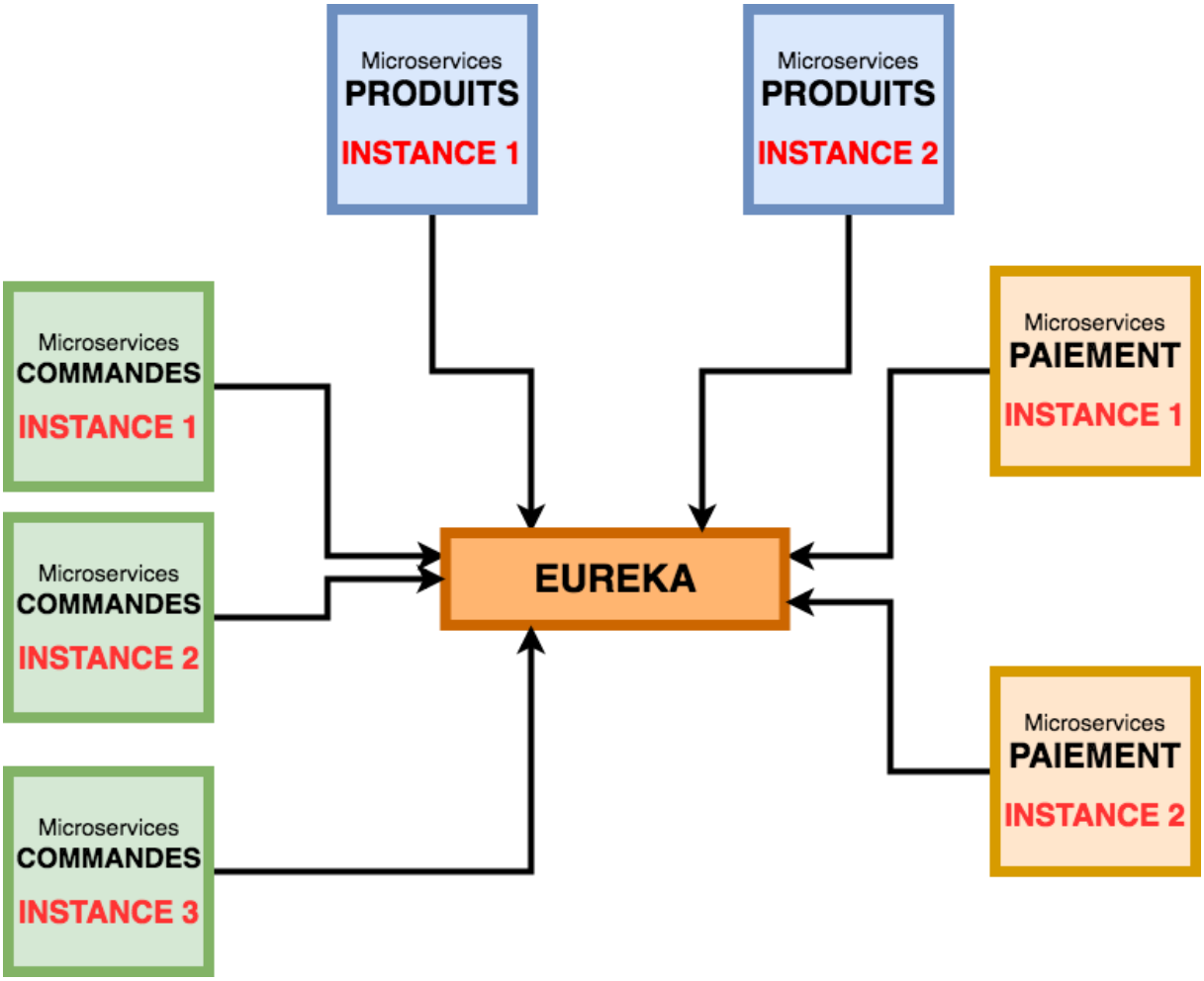
Atelier 5
Rendre les Microservices découvrables grâce à Eureka

Quand votre application répond à une montée en charge et que vous avez plusieurs instances de chaque Microservice, il est vital de pouvoir garder un **registre de toutes les instances disponibles** afin de distribuer la charge entre celles-ci.



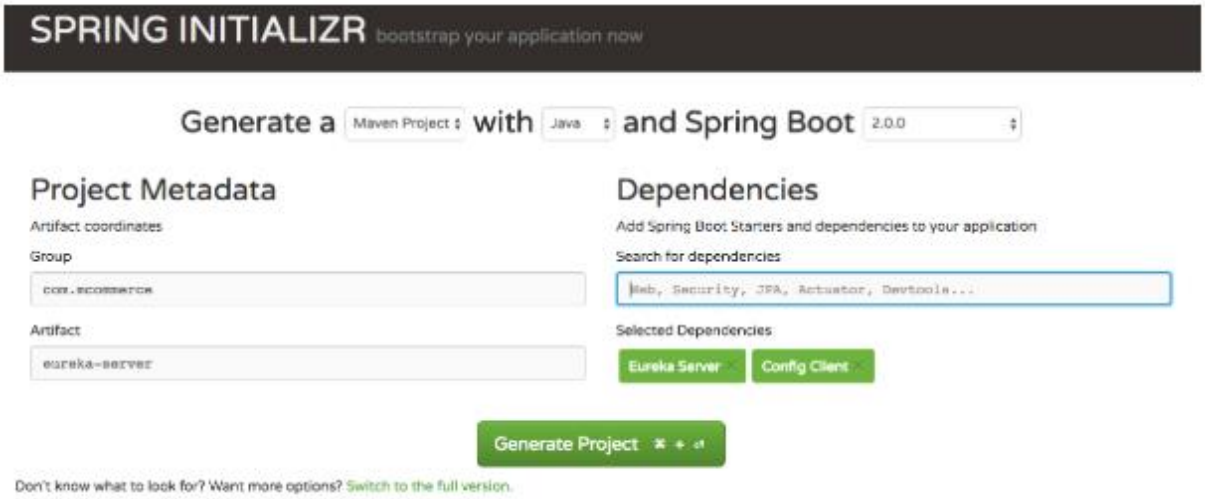
Eureka de Netflix remplit précisément cette fonction. Une fois en place, les instances des Microservices viennent s'enregistrer dans le registre d'Eureka. Pour appeler un Microservice, il suffira de piocher dans cette liste d'instances qu'Eureka expose via une API REST.

