esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2024-2025 **Applications Web Distribuées** **UP_WEB**

Implémentation « Eureka server »

Mise en situation

- Dans le cas d'une application qui répond à une montée en charge et dans le cas de plusieurs instances de chaque Microservice, il est nécessaire d'avoir un **registre de toutes les instances disponibles** afin de distribuer la charge entre celles-ci.

  ➔ **Eureka** de Netflix est une solution qui permet d'enregistrer les instances des Microservices et contient toutes les informations, sur quelle adresse IP et quel port les microservices client exécutés.

- L'appel d'un Microservice s'effectue à travers la liste des instances qu'Eureka expose via une API REST.

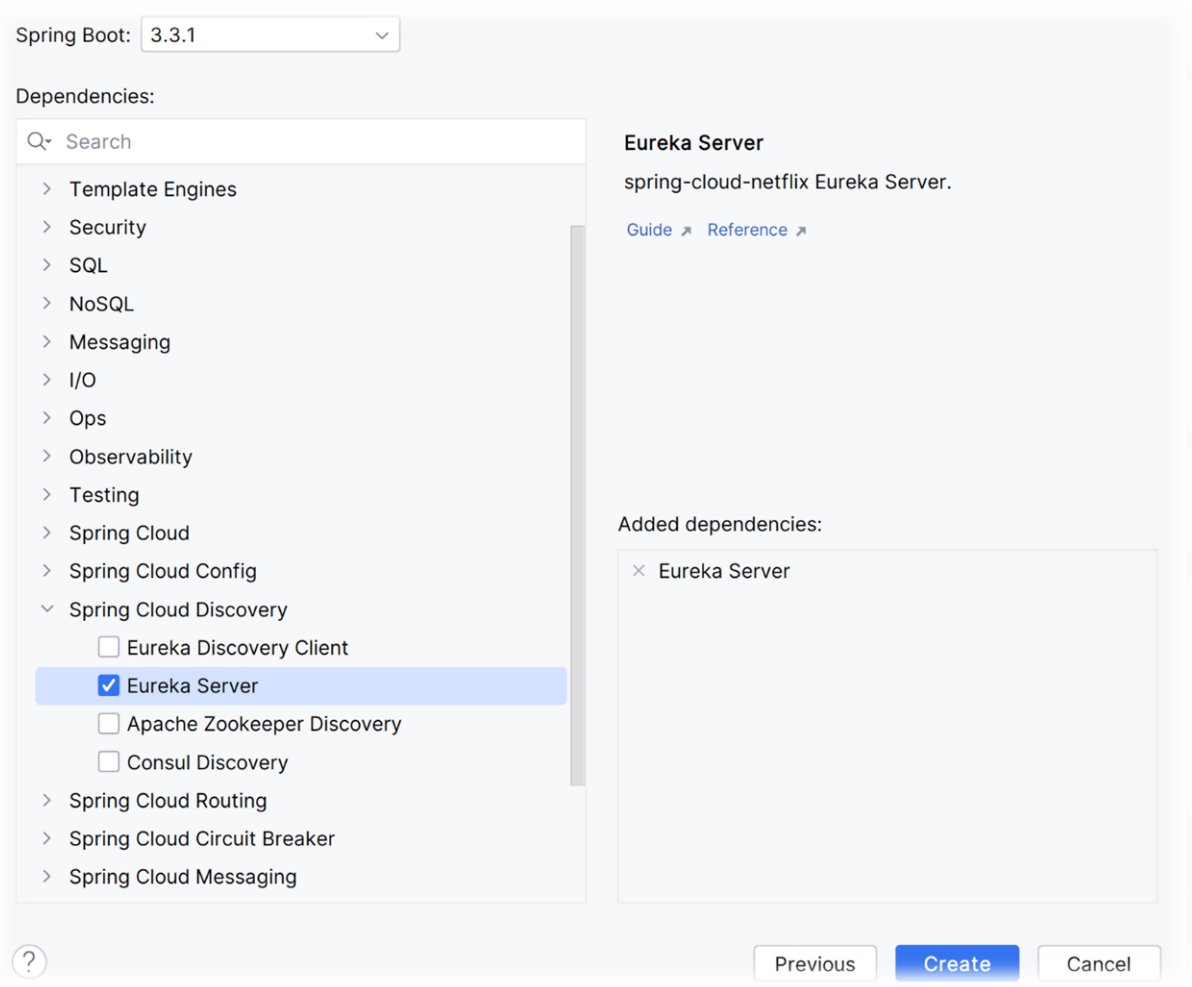
Objectifs

- Créer un **serveur de découverte** « **Eureka** ».
- Instanciation et découverte de Microservices.

