TP6

Objectifs du TP :

- Utilisation de Discovery Service
- Utilisation du service GateWay

I- Microservice Eureka Discovery service

L'objectif de cette section et de créer un Microservice (MS) Discovery qui permet de collecter des données telles que les noms, les adresse IP et les ports…etc. des différents MS de notre application.
Le fonctionnement du MS Discovery est décrit comme suit :
Chaque MS nouvellement lancé doit d'abord s'enregistrer auprès du MS Discovery. Ainsi le M.S Discovery reporte dans son registre interne une liste d'informations des MS lancés et éventuellement leurs instances. Par exemple dans la figure 1 le MS Discovery, en plus des MS Produit et Commande, enregistre la liste des instances du MS Catalogue. Ce registre est partagé ensuite avec les autres MS. En cas de panne du MS Discovery le MS concerné peut utiliser son propre registre pour communiquer avec un autres MS. Ceci garanti une meilleure fiabilité de notre application.

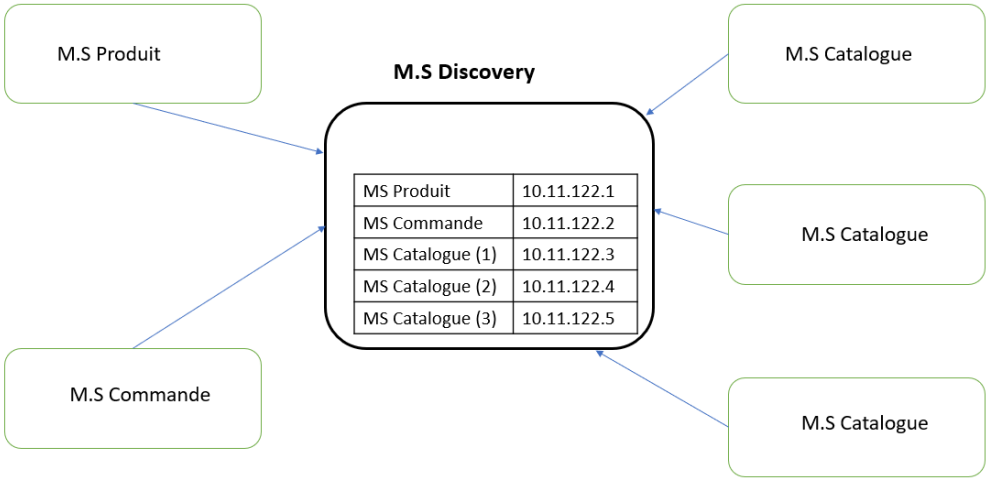
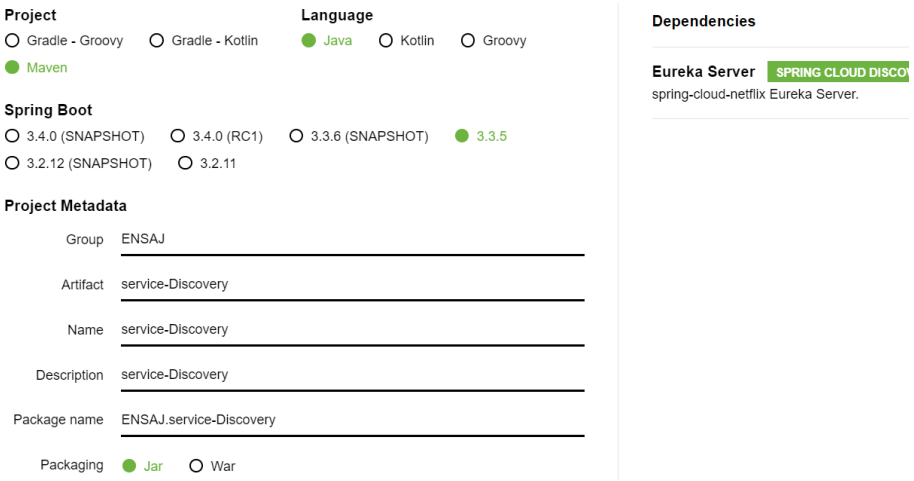


