

*Institut National des Postes et*

*Télécommunications*

*Microservices Architecture*

**Mini-Projet : Microservices Architecture**

*Réalisé par:*

Mme.EL MADANI Sanae

Mme.REDOUAN Salma

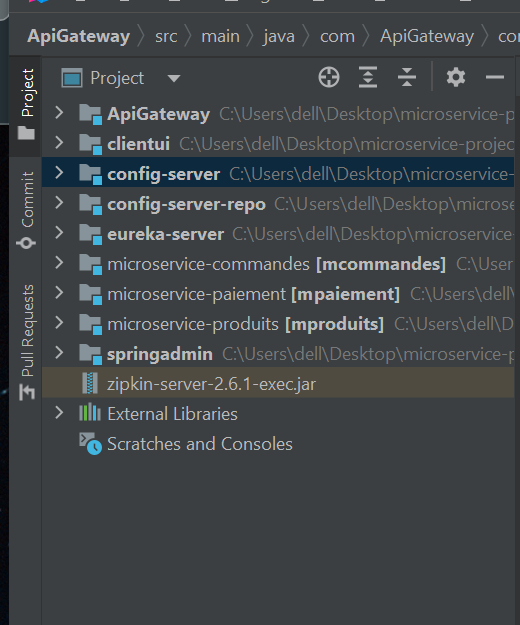
*Encadré par :*

*Professeur .Driss allaki*

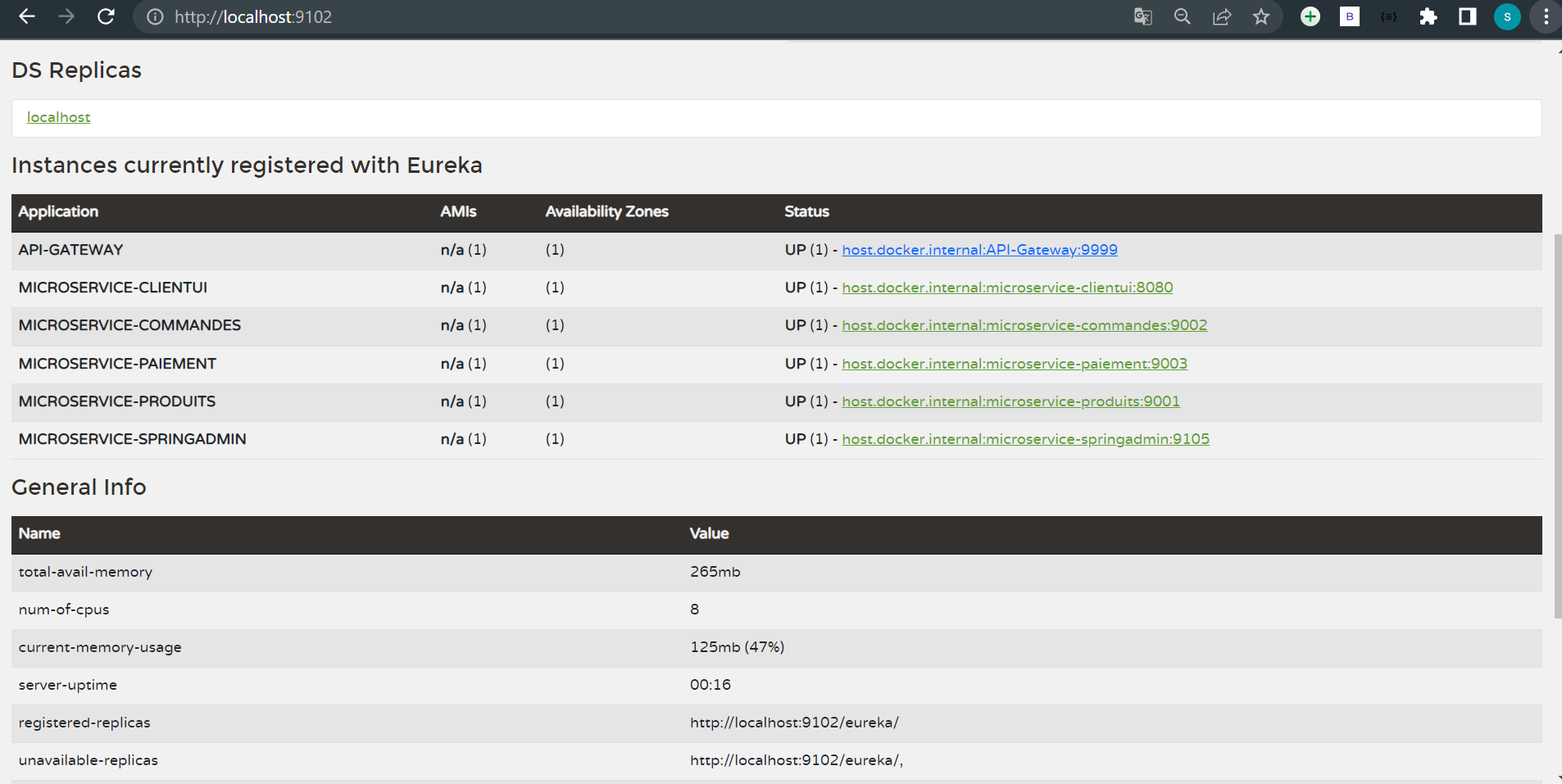
1-Hands-on Lab

Nous avons réalisé les 10 parties du "Hands-on Lab" (projet mcommerce) tout en modernisant bien sur les parties "outdated" moyennant des librairies plus récentes.