Eureka offre un client capable de réaliser des opérations de récupération des listes d'instances.

1. Mettre en place Eureka

Commencez par générer un Microservice Eureka sur Spring Initializr en y ajoutant **"Eureka Server"** et **"Config Client"** qui va nous permettre d'externaliser la configuration :

Importez le Microservice dans IntelliJ, de la même façon que vous l'avez fait pour les précédents.



Externalisez le fichier de configuration en créant un fichier

eureka-server.properties dans *config-server-repo* .

eureka-server.properties

```
server.port:9102

spring.application.name=eureka-server
eureka.client.serviceUrl.defaultZone:http://localhost:9102/eureka/
eureka.client.registerWithEureka:false
eureka.client.fetchRegistry:false
```

Explications :

- On définit le port dans lequel Eureka sera accessible : 9102.
- `eureka.client.serviceUrl.defaultZone` : quand vous utilisez le client Eureka pour accéder au registre d'Eureka, vous devez renseigner cette ligne de configuration pour pointer vers l'URL d'Eureka. Alors pourquoi allons-nous renseigner l'URL d'Eureka si ce Microservice l'est ? Eh bien parce qu'Eureka offre la possibilité de se répliquer en plusieurs instances ; vous pouvez alors pointer vers d'autres instances d'Eureka. Ce mode s'appelle **"cluster mode".** Comme nous n'allons utiliser qu'un seul serveur Eureka, nous allons pointer vers sa propre URL.
- `eureka.client.registerWithEureka` et `eureka.client.fetchRegistry` sont tous les 2 à false, étant donné que nous n'allons pas utiliser Eureka en mode cluster.

N'oubliez pas de renommer *application.properties* en *bootstrap.properties*.

Ajoutez l'annotation `@EnableEurekaServer` à EurekaServerApplication.java :

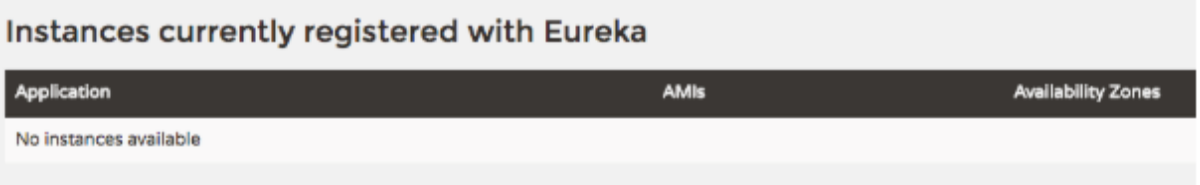
```
package com.mcommerce.eurekaserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```java
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {

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

C'est tout ! Assurez-vous que config-server est lancé, puis lancez Eureka.

Rendez-vous ensuite à http://localhost:9102/. Vous arriverez alors sur une page qui présente un certain nombre d'informations sur votre serveur Eureka.

Les plus importantes sont celles-ci :



C'est ici que vous verrez apparaître la liste des instances des Microservices qui viendront s'enregistrer.

2. Client Eureka

Afin que les différents Microservices commencent à s'enregistrer dans notre serveur Eureka, nous devons ajouter le client Eureka à chacun d'entre eux.

Prenons comme exemple Microservices-produits. Rendez-vous dans le *pom.xml* et ajoutez cette dépendance :

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

Ajoutez ensuite cette ligne à microservice-produits.properties dans le dépôt :

```
#Eureka
eureka.client.serviceUrl.defaultZone: http://localhost:9102/eureka/
```

Cette ligne indique l'URL d'Eureka à laquelle il faut s'enregistrer.

Rendez-vous enfin à MproduitsApplication.java et ajoutez l'annotation **`@EnableDiscoveryClient`** :

```java
package com.mproduits;

import com.mproduits.configurations.ApplicationPropertiesConfiguration;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
```

```java
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableConfigurationProperties
@EnableDiscoveryClient
public class MproduitsApplication {

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

Votre Microservice bénéficie à présent du client Eureka, qui ira enregistrer votre instance à chaque démarrage.

Démarrez votre service et rendez-vous à http://localhost:9102/ :

Vous voyez alors que votre Microservice est enregistré avec un statut "UP" indiquant qu'il est en fonction.

Si vous arrêtez le Microservice-produits, vous verrez qu'Eureka le détecte immédiatement et le supprime du registre !

La branche GIT pour ce chapitre est **Eureka.**