TP 5 - Service Discovery with Eureka

- Spring BOOT
- Spring Cloud Netflix
- Eureka

Estimated Time :

- **40** *minutes*

5.0 - Objectifs

Le présent TP se veut présenter comment mettre en place les technologies à partir de Spring cloud netflix eureka qui constituent les briques de bases d'un annuaire de services :

- Spring Cloud Eureka Server
- Spring Cloud Eureka Client
- Spring Boot Starter

Pour Chacun des aspects nous montrerons comment :

- Tester
- Mettre en place les bonnes pratiques

5.1 - Mise en place du serveur

TODO 01

Dans le projet resanet-eureka-serveur aller dans le starter Application.java et ajouter l'annotation pour avoir un serveur eureka

TODO 02

Dans la configuration de l'application, src/main/resources/application.yml, configurer un comportement standalone du serveur

```
    registerWithEureka: false
    fetchRegistry: false
```

NB 1 : Eureka même en standalone est fortement résilient et faut-tolérent ce qui en fait un bon choix. On peut vouloir ainsi supprimer le côté "client" du serveur ie aller rechercher les autres instances de serveur (peers). C'est ce qui est fait dans le TODO 02

NB 2 : La service-url pointe vers le même hôte.

TODO 03 : Lancer le serveur

Dans le projet faire :

```
mvn spring-boot:run
```

TODO 04 : Voir le client web du serveur qui donner l'état du cluster

aller sur http://localhost:/

NB : Nous avons choisi 9000 comme port. cf application.yml

Donc pointer vers http://localhost:9000/

Vérifier qu'il n'y a pas de client enregistré.

** ---------- FIN PARTIE SERVEUR----------**

5.2 Enregistrement des clients

TODO 05

Importer les dépendances adéquates pour enregistrer resanet-catalogue et resanet-reservations auprès de resanet-eureka-serveur

TODO 05 - a dans resanet-catalogue

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-netflix</artifactId>
                <version>1.0.7.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

TODO 05 - b

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

TODO 05 - c faire de même dans **resanet-reservations**

TODO 06

Ajouter les annotations pour faire de resanet-catalogue et resanet-reservations des clients du serveur eureka mis en place

Dans com.zenika.microservices.resanet.catalog.starter.Application mettre les annotations adéquates

```
@EnableEurekaClient
```

TODO 07

Configurer le serveur préalablement mis en place comme registry

Dans application.yml

```
eureka:
  client:
    serviceUrl:
      defaultZone: <url du serveur mis en place>
```

Penser aussi à donner un nom à l'application pour qu'elle soit enregistrée avec le nom adéquat. Toujours dans application.yml :

```
spring:
    application:
        name: <nom_application>
```

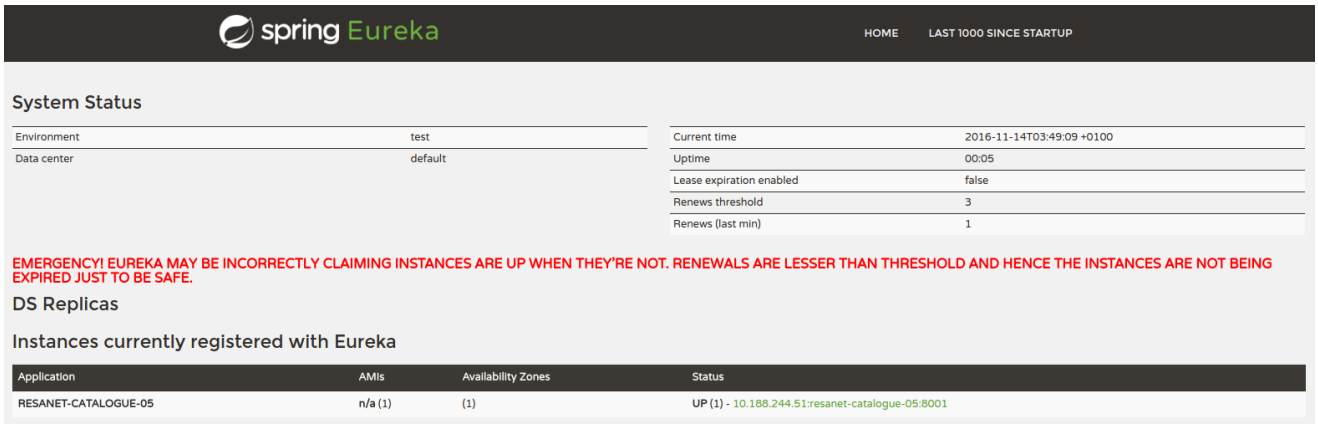
TODO 08

Vérifier que l'application a bien été enregistrée dans le Eureka-Serveur. Pour ce faire ouvrir depuis un browser : http://localhost:9000

Dans le menu *Instances currently registered with Eureka*

Il doit y avoir :

- RESANET-CATALOGUE-05
- RESANET-RESERVATION-05



TODO 09

Le message suivant relève d'un réglage côté client.

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

En effet, il y a tout un tas de paramètres disponibles avec Eureka. Ceux-ci sont définis dans la classe [EurekaInstanceConfigBean (https://github.com/spring-cloud/spring-cloud-netflix/blob/master/spring-cloud-netflix-eureka-client/src/main/java/org/springframework/cloud/netflix/eureka/EurekaInstanceConfigBean.java)]

Parmis ces derniers, il y a la façon dont le client envoie des heartbeats au serveur pour valider le fait qu'il est UP. Entrent en compte :

- `leaseRenewalIntervalInSeconds` (défaut à 30 secondes) : donne la durée d'envoi des heartsbeats.
- `leaseExpirationDurationInSeconds` (défaut à 90 secondes) : donne la durée de non réception maximale des heartsbeats. Si ce seuil est dépassé alors le serveur considère l'instance comme down. **il est conseillé d'avoir une valeur supérieure à `leaseRenewalIntervalInSeconds`**

Il faut donc ajouter la ligne suivante au fichier application.yml :

```
eureka:
 client:
   serviceUrl:
     defaultZone: <url du serveur mis en place>
   leaseRenewalIntervalInSeconds: 10
```

5.3 (Optionnel) Découverte de services

Utiliser la découverte de service dans un Test unitaire

5.4 (Optionnel) Load balancer

Utiliser Ribbon pour faire du load balancing

** ---------- FIN PARTIE CLIENT----------**