
TP N°5 :

Spring Cloud : Zuul API Gateway

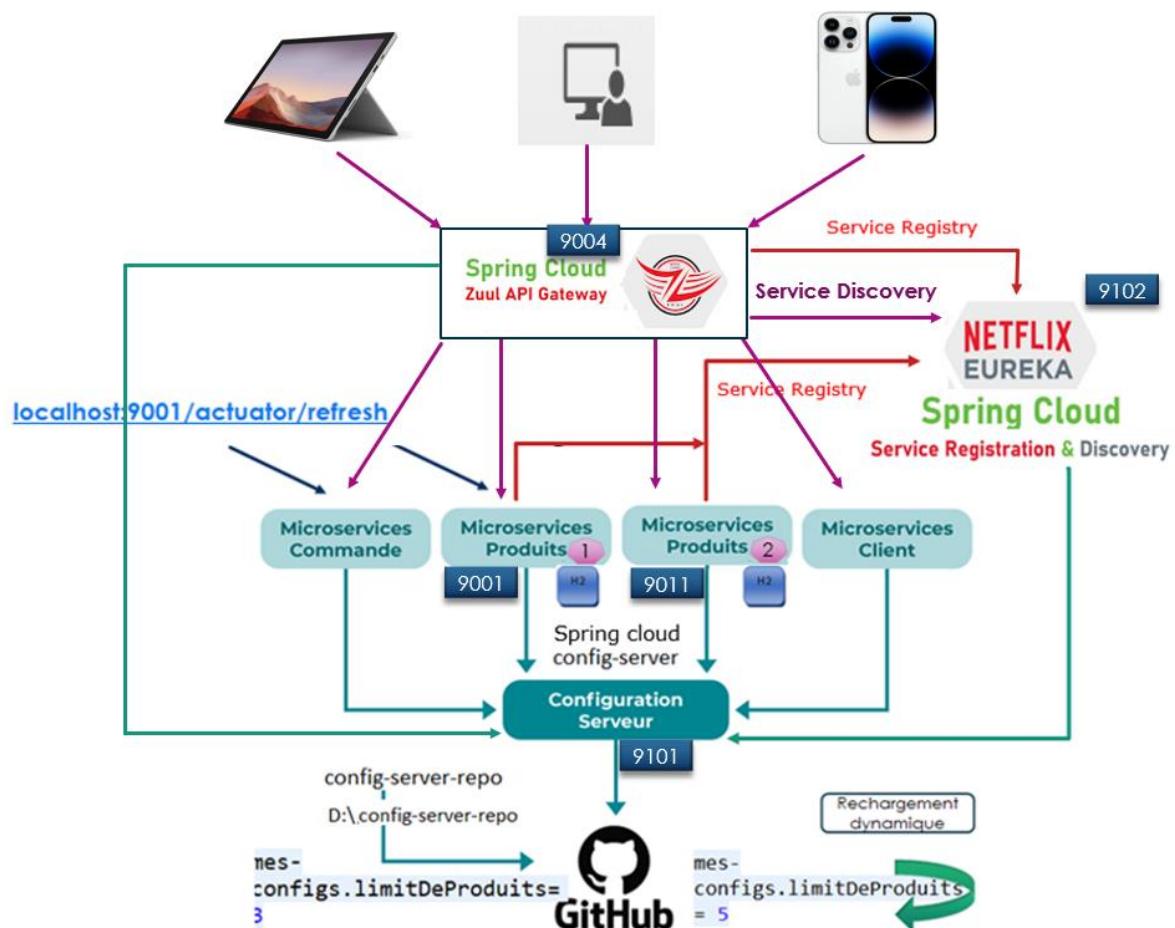
1. Prérequis

- TP3.1 Spring Cloud Config : toute fois ce n'est pas nécessaire d'avoir le Spring Cloud Config Server
- TP4.1 Spring Cloud Eureka Server : toute fois ce n'est pas nécessaire d'avoir le Spring Cloud Eureka 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 `@EnableZuulProxy`
3. Service Registry :
 - a. Annotation `@EnableDiscoveryClient`
 - b. Enregistrement de Zuul sur Eureka
 - c. Clonage d'un microservice et enregistrement auprès de Eureka Server
4. Service Discovery via Zuul API
5. Load Balancing
6. Gestion des filtres : class `ZuulFilter` de Netflix