Figure 1 Fonctionnement du MS Discovery

Dans ce TP, on va utiliser le service Eureka de Netflix comme MS Discovery. Pour pouvoir l'utiliser on se dirige vers spring initilizr et on génère un projet comme illustré dans la figure 2 :



Prof. BAIDADA Chafik
Chafik29@gmail.com

Figure 2 projet depuis Spring Initializer

On ouvre le fichier application.properties, et ajoute les informations suivantes :

```
spring.application.name=service-Discovery
eureka.instance.hostname=localhost
eureka.client.fetch-registry=false
eureka.client.register-with-eureka=false
server.port=8761
```

Figure 3 : Configuration du MS Discovery

Dans la figure 3, on nomme notre service et on lui attribut un port. Après on lui interdit de s'autoenregistrer en tant que client. On lui interdit aussi de chercher le registre puisqu'il possède son propre registre en mettant à false l'attribut fetch-registry. Ensuite, on ouvre la main classe et on ajoute l'annotation : `@EnableEurekaServer.` Enfin, on lance le projet et on vérifie dans le navigateur que ce MS est bien lancé.

II- MicroService Catalogue

1. Installation du client Eureka

Pour que ce MS puisse s'enregistrer en tant que client dans le MS Discovery, On ouvre le projet du MS Catalogue et on ajoute les dépendances Eureka client de Netflix comme indiqué sur la figure4

```
    <properties>
        <java.version>17</java.version>
        <spring-cloud.version>2023.0.3</spring-cloud.version>
    </properties>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </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>
```

Figure 4 dépendance de eureka client dans le fichier pom.xml

NB : on ajoute dans la balise properties la version du Spring cloud utilisé.