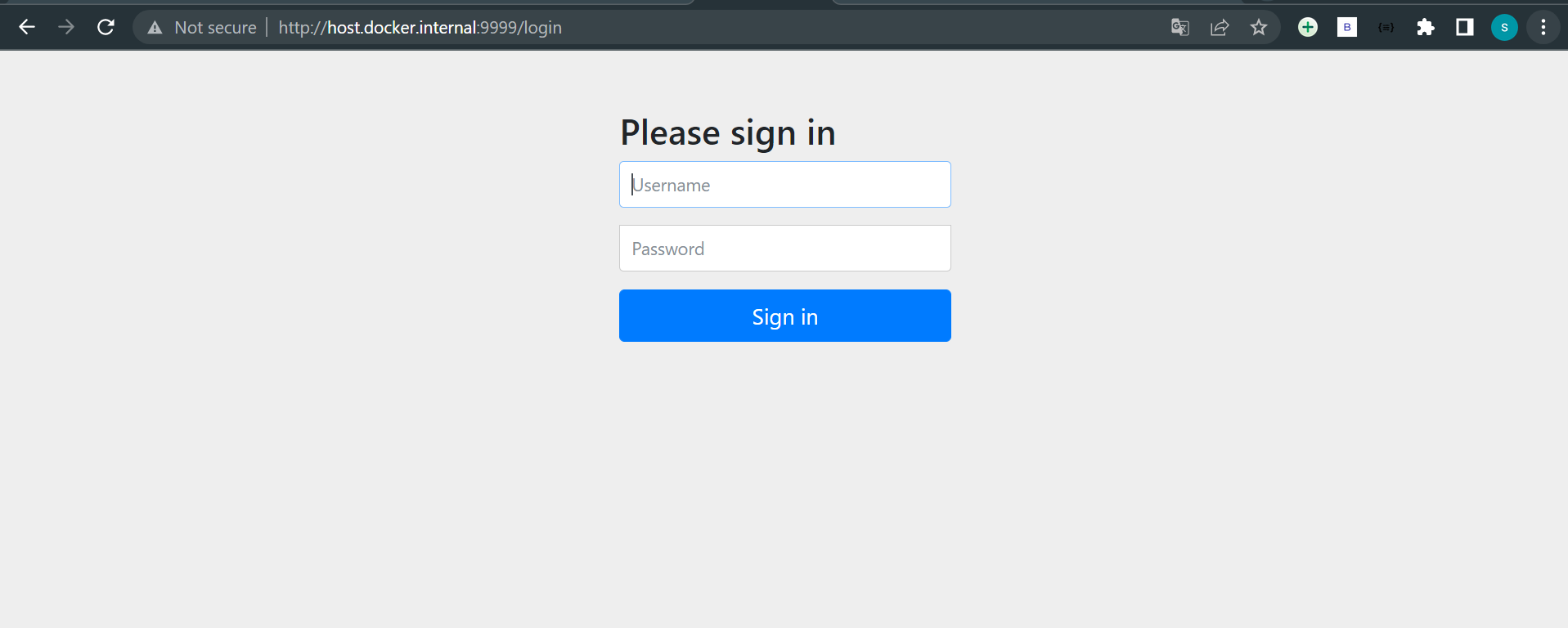
1-Les microservices de projet et le fichier de configuration distant:



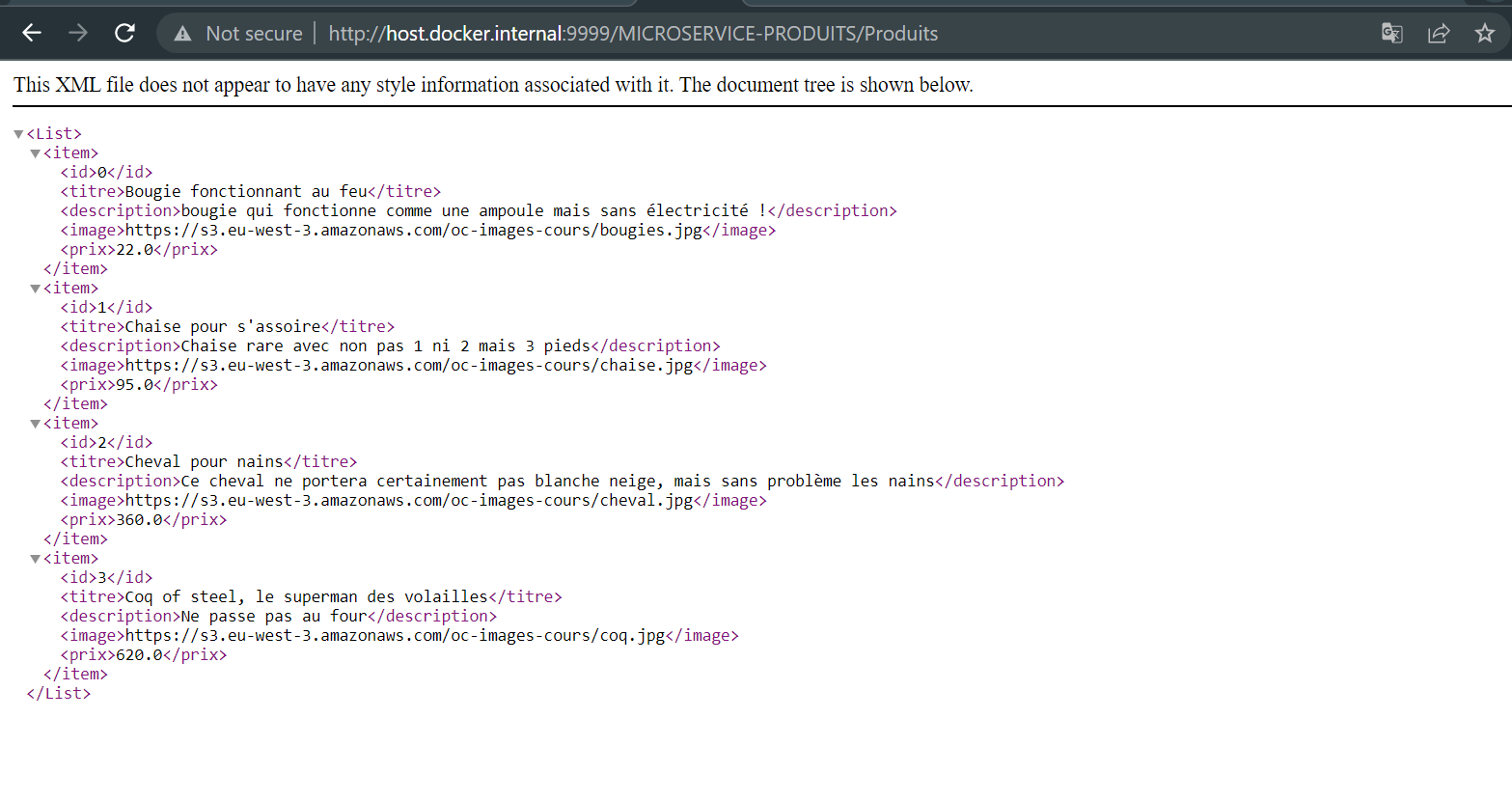
2-Eureka-service



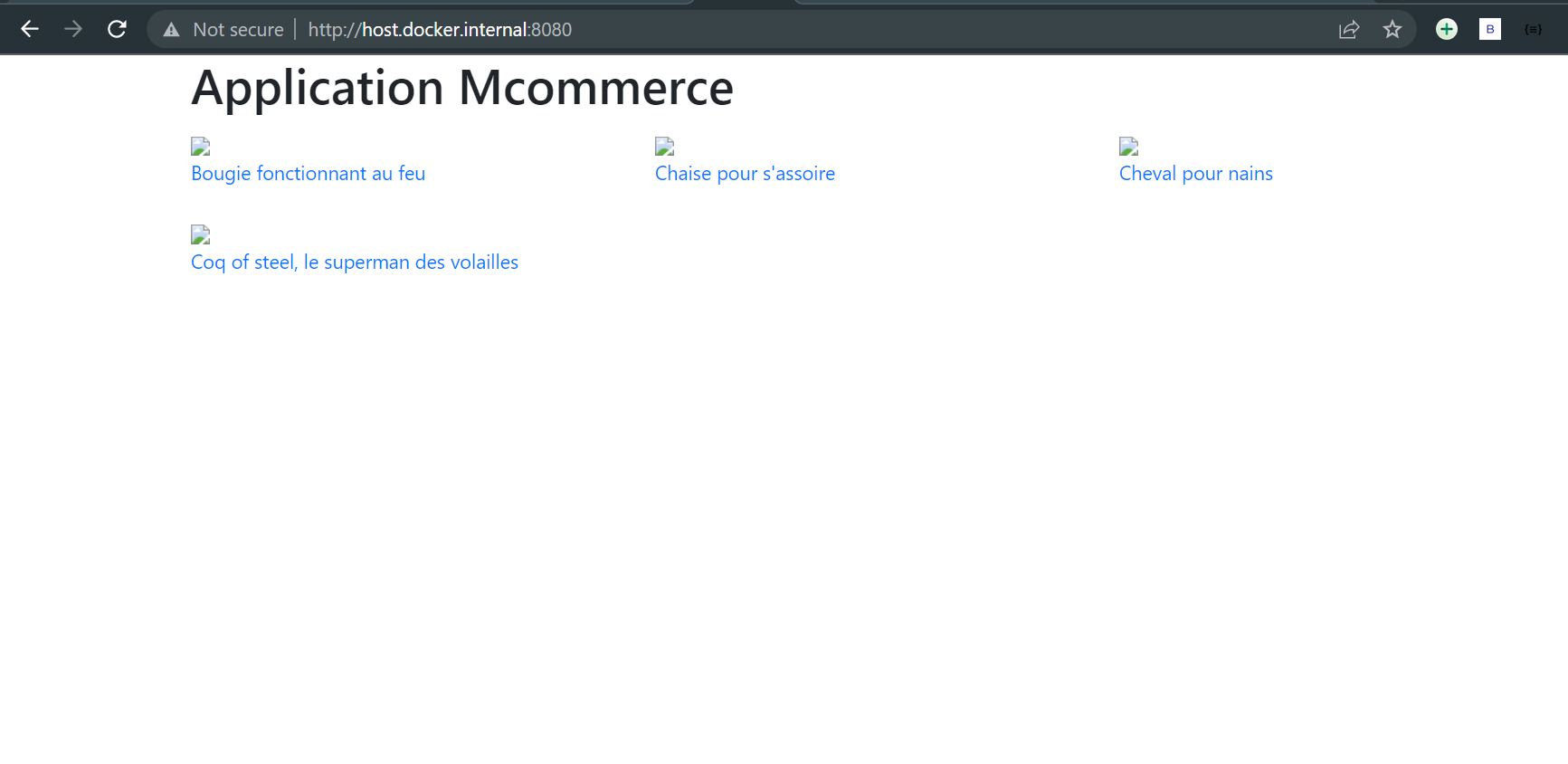
3-Login



4-List des produits sous format json



5-Interface client :

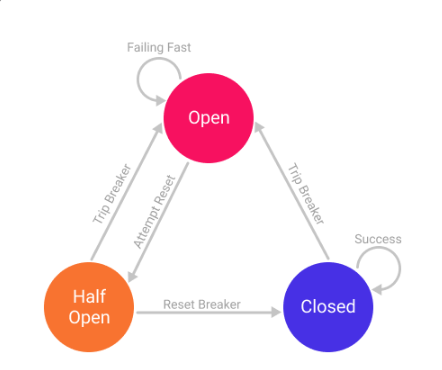


Voila le lien vers le projet réalisé:

<https://github.com/sanae-elmadani/Microservices-Ecomerce-Project/tree/975aef56621da4bb97e3d743a4242db1a3d84530>

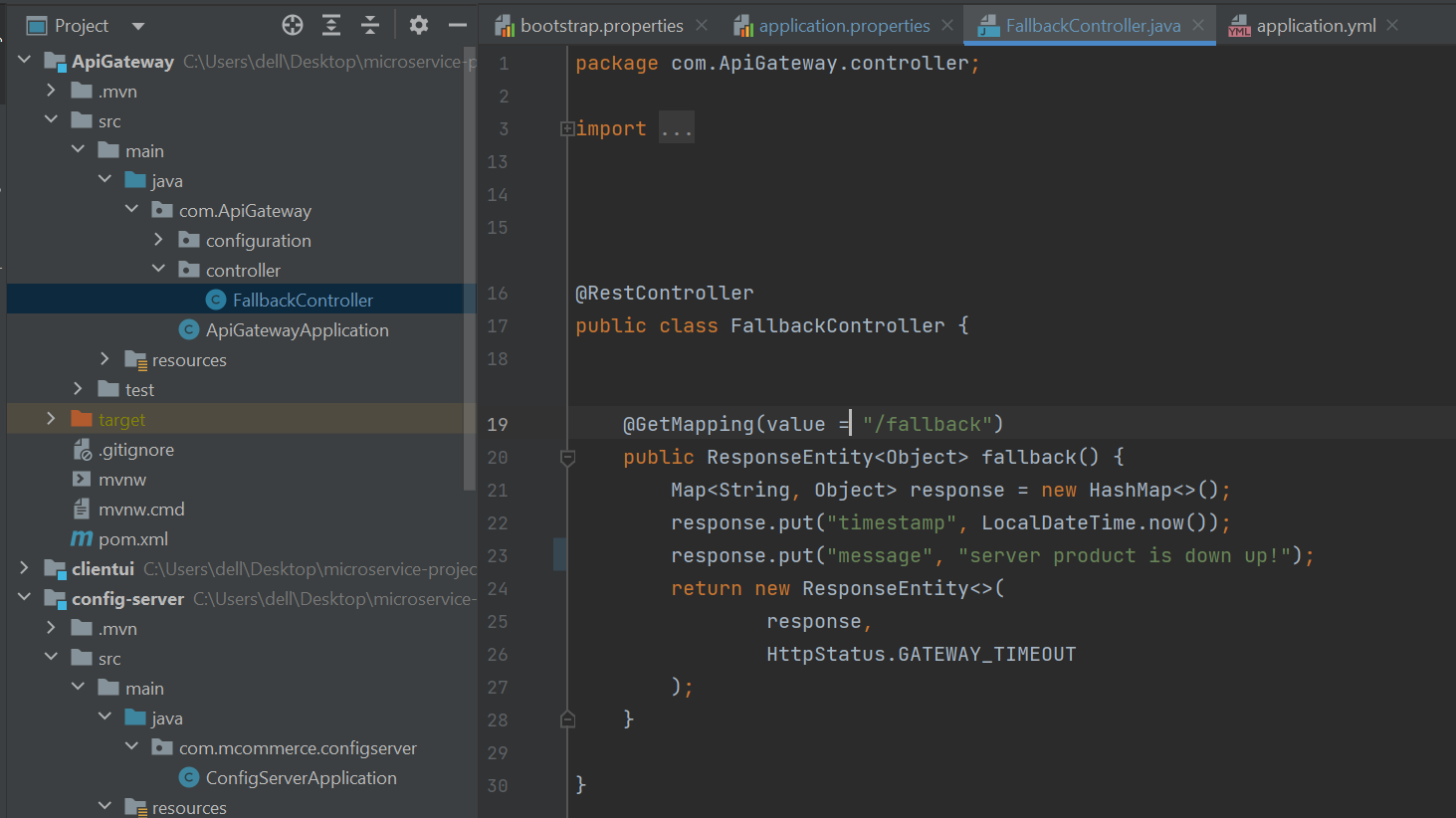
2- Implémentation du circuit breaker (côté applicatif) au niveau de l'API Gateway.