3. Architecture de mise en œuvre

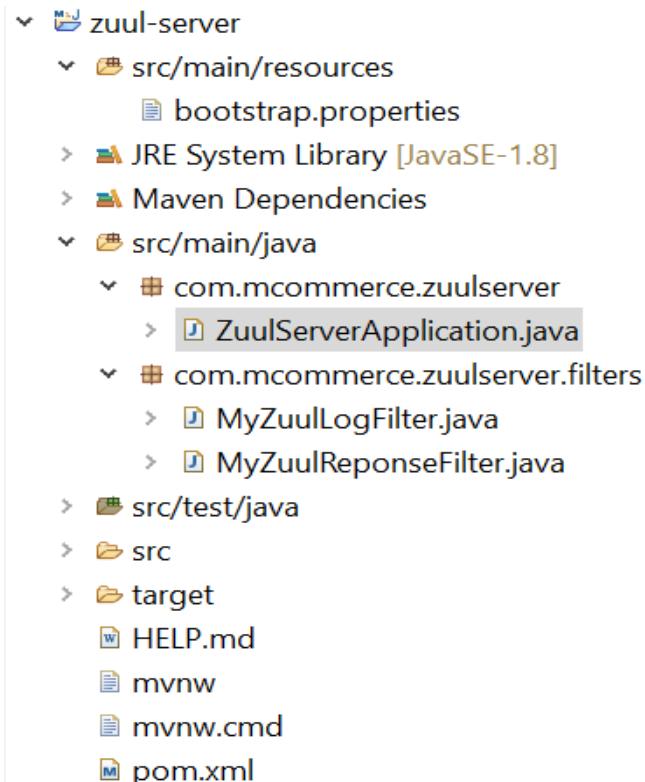


- Un serveur Spring cloud Zuul
- Deux instances du micro-service-produit qui s'enregistrent auprès de Eureka
- Remarquer que Eureka Server et Zuul Server s'enregistrent eux même auprès de Spring cloud config Server à l'instar des autres microservices applicatifs (Produit, commande, client)

4. Mise en place de l'API Gateway Zuul

The screenshot shows the Spring Initializr web application. On the left, under 'Project', 'Language' is set to Java. Under 'Spring Boot', version 2.7.17 is selected. In the 'Project Metadata' section, the group is com.mcommerce, artifact is zuul-server, name is zuul-server, description is 'Demo project for Spring Boot', package name is com.mcommerce.zuul-server, and packaging is Jar. On the right, under 'Dependencies', Eureka Discovery Client and Config Client are selected. A button 'ADD DEPENDENCIES... CTRL + B' is visible.

- Arborescence du projet Eclipse « zuul-server »



- Modifier le fichier « pom.xml » généré comme suit :

```

<?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>2.1.3.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.mcommerce</groupId>
  <artifactId>zuul-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>zuul-server</name>
  <description>Demo project for Spring Boot</description>
  <properties> <java.version>1.8</java.version><spring-cloud.version>Greenwich.RELEASE</spring-cloud.version></properties>
  <dependencies>

    <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-client</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
    </dependency>

    <dependency><groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope></dependency>
    <!-- Add the following dependency to avoid : No spring.config.import property has been defined -->
    <dependency><groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-bootstrap</artifactId><version>4.0.4</version>
    </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 « zuul-server »:

```
spring.application.name=zuul-server  
spring.cloud.config.uri=http://localhost:9101
```

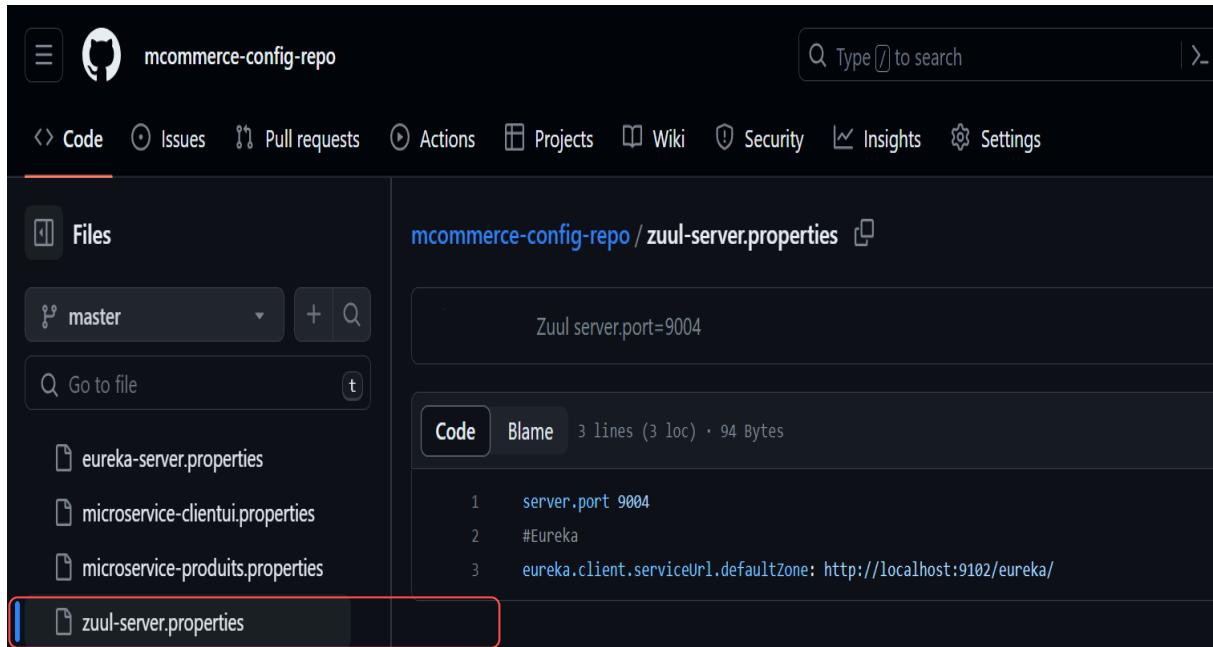
- b. Le fichier « `zuul-server.properties` » qui sera synchronisé avec Github :

```
server.port=9004  
#L'API Gateway Zuul s'enregistre auprès de Eureka  
eureka.client.serviceUrl.defaultZone=http://localhost:9102/eureka/
```

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



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



c. La Classe principale `com.mcommerce.zuulserver.ZuulServerApplication`:

```
package com.mcommerce.zuulserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
@EnableDiscoveryClient

public class ZuulServerApplication {

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

d. Vérifier le bon démarrage de L'API Gateway Zuul :

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

Ne pas oublier de démarrer par la suite Spring Cloud Eureka Server qui écoute sur le port 9102.

