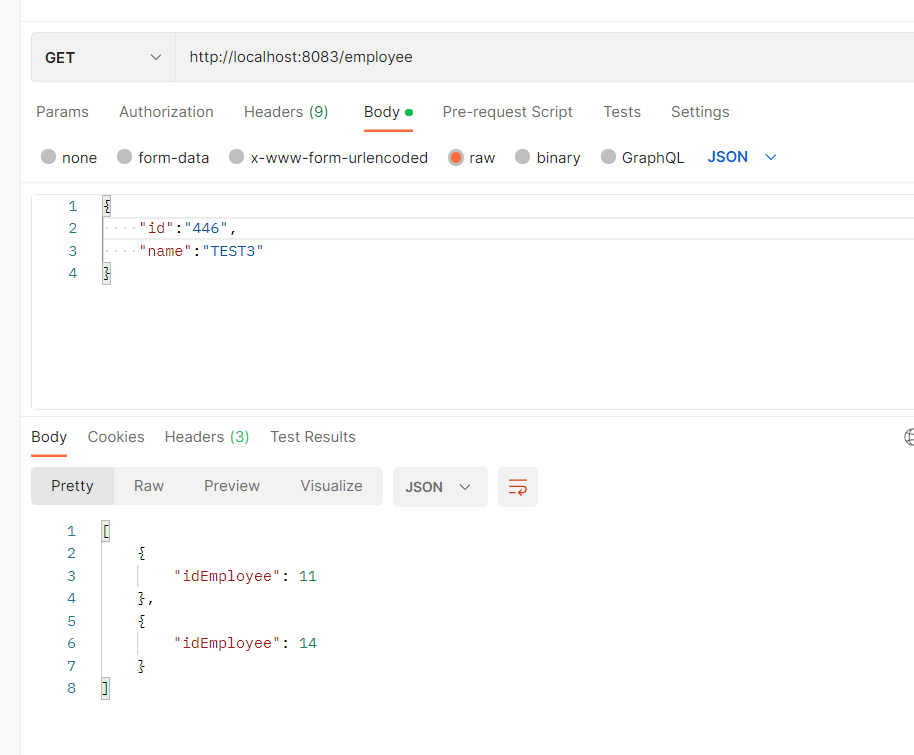
server :  
 port : 8083  
eureka:  
 client:  
 serviceUrl:  
 defaultZone : http://localhost:8761/eureka  
 instance:  
 preferIpAddress : true  
management:  
 tracing:  
 sampling:  
 probability : 1.0  
spring:  
 application:  
 name : api-gateway  
 cloud:  
 gateway:  
 routes:  
 - id: employee-service  
 uri: lb://employee-service  
 predicates:  
 - Path=/employee/\*\*  
 - id: departement-service  
 uri: lb://departement-service  
 predicates:  
 - Path=/department/\*\*  
  
 config:  
 import: "optional:configserver:http://localhost:8088"



API GATEWAY IL EST IMPLEMENTEE /

JE PEUX MAINTEANNT APPELER A PARTIR DE L API SANS PASSER DIRECTMEENT PAR API DE CHAQUE MICROSERVICE /

Example :



8083 port of api gateway

8082 OF MICROSERVICE :

