
TP N°4 :

**Service Registry/Discovery des
microservices : Spring Cloud Eureka**

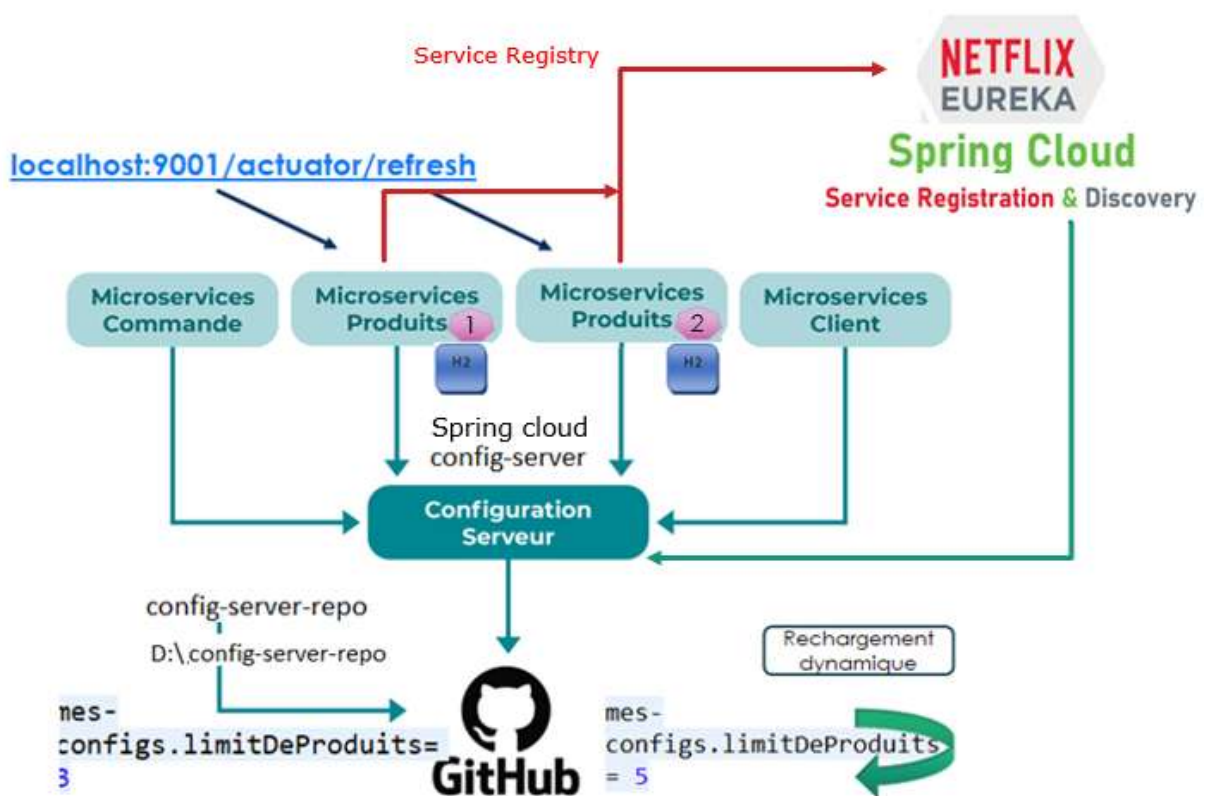
1. Prérequis

- TP3.1 Spring Cloud Config : ce n'est pas nécessaire d'avoir le Spring Cloud Config Server
- POSTMAN ou un autre outil pour tester les méthodes POST, PUT et DELETE.

2. Objectifs

1. Mise en place d'un Eureka Server : console Eureka
2. Utiliser l'annotation `@EnableEurekaServer`
3. Service Registry :
 - a. Annotation `@EnableDiscoveryClient`
 - b. Clonage d'un microservice et enregistrement auprès de Eureka Server
4. Service Discovery

3. Architecture de mise en œuvre



- Un serveur Eureka
- Deux instances du micro-service-produit qui s'enregistrent auprès de Eureka
- Remarquer que Eureka Server s'enregistre lui-même auprès de Spring cloud config Server à l'instar des autres microservices applicatifs (Produit, commande, client)
- Dans ce TP on va se baser sur la version avec Spring Cloud Config Server (TP3.1). Toutefois vous pouvez configurer « Eureka server » et le MS-Produits à fonctionner en mode configuration interne sans Spring Cloud Config.