The Spring Boot logo consists of a grid of characters. The top row contains two backslashes followed by a single slash. The second row contains two left parentheses followed by a right parenthesis. The third row contains two left slashes followed by a right slash. The fourth row contains two left brackets followed by a right bracket. The fifth row contains two left braces followed by a right brace. The bottom row contains three equals signs followed by three forward slashes and three backslashes. To the right of the grid, the text 'Spring Boot' is written in a bold font, followed by '(v2.1.3.RELEASE)' in a smaller font.

```
Fetching config from server at : http://localhost:9101
Located environment: name=zuul-server, profiles=[default], label=null,
{name='https://github.com/XXX/mcommerce-config-repo.git/zuul-
server.properties'}]}
c.m.zuulserver.ZuulServerApplication : No active profile set, falling
back to default profiles: default
Tomcat initialized with port(s): 9004 (http)
o.s.c.n.zuul.ZuulFilterInitializer : Starting filter initializer
o.s.b.a.e.web.EndpointLinksResolver: Exposing 2 endpoint(s) beneath
base path '/actuator'
eureka.InstanceInfoFactory:Setting initial instance status as:STARTING
com.netflix.discovery.DiscoveryClient: Initializing Eureka in region
us-east-1
com.netflix.discovery.DiscoveryClient: Getting all instance registry
info from the eureka server
com.netflix.discovery.DiscoveryClient : The response status is 200
com.netflix.discovery.DiscoveryClient : Starting heartbeat executor:
renew interval is: 30
c.n.discovery.InstanceInfoReplicator : InstanceInfoReplicator
onDemand update allowed rate per min is 4
com.netflix.discovery.DiscoveryClient : Discovery Client initialized
at timestamp 1700937401774 with initial instances count: 0
o.s.c.n.e.s.EurekaServiceRegistry: Registering application ZUUL-SERVER
with eureka with status UP
com.netflix.discovery.DiscoveryClient: Saw local status change event
StatusChangeEvent [timestamp=1700937401781, current=UP,
previous=STARTING]
[nfoReplicator-0] com.netflix.discovery.DiscoveryClient :
DiscoveryClient_ZUUL-SERVER/local:zuul-server:9004: registering
service...
oembedded.tomcat.TomcatWebServer : Tomcat started on port(s): 9004
s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 9004
zuulserver.ZuulServerApplication : Started ZuulServerApplication
in 9.123 seconds (JVM running for 9.505)
[nfoReplicator-0] com.netflix.discovery.DiscoveryClient :
DiscoveryClient_ZUUL-SERVER/local:zuul-server:9004 - registration
status: 204
com.netflix.discovery.DiscoveryClient : Getting all instance registry
info from the eureka server
com.netflix.discovery.DiscoveryClient : The response status is 200
```

18:41:41.467 c.n.d.s.r.aws.ConfigClusterResolver : Resolving

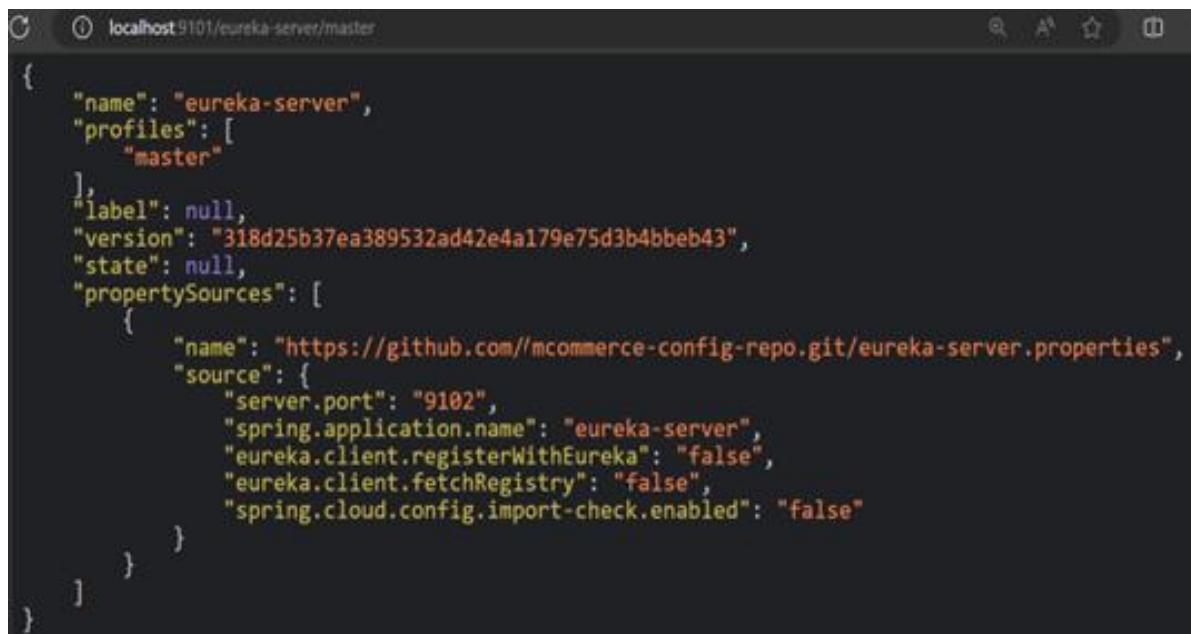
```

eureka endpoints via configuration
18:46:41.479 c.n.d.s.r.aws.ConfigClusterResolver      : Resolving
eureka endpoints via configuration
18:51:41.489 c.n.d.s.r.aws.ConfigClusterResolver      : Resolving
eureka endpoints via configuration
18:56:41.490 c.n.d.s.r.aws.ConfigClusterResolver      : Resolving
eureka endpoints via configuration
19:01:41.495 c.n.d.s.r.aws.ConfigClusterResolver      : Resolving
eureka endpoints via configuration
19:06:41.498 c.n.d.s.r.aws.ConfigClusterResolver      : Resolving
eureka endpoints via configuration

2023-11-25 19:20:54.758 [Tomcat].[localhost].[/]      : Initializing
Spring DispatcherServlet 'dispatcherServlet'
2023-11-25 19:20:54.758 o.s.web.servlet.DispatcherServlet      :
Initializing Servlet 'dispatcherServlet'
2023-11-25 19:20:54.777 o.s.web.servlet.DispatcherServlet      :
Completed initialization in 19 ms
2023-11-25 19:20:54.806 o.s.c.n.zuul.web.ZuulHandlerMapping      : No
routes found from RouteLocator
2023-11-25 19:21:41. c.n.d.s.r.aws.ConfigClusterResolver      :
Resolving eureka endpoints via configuration