Prof. BAIDADA Chafik
Chafik29@gmail.com

On ouvre le fichier application.properties et on ajoute la configuration illustré dans la figure 5 :

```
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

Figure 5 Configuration du MS Catalogue

Cette configuration indique à au client Eureka où trouver le serveur Eureka pour l'enregistrement. Esuite, on lance le projet Catalogue et vérifie que ce MS s'est enregistré dans Eureka server (figure 6).

Application	AMIs	Availability Zones	Status
SERVICE-CATALOGUE	n/a (1)	(1)	UP (1) - instance-id=service-catalogue:c7f5dd6d6f52adec9a681ccaaf74b7b3

Figure 6 Enregistrement du MS Catalogue da

On fait le même traitement pour les autres MS à savoir Produit et Commande.

2. Lancement de plusieurs instances

Maintenant, On veut lancer plusieurs instances du MS Catalogue. Pour ce faire, on clique sur Edit configuration comme indiqué sur la figure 7, une fenêtre s'affiche.

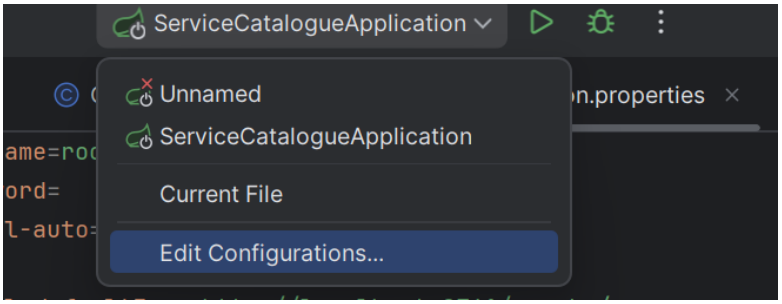


Figure 7 Edit Configuration

On clique sur « Modify Option », puis sur « Allow multiple inctance » et on termine par cliquer sur « Aplly » et puis « ok »

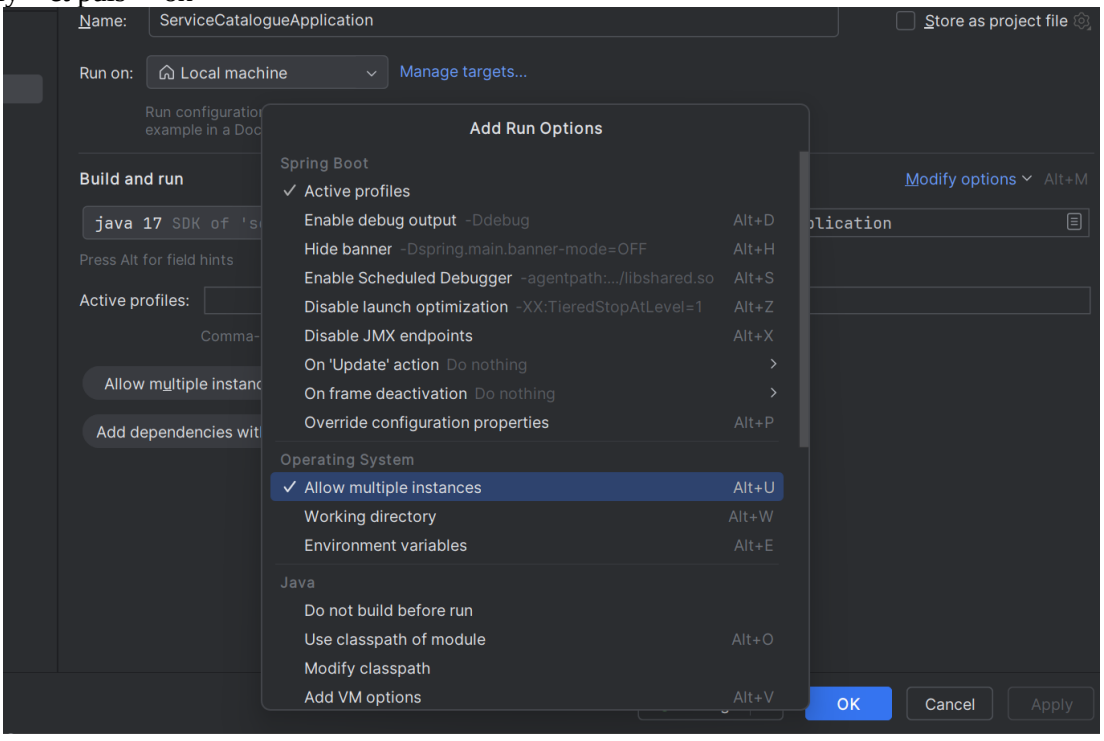


Figure 8 autoriser l'exécution en plusieurs instances

Prof. BAIDADA Chafik
Chafik29@gmail.com

Pour que notre MS puisse se lancer sur plusieurs instances et à chaque fois dans plusieurs ports, on doit mettre le port à 0 dans le fichier prperties.

```
server.port=0
```

Ensuite, il faut préciser à Spring de donner un identifiant unique à chaque instance en utilisant le nom du MS suivi d'une valeur aléatoire :

```
eureka.instance.instance-id=instance-id=${spring.application.name}:${random.value}
```

Un MS pour indiquer à Eureka Server qu'il toujours en exécution (en vie), il envoi à chaque intervalle de temps un signal appelé **"heartbeat".**

```
eureka.instance.lease-renewal-interval-in-seconds=10
```

Cette configuration définit l'intervalle de renouvellement «heartbeat » à Eureka tous les 10 secondes pour signaler qu'elle est toujours en exécution.
On ajoute aussi la configuration suivante pour que l'intervalle de renouvellement soit à 30 pour donner suffisamment de temps aux instances pour renouveler leur bail, en prenant en compte les délais réseau

```
eureka.instance.lease-expiration-duration-in-seconds=30
```

Après On lance une 3 instances de MS Catalogue et lance le navigateur pour vérifier l'enregistrement de ces trois services sur Eureka (figure 9)

Figure 9 Eregistrement des trois instances du MS catalogue sur Ereaka Server

I- MicroService commande

Dans cette Section on veut faire communiquer le MS Comande avec les différentes instances du MS catalogue avec WebClient en profitant des possibilités offertes par le MS Discovery de Eureka.
Après avoir configurer se MS pour être un client de Eureka Server, on ouvre le projet du MS Commande. On ouvre le fichier CatalogueService.java et on remplace l'adresse IP du MS appelé par web client par son nom à savoir : service catalogue