4. Mise en place de Eureka Server

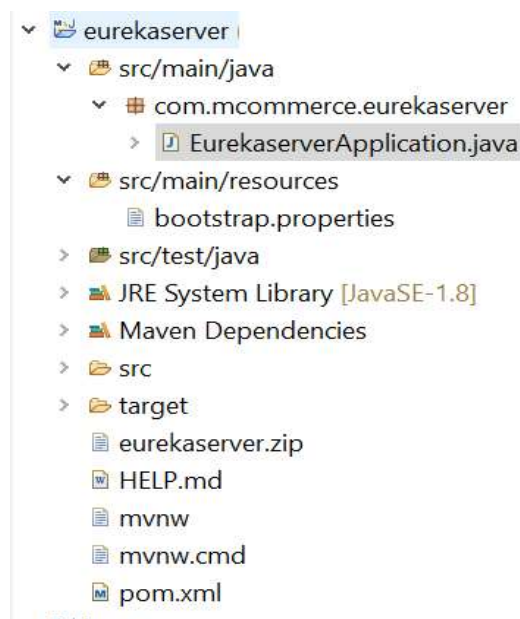
Project
☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy
☒ Maven

Spring Boot
☐ 3.2.0 (SNAPSHOT) ☐ 3.2.0 (RC1) ☐ 3.1.6 (SNAPSHOT) ☐ 3.1.5
☐ 3.0.13 (SNAPSHOT) ☐ 3.0.12 ☐ 2.7.18 (SNAPSHOT) ☒ 2.7.17

Project Metadata
Group
Artifact
Name
Description
Package name
Packaging ☒ Jar ☐ War
Java ☐ 21 ☐ 17 ☐ 11 ☒ 8

Dependencies ADD DEPENDENCIES... CTRL + B
Eureka Server SPRING CLOUD DISCOVERY
spring-cloud-netflix Eureka Server.
Config Client SPRING CLOUD CONFIG
Client that connects to a Spring Cloud Config Server to fetch the application's configuration.

- Arborescence du projet Eclipse « eureka-server »



- Fichier « pom.xml »

```
> <?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>3.2.1</version>
      <relativePath /> <!-- lookup parent from repository -->
    </parent>
```

```

<groupId>com.mcommerce</groupId>
<artifactId>eurekaServer</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>eurekaServer</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>17</java.version>
  <spring-cloud.version>2023.0.0</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-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-
server</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-bootstrap</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>

```

a. Le fichier « **bootstrap.properties** » du projet du « eurekaServer »:

```
spring.application.name=eureka-server
```

```
spring.cloud.config.uri=http://localhost:9101  
management.endpoints.web.exposure.include=refresh
```

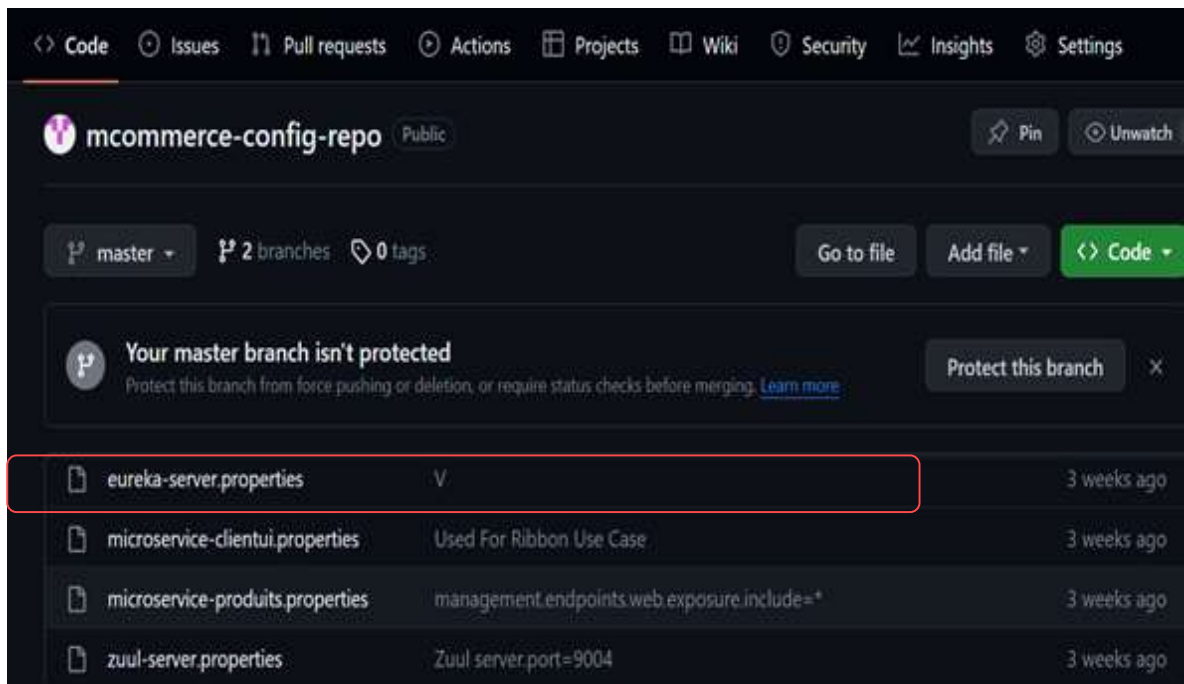
- b. Le fichier « eureka-server.properties » qui sera synchronisé avec Github :

```
server.port= 9102  
spring.application.name=eureka-server  
# Single eureka mode, and note Cluster mode  
eureka.client.registerWithEureka=false  
eureka.client.fetchRegistry=false  
spring.cloud.config.import-check.enabled=false  
#To avoid eureka server Connect to localhost:8761 timed out  
eureka.server.maxThreadsForPeerReplication=0
```