```

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



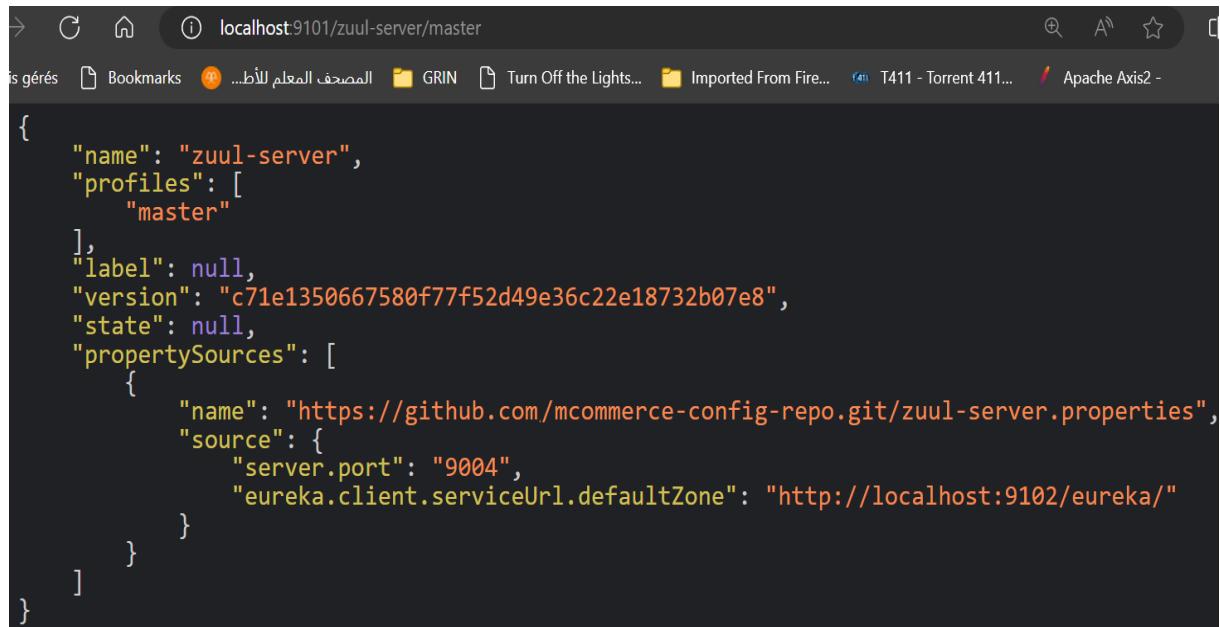
The screenshot shows a browser window with the URL <http://localhost:9101/eureka-server/master>. The page displays a JSON object representing the configuration for the Eureka Server. The JSON structure includes fields for name, profiles, label, version, state, and propertySources.

```

{
  "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"
      }
    }
  ]
}

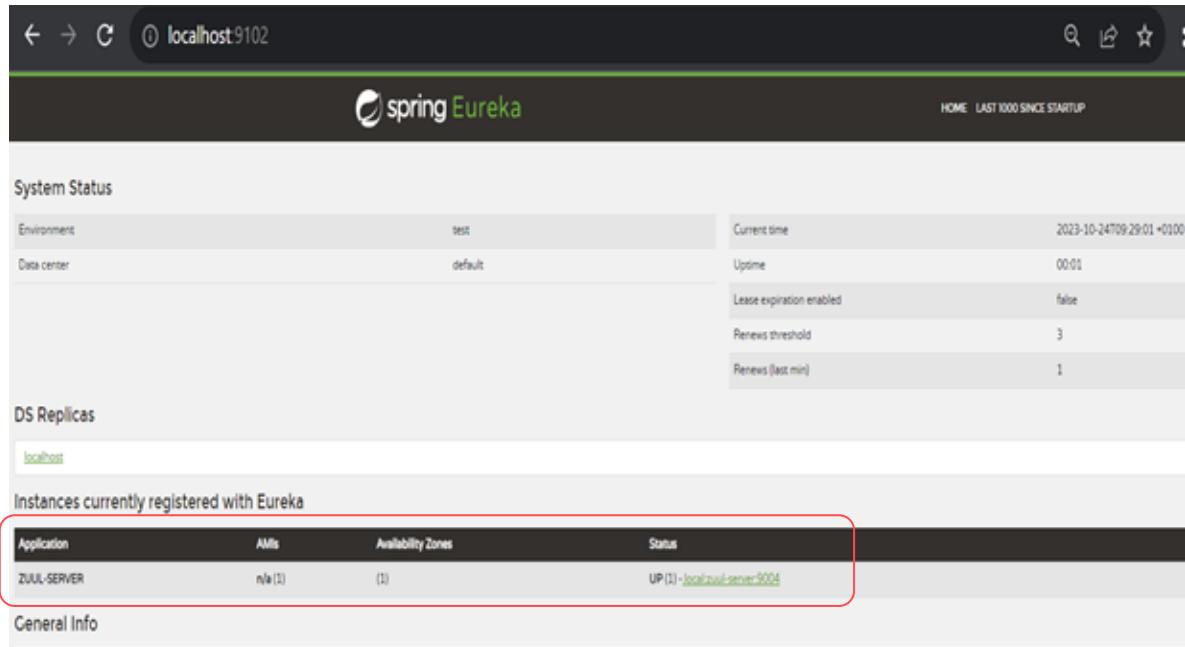
```