```
// appeller le service catalogue pour vérifier l'éxistance de produit
CatalogueReponse[] catalogueReponsesArray =webClient.get()
        .uri("http://service-catalogue/api/catalogue",
```

Figure 10 Appel du service Catalogue par son nom

On lance le MS Catalogue deux fois en utilisant le mode Debug et on vérifie que les deux instances sont enregistrées dans le MS Discovery (Figure1).

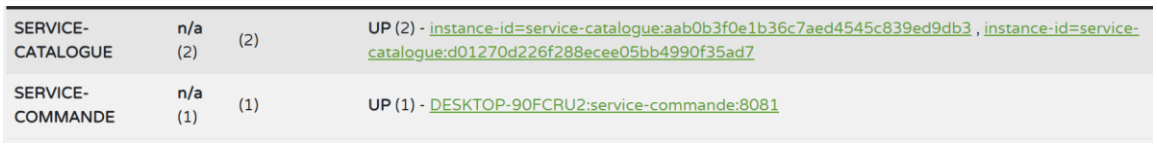


Figure 11 Communication Synchrone entre microservices

Maintenant on va ouvrir Postman et on essayer d'appeler la méthode ***établirCommande().***
Lorsque on clique sur « send » de PostMan une Erreur est générée (figure 12).

Prof. BAIDADA Chafik
Chafik29@gmail.com

Figure 12 Erreur lors de la méthode POST de Postman

En vérifiant le log sur le terminal de IntellJ on constate l'erreur dans la figure 13 :

```
reactive.function.client.WebClientRequestException: Failed to resolve 'service-catalogue' [A(1)]] with root caus
```

Figure 13 Erreur ne pas pouvoir connaitre identifier quelle instance de service-catalogue sur IntellJ

Le service commande n'arrive pas à se connecter au service catalogue puisque ce dernier s'exécute en plusieurs instances. La question qui se pose à quelle instance on doit se connecter ?
Pour y répondre, on doit activer le **load balancer** (équilibreur de charge) du coté MS Commande. Cet outil permet de chercher dans le registre du Discovery service instance par instance jusqu'à trouver la première instance libre et communiquer avec elle.
Pour réaliser ceci, on doit modifier la classe de configuration de WebClient :

```java
@Configuration
public class WebClientConfig {

    @Bean
    @LoadBalanced
    public WebClient.Builder webClientBuilder() {
        return WebClient.builder();
    }
}
```

Figure 14 Classe WebClientConfig

On change le nom de l'objet dans la classe service :

```java
private final WebClient.Builder webClientBuilder;
```

On ajoute la modification suivante au niveau de la fonction **`établirCommande`**

```java
CatalogueReponse[] catalogueReponsesArray =webClientBuilder.build().get()
        .uri("http://service-catalogue/api/catalogue",
                uriBuilder ->
uriBuilder.queryParam("skuCode",skuCodes).build()
        )
```

Figure 15 ligne modifiée dans la classe

On relance notre MS Commande et on attend 30s avant de lancer la Post http sur PostMan.
On obtient le résultat (figure 16)
Ceci montre que notre MS est arrivé à communiquer avec une instance du MS Catalogue pour vérifier l'existence d'un article dans le stock.

Prof. BAIDADA Chafik
Chafik29@gmail.com

Figure 16 test de la commande Post réussi

Pour s'assurer que les MS gardent une copie interne du registre des autres MS, on va stopper le Discovery service et on va utiliser Postman pour établir une commande. On remarque que nola commande a été établie avec succès.
Maintenant on va stopper toutes les instances du MS Catalogue et on démarre une nouvelle instance pour voir si la commande sera établie.
Une fois le test est réalisé, on aura une erreur car cette nouvelle instance n'est pas contenue dans le registre du MS Commande. On doit relancer le MS Descovery service pour que cette nouvelle instance puisse s'enregistrer. Ceci permet au Descovery Service de redistribuer son registre sur les autres MS et ainsi cette nouvelle instance soit visible.
En effectuant le test on voit bien que la commande est établie ce qui montre que notre application marche bien

II- Microservice GateWay

Travail à faire

Prof. BAIDADA Chafik
Chafik29@gmail.com