Partie 1 : Mise en place de serveur « Eureka »

Etape 1 : Création d'une application Spring Eureka Server

Créer une application **Spring** et ajouter la dépendance « **Eureka Server**»



Vérifier si la dépendance ajoutée procède comme suit dans le fichier **POM.xml**.

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

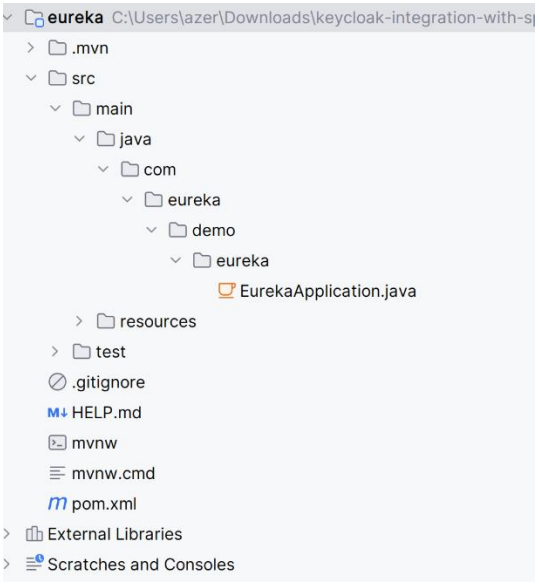
Etape 2: Spring Boot Starter Application

Dans la classe **EurekaApplication.java**, en haut de la classe ajouter l'annotation suivante : **@EnableEurekaServer :** utilisée pour que l'application Spring Boot se comporte comme un serveur Eureka.

```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaApplication {

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

}
```