- f. Vérifier que Zuul Server a été bien enregistré dans notre Spring Cloud Config Server
<http://localhost:9101/zuul-server/master>



```
{  
  "name": "zuul-server",  
  "profiles": [  
    "master"  
  ],  
  "label": null,  
  "version": "c71e1350667580f77f52d49e36c22e18732b07e8",  
  "state": null,  
  "propertySources": [  
    {  
      "name": "https://github.com/mcommerce-config-repo.git/zuul-server.properties",  
      "source": {  
        "server.port": "9004",  
        "eureka.client.serviceUrl.defaultZone": "http://localhost:9102/eureka/"  
      }  
    }  
  ]  
}
```

- a. Accéder à Eureka Server : <http://localhost:9102/> et vérifier que zuul-server a été bien enregistré dans Eureka Server :



The screenshot shows the Spring Eureka interface at <http://localhost:9102>. It displays the system status and a list of registered instances.

System Status

Environment	test	Current time	2023-10-24T09:29:01+01:00
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renew threshold	3
		Renew (last min)	1

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ZUUL-SERVER	n/a (1)	(1)	UP (1) - localhost:zuul-server:9004

General Info

5. Développement du microservice « Produit »

- a. Le MS-Produits est similaire à celui du TP4.
- b. Fichier « bootstrap.properties » du MS-Produits :

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

- c. Fichier « **microservice-produits** » du MS-Produits pushé sur Github:

```
#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/

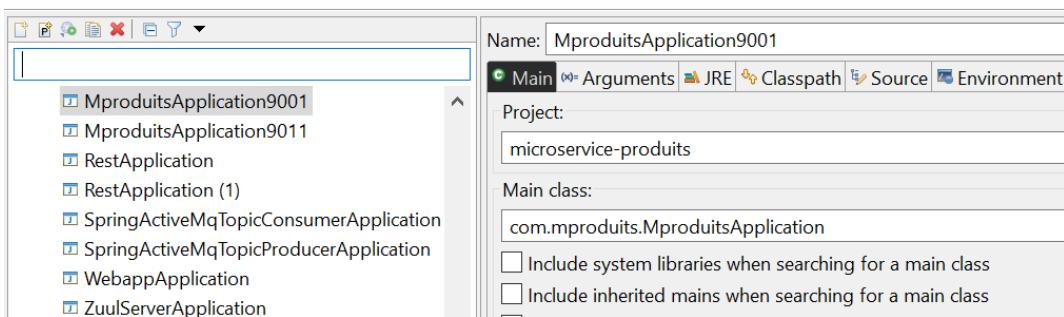
#Actuator : management.endpoints.web.exposure.include=*
management.endpoints.web.expose=info, health, refresh
#Les configurations personnalisés
mes-configs.limitDeProduits= 3
```