Un circuit breaker est un solution pour gérer les pannes dans les microservices et est maintenant largement implémenté à la fois en tant que bibliothèque et en tant que modèle intégré dans les modules de service et de client.



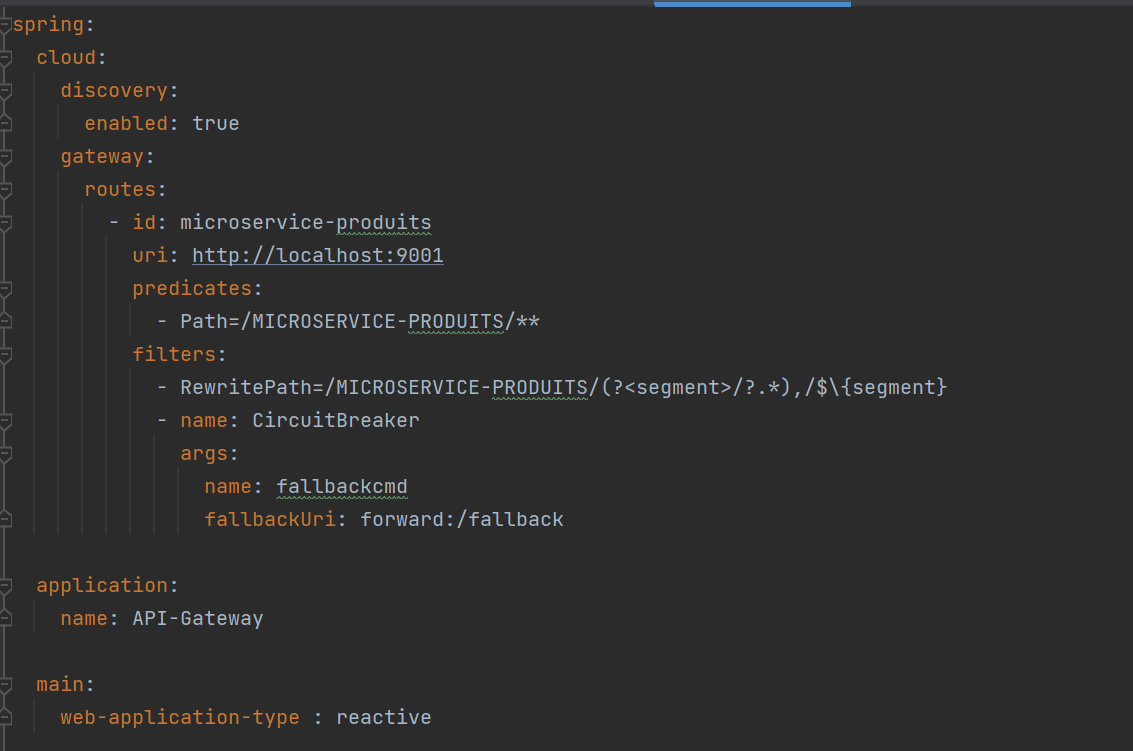
Dans notre cas, on va implémenter le circuit breaker au niveau de l’API Gateway pour se faire :

1-Nous avons créé un FallbackController et ApiResponse pour gérer l'indisponibilité du backend du service.