Pour rappel on utilise le projet Eclipse du TP3.1 « config-server-repo » pour se synchroniser avec Github :



- Vérifier que le fichier « eureka-server.properties » est bien pushé sur Github :



c. La Classe principale `com.mcommerce.eurekaserver.EurekaserverApplication` :

```
package com.mcommerce.eurekaserver;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@EnableEurekaServer
@SpringBootApplication
public class EurekaserverApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaserverApplication.class, args);
    }
}
```

- d. Vérifier le bon démarrage de EurekaServer :

Ne pas oublier de démarrer Spring Cloud Config Server qui écoute sur le port 9101.

.\ / _ _ _ _ _ (_) _ _ _ _ \ \ \ \

```
[main]: Fetching config from server at : http://localhost:9101
[main] Located environment: name=eureka-server, profiles=[default],
label=null, version=, state=null
[main] Located property source: [BootstrapPropertySource
{name='bootstrapProperties-configClient'}, BootstrapPropertySource
{name='bootstrapProperties-https://github.com/XXXX/mcommerce-config-
repo.git/eureka-server.properties'}]
[main] Exposing 1 endpoint(s) beneath base path '/actuator'
[main] EurekaServiceRegistry: Registering application EUREKA-SERVER
with eureka with status UP
[main] Tomcat started on port(s): 9102 (http)
EurekaServerInitializerConfiguration : Started Eureka Server
```

- e. Vérifier que Eureka Server a été bien enregistré dans notre Spring Cloud Config Server <http://localhost:9101/eureka-server/master>



```
{
  "name": "eureka-server",
  "profiles": [
    "master"
  ],
  "label": null,
  "version": "318d25b37ea389532ad42e4a179e75d3b4bbeb43",
  "state": null,
  "propertySources": [
    {
      "name": "https://github.com/mcommerce-config-repo.git/eureka-server.properties",
      "source": {
        "server.port": "9102",
        "spring.application.name": "eureka-server",
        "eureka.client.registerWithEureka": "false",
        "eureka.client.fetchRegistry": "false",
        "spring.cloud.config.import-check.enabled": "false"
      }
    }
  ]
}
```

f. Accéder à Eureka Server : <http://localhost:9102/>

The screenshot displays the Spring Eureka Server web interface. The browser's address bar shows 'localhost:9102'. The page header includes the 'spring Eureka' logo and a 'HOME' link. The main content area is divided into several sections:

- System Status:** A table showing environment details (test, default) and system metrics (Current time: 2023-10-21T19:31:17 +0100, Uptime: 00:00, Lease expiration enabled: false, Renewal threshold: 1, Renewal (last min): 0).
- DS Replicas:** A section with a link to 'localhost'.
- Instances currently registered with Eureka:** A table with columns 'Application', 'AMIs', 'Availability Zones', and 'Status'. It currently shows 'No instances available'.
- General Info:** A table listing system metrics such as total-avail-memory (295mb), num-of-cpus (8), current-memory-usage (151mb (51%)), server-up-time (00:00), registered-replicas (http://localhost:8761/eureka/), unavailable-replicas (http://localhost:8761/eureka/), and available-replicas.
- Instance Info:** A table showing instance details like ipAddr (192.168.1.105) and status (UP).