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

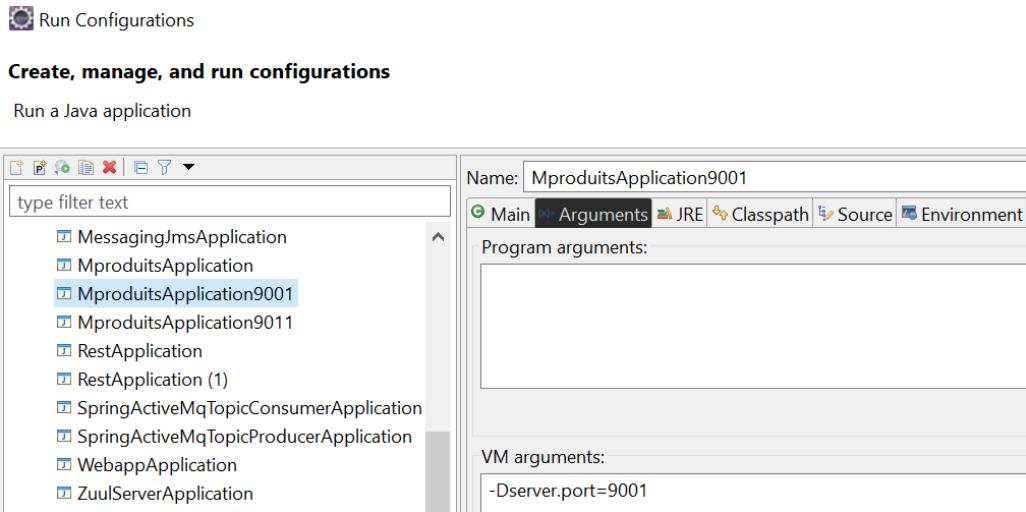


Create, manage, and run configurations

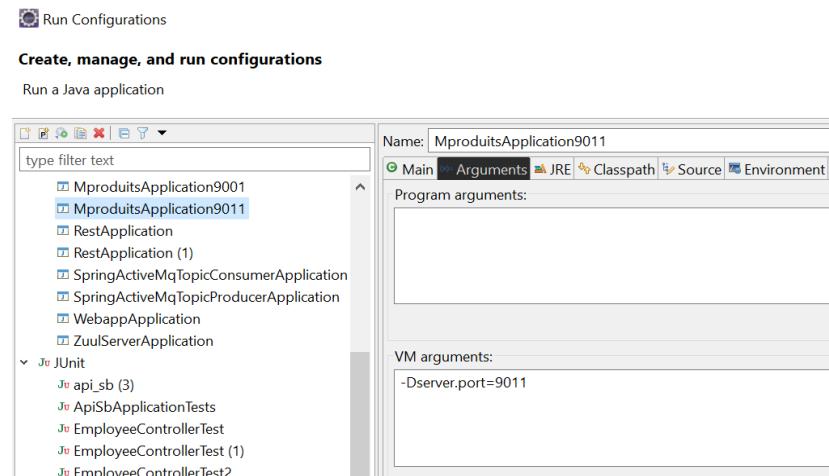
Run a Java application



Utiliser l'argument JVM « -Dserver.port » pour forcer le numéro de port :



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



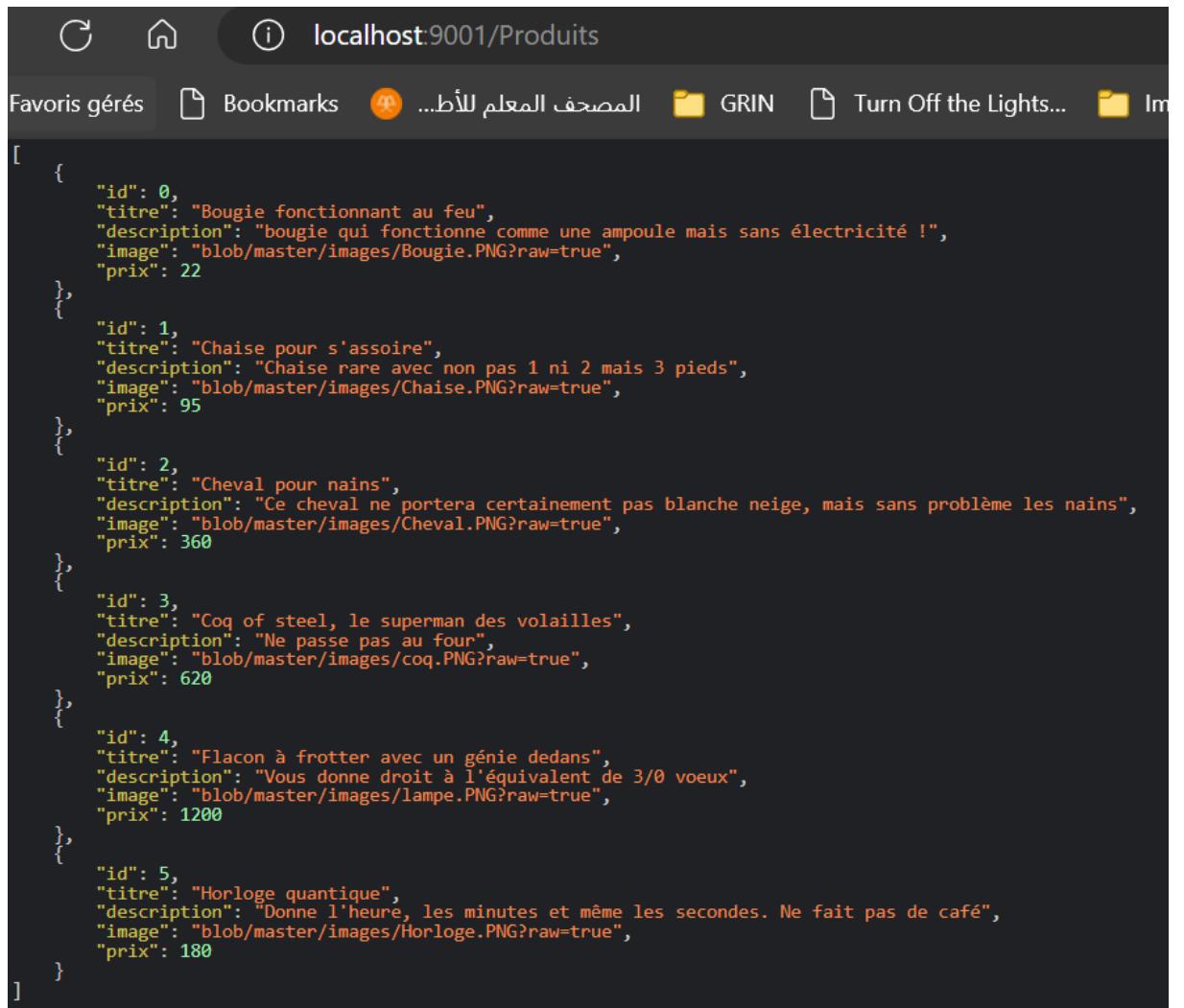
f. Démarrage des instances des micro services Produit sur le port 9001 et 9011