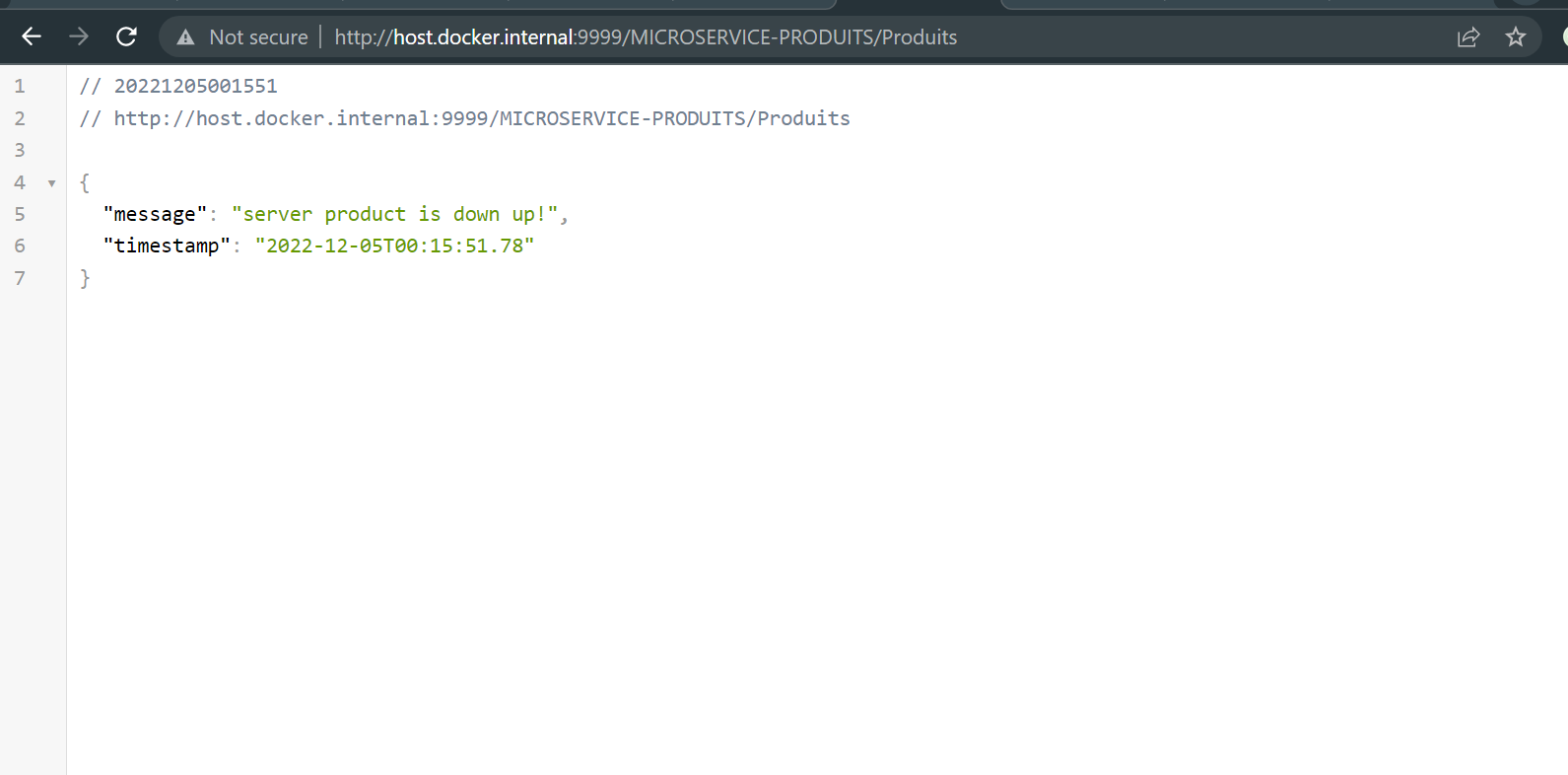


Ainsi, cela signifie que lorsque le service backend n'est pas disponible, il sera redirigé vers /fallback dès que possible.

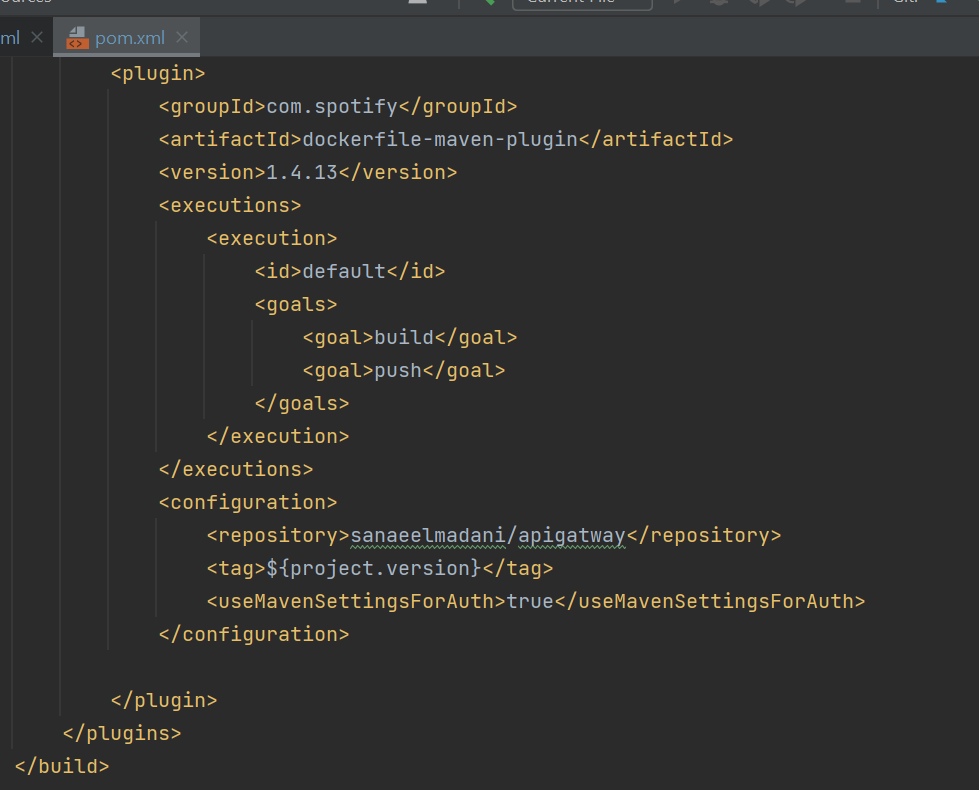
2- Après nous avons ajouté CircuitBreaker à application.yml comme suit ci-dessous.



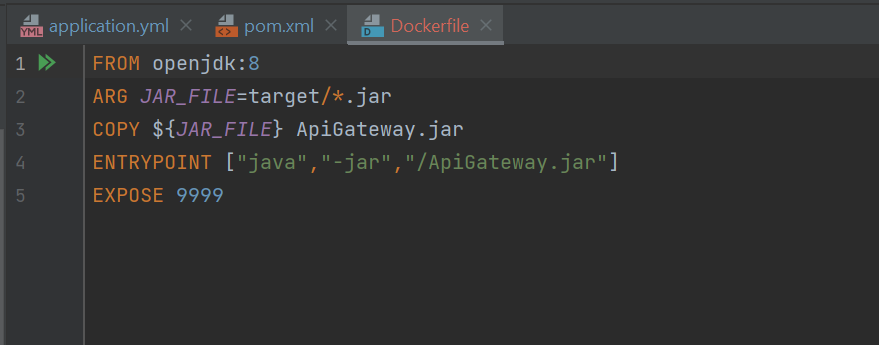
3-Finalement nous avons réussi l'implémentation de circuite breaker au niveau de l’API gateway:



3-Déploiement de l'application microservices commerce sur Kubernetes :

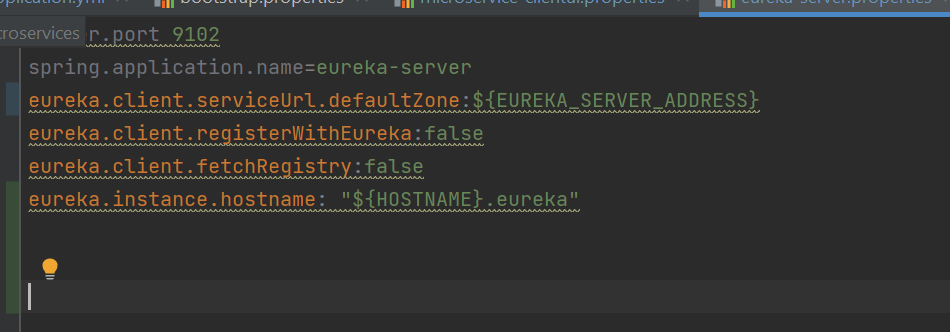
1-L’ajout de dockerfile plugin pour chaque microservices mais prenons l’exemple de API Gateway:

2-Création de Dockerfile

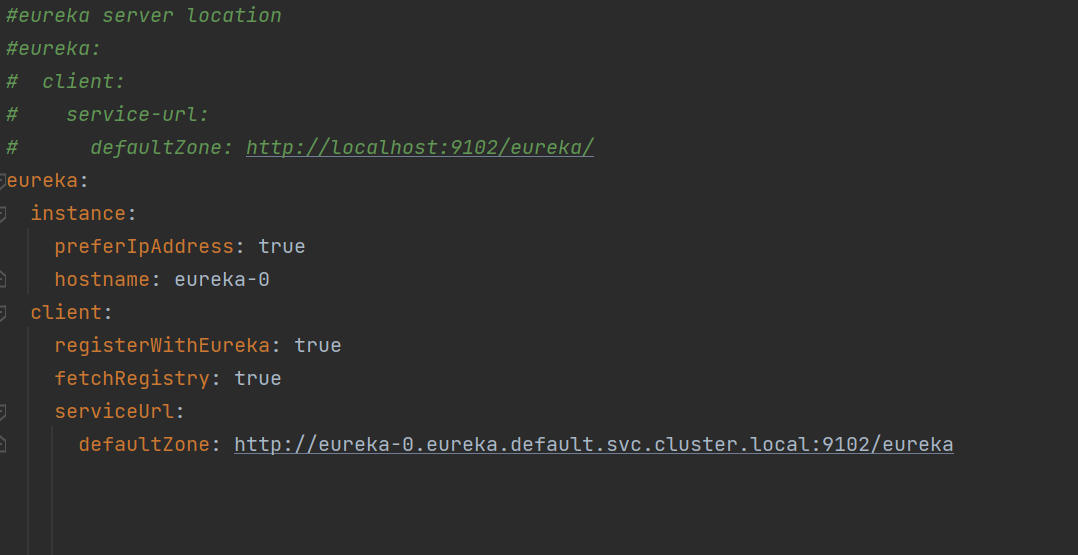


3- L’ajout de configurations pour Service Registry

Pour notre service eureka nous avons ajouté cette configuration:

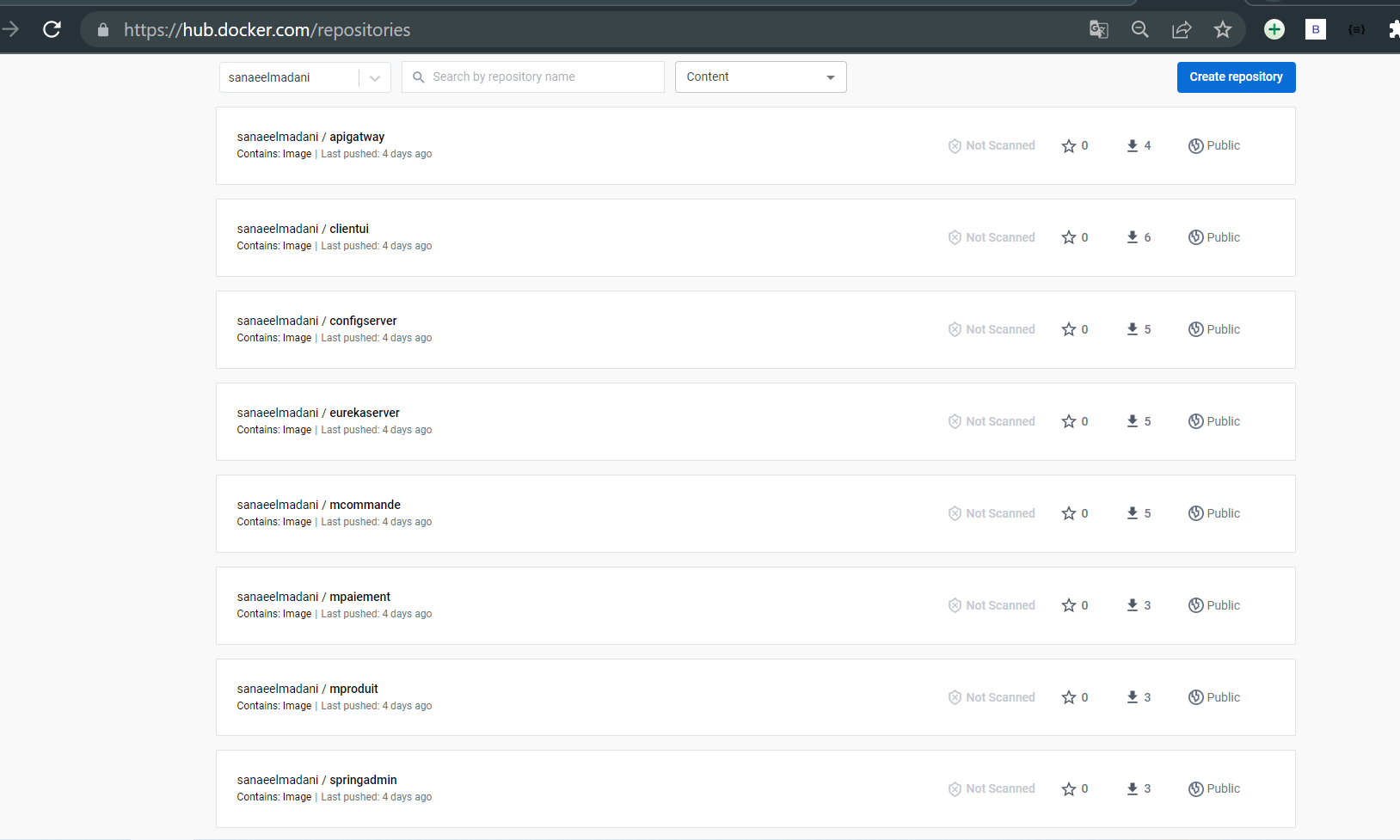


pour les autres microservices nous avons ajouté cette configuration :