5. Développement du microservice « Produit » : Client Eureka

- a. Fichier « bootstrap.properties » du MS-Produits :

```
spring.application.name=microservice-produits
spring.cloud.config.uri=http://localhost:9101
```

- b. Fichier « **microservice-produits** » du MS-Produits pushé sur Github: le microservice-produits va s'enregistrer à Eureka Server :

```
#Configurations H2
spring.jpa.show-sql=true
spring.h2.console.enabled=true
#defini l'encodage pour data.sql
spring.datasource.sql-script-encoding=UTF-8

#Eureka :indique l'URL d'Eureka à laquelle il faut s'enregistrer
eureka.client.serviceUrl.defaultZone=http://localhost:9102/eureka/

#Actuator : management.endpoints.web.exposure.include=*
management.endpoints.web.expose=info, health, refresh
#Les configurations personnalisés
mes-configs.limitDeProduits= 3
```

- c. Vérifier que le starter « **spring-cloud-starter-netflix-eureka-client** » existe au niveau du fichier pom.xml :

```
<?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>3.2.0</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.mproduits</groupId>
  <artifactId>microservice-produits</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>microservice-produits</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>17</java.version>
    <spring-cloud.version>2023.0.0</spring-cloud.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
```

```

</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</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</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-bootstrap</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>

<!--
https://mvnrepository.com/artifact/org.springframework.cloud/spring-
cloud-starter-sleuth -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
    <version>3.1.10</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
</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>

```

d. Classe principale du MS-Produits :

```

package com.mproduits;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
// pour Spring Cloud Config
@EnableConfigurationProperties

// pour Eureka Client
@EnableDiscoveryClient

public class MproduitsApplication {
    public static void main(String[] args) {
        SpringApplication.run(MproduitsApplication.class, args);
    }
}

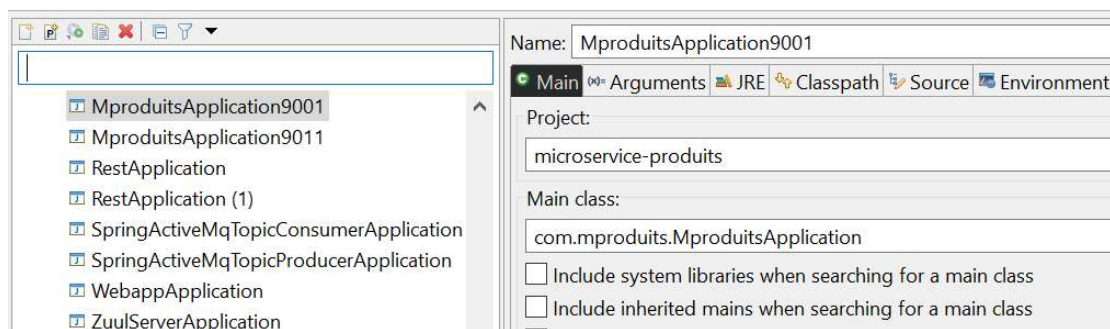
```

e. Démarrer la 1^{ère} instance du MS-Produits qui va écouter sur le port 9001 :