→ <http://localhost:9102/>

Application	AMIs	Availability Zones	Status
MICROSERVICE-PRODUCTS	n/a (2)	(2)	UP (2) - local/microservice:produits:9011:local/microservice:produits:9001
ZUUL-SERVER	n/a (1)	(1)	UP (1) - local/zuul-server:9004

6. Service Discovery avec Zuul

- a. Accès direct au microservice « Produits » sans passer par la Gateway zuul :
<http://localhost:9001/Produits>



```
[{"id": 0, "titre": "Bougie fonctionnant au feu", "description": "bougie qui fonctionne comme une ampoule mais sans électricité !", "image": "blob/master/images/Bougie.PNG?raw=true", "prix": 22}, {"id": 1, "titre": "Chaise pour s'assoir", "description": "Chaise rare avec non pas 1 ni 2 mais 3 pieds", "image": "blob/master/images/Chaise.PNG?raw=true", "prix": 95}, {"id": 2, "titre": "Cheval pour nains", "description": "Ce cheval ne portera certainement pas blanche neige, mais sans problème les nains", "image": "blob/master/images/Cheval.PNG?raw=true", "prix": 360}, {"id": 3, "titre": "Coq of steel, le superman des volailles", "description": "Ne passe pas au four", "image": "blob/master/images/coq.PNG?raw=true", "prix": 620}, {"id": 4, "titre": "Flacon à frotter avec un génie dedans", "description": "Vous donne droit à l'équivalent de 3/0 voeux", "image": "blob/master/images/lampe.PNG?raw=true", "prix": 1200}, {"id": 5, "titre": "Horloge quantique", "description": "Donne l'heure, les minutes et même les secondes. Ne fait pas de café", "image": "blob/master/images/Horloge.PNG?raw=true", "prix": 180}]
```

- b. Accès direct au microservice « Produits » en passant par la Gateway zuul :
<http://localhost:9004/microservice-produits/Produits>
- c. Vous allez remarquer que à chaque appel, une instance du MS-Produits est appelée : il s'agit de l'algorithme de partage de charge par défaut : «round-robin ».

```

[{"id": 0, "titre": "Bougie fonctionnant au feu", "description": "bougie qui fonctionne comme une ampoule mais sans \u00e9lectricit\u00e9 !", "image": "blob/master/images/Bougie.PNG?raw=true", "prix": 22}, {"id": 1, "titre": "Chaise pour s'asseoir", "description": "Chaise rare avec non pas 1 ni 2 mais 3 pieds", "image": "blob/master/images/Chaise.PNG?raw=true", "prix": 95}, {"id": 2, "titre": "Cheval pour nains", "description": "Ce cheval ne portera certainement pas blanche neige, mais sans probl\u00e8me les nains", "image": "blob/master/images/Cheval.PNG?raw=true", "prix": 360}, {"id": 3, "titre": "Coq of steel, le superman des volailles", "description": "Ne passe pas au four", "image": "blob/master/images/coq.PNG?raw=true", "prix": 620}, {"id": 4, "titre": "Flacon \u00e0 frotter avec un g\u00e9nie dedans", "description": "Vous donne droit \u00e0 l'\u00e9quivalent de 3/0 voeux", "image": "blob/master/images/lampe.PNG?raw=true", "prix": 1200}, {"id": 5, "titre": "Horloge quantique", "description": "Donne l'heure, les minutes et m\u00eame les secondes. Ne fait pas de caf\u00e9", "image": "blob/master/images/Horloge.PNG?raw=true", "prix": 180}]

```

d. Remarquer au niveau de la console de zuul server :

```

ConfigClusterResolver : Resolving eureka endpoints via configuration
zuulserver.filters.MyZuulLogFilter : **** MyZuulLogFilter :
Requ\u00eate intercept\u00e9e ! L'URL est :
http://localhost:9004/microservice-produits/
netflix.config.ChainedDynamicProperty : Flipping property:
microservice-produits.ribbon.ActiveConnectionsLimit to use NEXT property:
loadbalancer.availabilityFilteringRule.activeConnectionsLimit =
2147483647
installed for: NFLoadBalancer-PingTimer-microservice-produits
c.netflix.loadbalancer.BaseLoadBalancer : Client:
microservice-produits instantiated a LoadBalancer:
DynamicServerListLoadBalancer:{NFLoadBalancer:name=microservice-
-produits,
DynamicServerListLoadBalancer:{NFLoadBalancer:name=microservice-
-produits, current list of