Etape 3: Application Properties File

Dans le fichier **application.properties** situé à l'adresse *src / main / resources*, ajouter la liste des propriétés ci-dessous.




```
spring.application.name=eureka
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

spring.application.name donne un nom unique pour l'application.

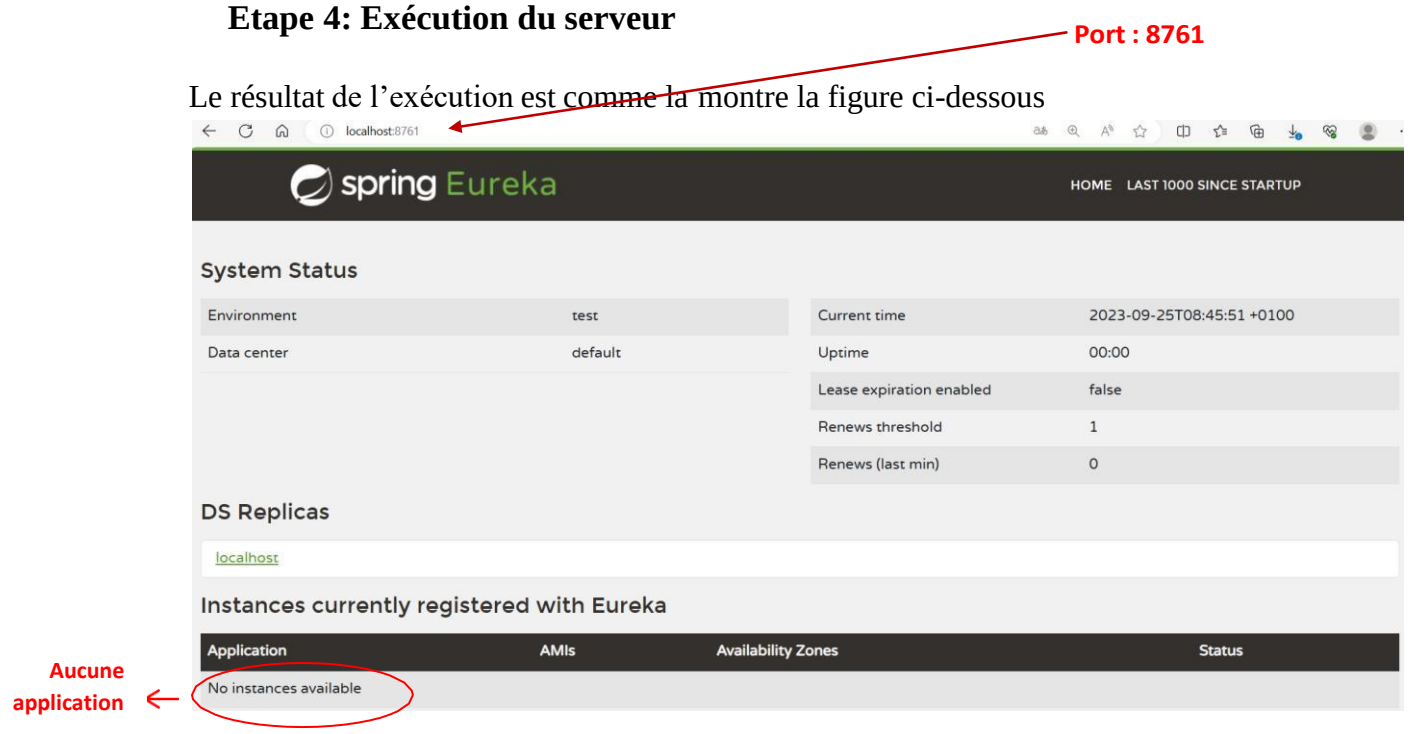
server.port définit un port (8761) par défaut pour le serveur eureka.

eureka.client.fetch-registry : désigne qu'il ne faut pas cacher en local les clients enregistrés, puisqu'on est le serveur

eureka.client.register-with-eureka : Ne pas s'auto-déclarer sur Eureka, puisqu'on est le serveur.

Etape 4: Exécution du serveur

Le résultat de l'exécution est comme la montre la figure ci-dessous



Partie 2 : découverte de Microservices (Eureka clients)

La découverte de services est l'un des principes clés de l'architecture Microservices. La configuration manuelle de chaque client est difficile à faire.

Eureka est le serveur Netflix de découverte de services et de clients qui peut être configuré et déployé pour être hautement disponible où chaque serveur répliquant l'état des services enregistrés aux autres.

Etape 1 : intégration de client eureka

- Dans le fichier **pom.xml**, ajouter la dépendance suivante

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

Etape 2 : S'inscrire à Eureka

Lorsqu'un client s'enregistre auprès d'Eureka, il fournit des métadonnées sur lui-même, telles que l'hôte et le port, l'URL, la page d'accueil, etc. Eureka reçoit des messages de chaque instance appartenant à un service. Si l'appel échoue, l'instance sera supprimée du registre.

1. Dans le fichier **CandidatApplication.java**, ajouter l'annotation **:**

@EnableDiscoveryClient: sert à annoter (annotate) que l'application sera transformée en Eureka Client.

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
@EnableDiscoveryClient
public class CandidatApplication {

    public static void main(String[] args) {
```

2. Dans le fichier **application.properties** (application Candidat), ajouter les lignes de code suivantes sachant que :

eureka.client.service-url.default-zone fournit l'URL du service à tout client n'exprimant pas de préférence (il s'agit d'un paramètre utile par défaut).

```
# Service
spring.application.name=candidat-s
server.port=8082

# eureka registration
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
eureka.client.register-with-eureka=true
```

Etape 2 : Authentification avec le serveur Eureka

L'authentification HTTP de base sera automatiquement ajoutée à votre client eureka si l'une des URL ***eureka.client.service-url.default-Zone*** contient des informations d'identification.

Le résultat de l'exécution est la suivante

URL : *localhost :8761*

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CANDIDAT-S	n/a (2)	(2)	UP (2) - DESKTOP-62GT31U:candidat-s:8082 , DESKTOP-62GT31U:candidat-s:8083