 Run Configurations

Create, manage, and run configurations

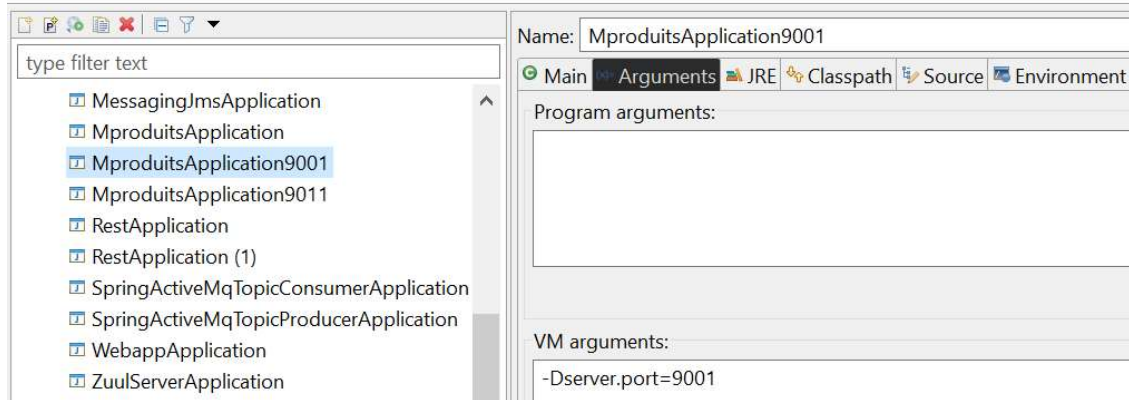
Run a Java application





Create, manage, and run configurations

Run a Java application

[illegible]

```
[main] Fetching config from server at : http://localhost:9101
[main] Located environment: name=microservice-produits,
[main] Located property source: [BootstrapPropertySource
{name='bootstrapProperties-https://github.com/XXX/mcommerce-config-
repo.git/microservice-produits.properties'}]]
H2 console available at '/h2-console'. Database available at
'jdbc:h2:mem:aba96b16-544a-47fa-9915-50a70980def0'
[main] EurekaServiceRegistry: Registering application MICROSERVICE-
PRODUITS with eureka with status UP
[main] com.netflix.discovery.DiscoveryClient: Saw local status
change event StatusChangeEvent [timestamp=1700392806825, current=UP,
previous=STARTING]
[nfoReplicator-0] com.netflix.discovery.DiscoveryClient:
DiscoveryClient_MICROSERVICE-PRODUITS/localhost:microservice-
produits:9001: registering service...
[main] TomcatWebServer : Tomcat started on port(s): 9001 (http)
with context path
```

➔ <http://localhost:9102/>

The screenshot shows the Spring Eureka web interface at localhost:9102. The 'Instances currently registered with Eureka' section is highlighted with a red box. It shows one instance of 'MICROSERVICE-PRODUITS' with status 'UP (1)' and a link to 'localhost:9001'.

| Application | AMIs | Availability Zones | Status |
|-----------------------|---------|--------------------|---|
| MICROSERVICE-PRODUITS | n/a (1) | (1) | UP (1) - localhost:9001 |

Below this, the 'General Info' and 'Instance Info' sections provide details about the system and the specific instance.

➔ On remarque que la 1^{ère} instance du MS-Produit a été enregistrée auprès de Eureka.


➔ Également, Au niveau de la console de Eureka Server, on remarque l'enregistrement de la 1^{ère} instance qui écoute sur 9001:

Registered instance MICROSERVICE-PRODUITS/localhost:microservice-produits:9001 with status UP (replication=false)

f. Clonage du MS-Produit: Démarrer une 2^{ème} instance du MS-Produits qui écoute sur 9011

The screenshot shows the 'Run Configurations' dialog in an IDE. The configuration 'MproductsApplication9011' is selected. The 'Main' tab is active, showing the program arguments as '-Dserver.port=9011'.

➔ <http://localhost:9102/>


HOME LAST 1000 SINCE STARTUP

System Status

| | | | |
|-------------|---------|--------------------------|---------------------------|
| Environment | test | Current time | 2023-11-19T12:02:07 +0000 |
| Data center | default | Uptime | 01:14 |
| | | Lease expiration enabled | false |
| | | Renews threshold | 5 |
| | | Renews (last min) | 4 |

DS Replicas

localhost

Instances currently registered with Eureka

| Application | AMIs | Availability Zones | Status |
|-----------------------|---------|--------------------|--|
| MICROSERVICE-PRODUITS | n/a (2) | (2) | UP (2) - local:microservice-produits:9011 , local:microservice-produits:9001 |

General Info

| Name | Value |
|--------------------|-------|
| total-avail-memory | 358mb |

- ➔ On remarque que la 2^{ème} instance du MS-Produit a été enregistrée auprès de Eureka.
- ➔ Également, Au niveau de la console de Eureka Server, on remarque l'enregistrement de la 2^{ème} instance qui écoute sur 9011:

Registered instance MICROSERVICE-PRODUITS/localhost: **microservice-produits:9011** with status UP (replication=false)

NB : Le service Discovery sera mis en œuvre via l'API Gateway Zuul qui va exploiter les fonctionnalités de Eureka Server.

6. Annexe

Au niveau de Eclipse, il est possible de visualiser les différentes consoles de microservices qui sont lancés :