```

```

Servers=[local:9011,local:9001],Load balancer
stats=Zone stats: {defaultzone=[Zone:defaultzone;      Instance
count:2;  Active connections count: 0;      Circuit breaker]
},Server stats: [[local:9001;      Zone:defaultZone;
Total Requests:0;      Successive connection failure:0;
Total blackout seconds:0;  Last connection made:Thu Jan 01
00:00:00 WET 1970;  First connection made: Thu Jan 01
00:00:00 WET 1970;  Active Connections:0; total failure count
in last (1000) msecs:0;  average resp time:0.0;      90
percentile resp time:0.0;  95 percentile resp time:0.0;      min
resp time:0.0;  max resp time:0.0;      stddev resp time:0.0]
, [local:9011;  Zone:defaultZone;  Total Requests:0;
Successive connection failure:0;Total blackout seconds:0;

]}ServerList:org.springframework.cloud.netflix.ribbon.eureka.Do
mainExtractingServerList@f0ee4e7
2023-11-25 19:55:34.763  INFO 12928 --- [nio-9004-exec-2]
c.m.z.filters.MyZuulReponseFilter      : ***
MyZuulReponseFilter :  CODE HTTP 400  ****

```

En résumé :

- **ZUUL** fonctionne nativement avec Eureka: Il récupère la liste de tous les microservices disponibles dans Eureka, et les expose via l'URL
<http://localhost:9004/nom-du-microservice>
- Pour récupérer la liste des produits : <http://localhost:9004/microservice-produits/Produits>
- On récupère la liste des produits, comme si on appelle les Microservice-produits directement.

7. Gestion des filtres avec Zuul

- Les filtres peuvent être des classes Java qui vont étendre **ZuulFilter** :
- Créer un package ‘**filters**’ au sein de service-gateway qui contiendra un sous-package par type de filtre :

```

package com.mcommerce.zuulserver.filters;
import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;

```

```

import com.netflix.zuul.exception.ZuulException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import javax.servlet.http.HttpServletRequest;
@Component
public class MyZuulLogFilter extends ZuulFilter
{
    Logger log = LoggerFactory.getLogger(this.getClass());
    @Override
    public String filterType() {
        // pre : contiendra les filtres exécutés avant le routage d'une requête (règles de pré-routage).
        return "pre";
    }
    @Override
    public int filterOrder() { return 1; }
    @Override
    public boolean shouldFilter() { return true; }
    @Override
    public Object run() throws ZuulException {
        HttpServletRequest req = RequestContext.getCurrentContext().getRequest();
        log.info("***** MyZuulLogFilter : Requête interceptée ! L'URL est : {} ", req.getRequestURL());
        return null;
    }
}

```

```

package com.mcommerce.zuulserver.filters;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import javax.servlet.http.HttpServletResponse;

@Component
public class MyZuulReponseFilter extends ZuulFilter{

```

```

Logger log = LoggerFactory.getLogger(this.getClass());

@Override
public String filterType() {

    //post : contiendra les filtres exécutés après le routage d'une requête (règles
    de post-routage).
    return "post";
}

@Override
public int filterOrder() {      return 1;      }

@Override
public boolean shouldFilter() {   return true; }

@Override
public Object run() throws ZuulException {
HttpServletResponse response =
RequestContext.getCurrentContext().getResponse();

    response.setStatus(400);

    log.info(" *** MyZuulReponseFilter :   CODE HTTP {}  *** ",
response.getStatus());

    return null;
}
}

```

- Remarquer au niveau de la console l'application et l'ordre des filtres lors de l'exécution des MS-Produits.
- Ce filtre récupère toutes les réponses et change le code en 400.
- Redémarrez ZUUL et envoyez via Postman des requêtes vers n'importe quel microservice ; vous recevrez invariablement un code « HTTP 400 Bad Request » :
<http://localhost:9004/microservice-produits/Produits>
- N'oubliez pas de désactiver ce filtre (**shouldFilter à false**), pour qu'il ne vous bloque pas par la suite

GET http://localhost:9004/microservice-produits/Produits

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results

400 Bad Request 24 ms 1.22 KB Save as example ...

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 0,
4     "titre": "Bougie fonctionnant au feu",
5     "description": "bougie qui fonctionne comme une ampoule mais sans électricité !",
6     "image": "blob/master/images/Bougie.PNG?raw=true",
7     "prix": 22.0
8   },
9   {
10    "id": 1,
11    "titre": "Chaise pour s'asseoir",
12    "description": "Chaise rare avec non pas 1 ni 2 mais 3 pieds",
13    "image": "blob/master/images/Chaise.PNG?raw=true",
14    "prix": 95.0
  }
]
  
```

zuulserver.filters.MyZuulLogFilter : **** MyZuulLogFilter : Requête interceptée ! L'URL est :
<http://localhost:9004/microservice-produits/Produits>

z.filters.MyZuulReponseFilter : *** MyZuulReponseFilter : CODE HTTP 400 ****

: Flipping property: microservice-produits.ribbon.ActiveConnectionsLimit to use NEXT

8. Annexe

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

- 3 Edges Services : Spring cloud Config Server / Eureka Server / Zuul Server
- 2 instances du microservice-produits