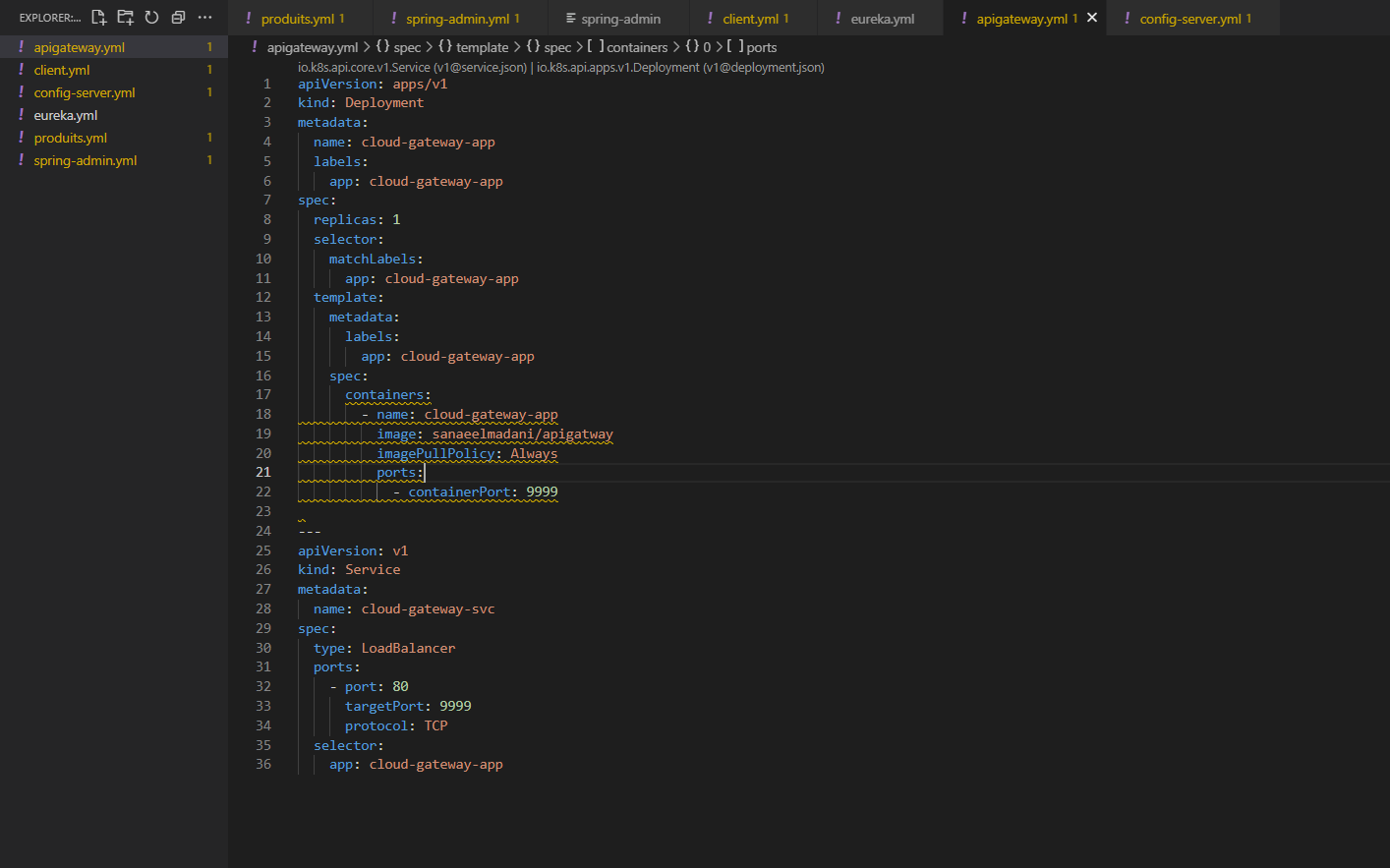


3-Après nous avons tapé la commande suivante mvn clean package dockerfile :push

pour transférer l’image construite vers Docker Hub:

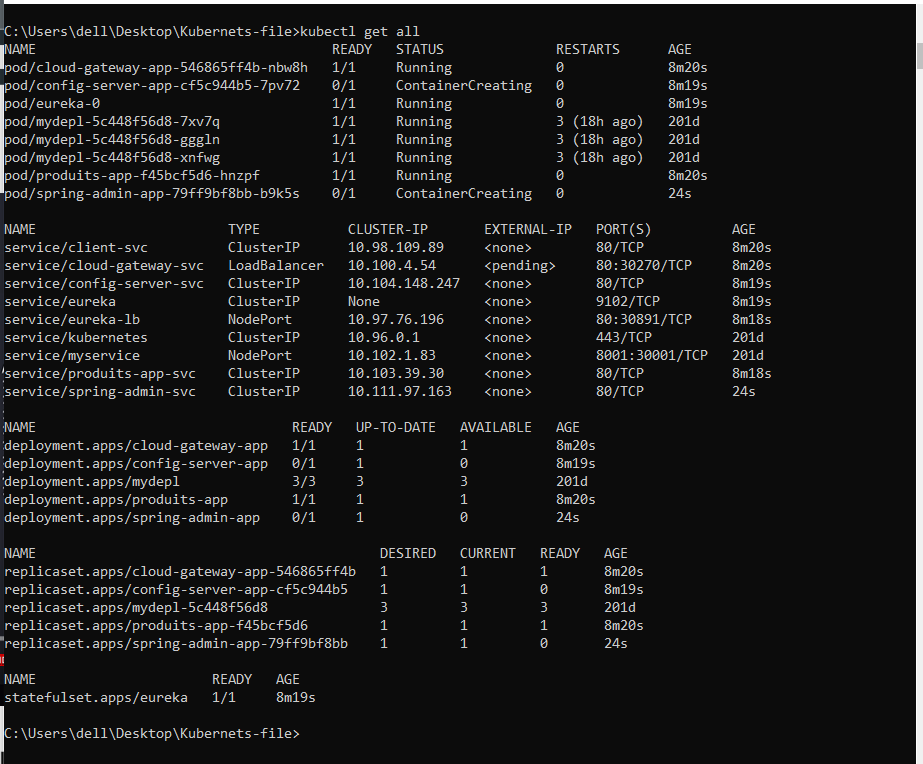


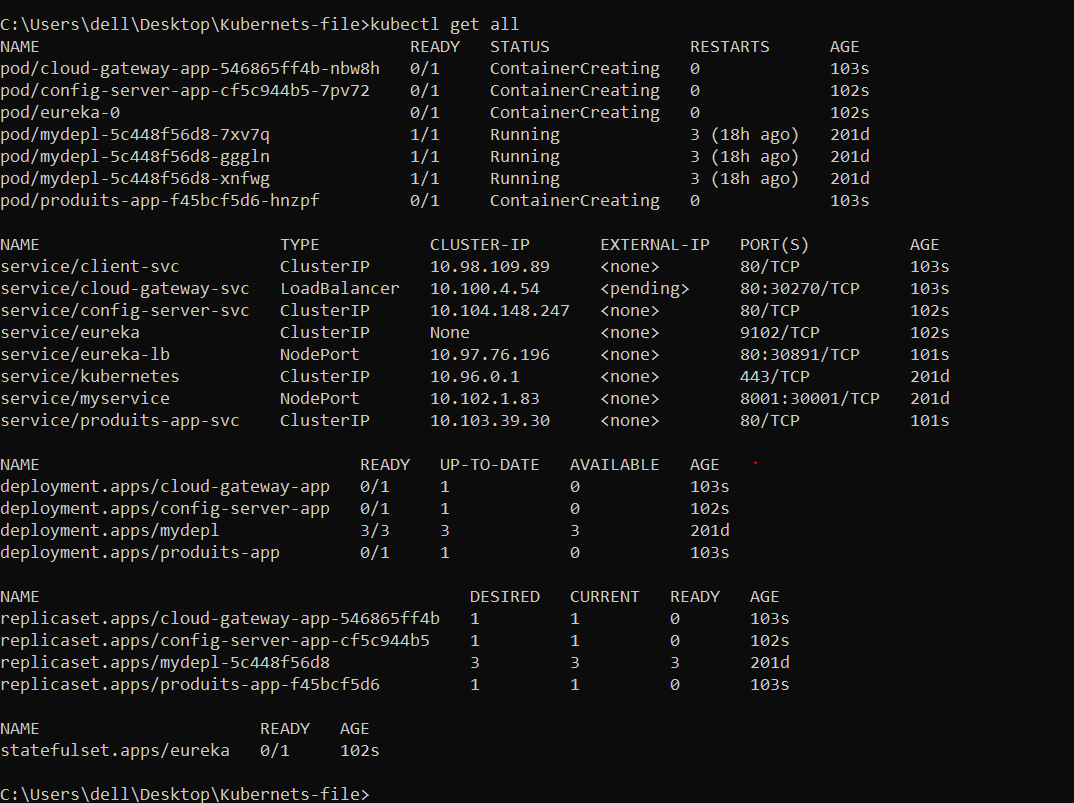
4-Après la construction des images,on doit maintenant créer les fichiers de configuration Kubernetes pour chaque microservices:

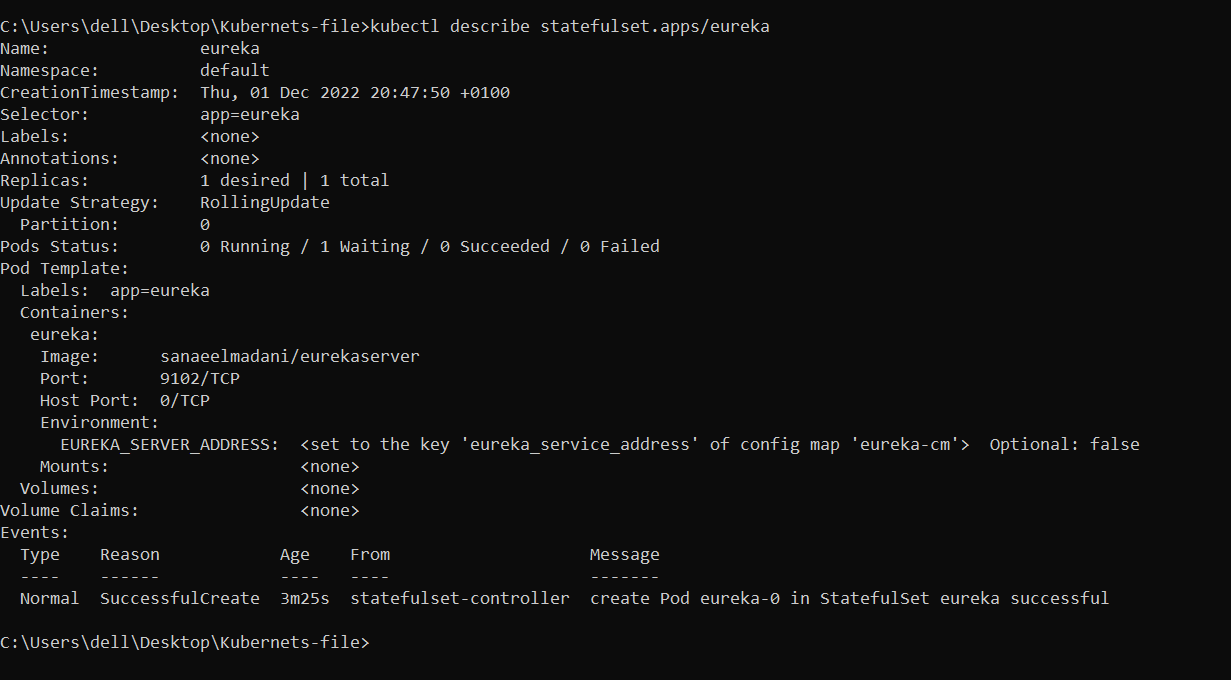


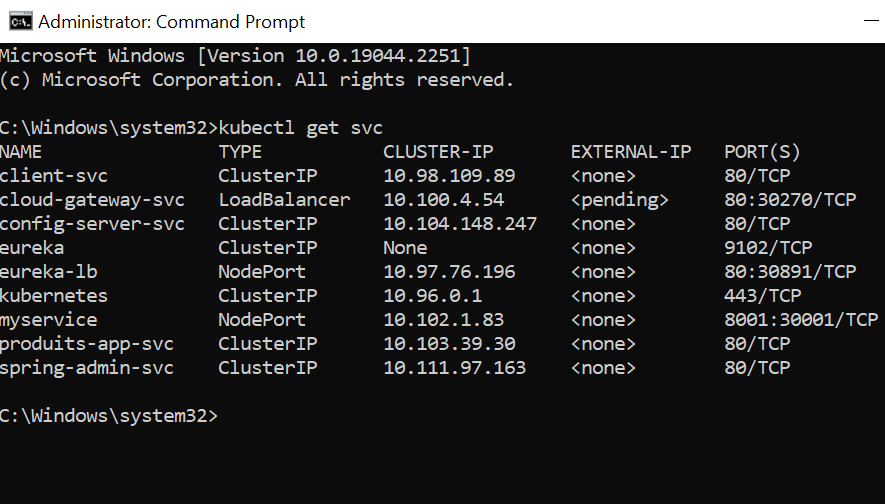
5-On exécute la commande suivante pour déployer les ressources telles que définies dans kubernetes.yaml:

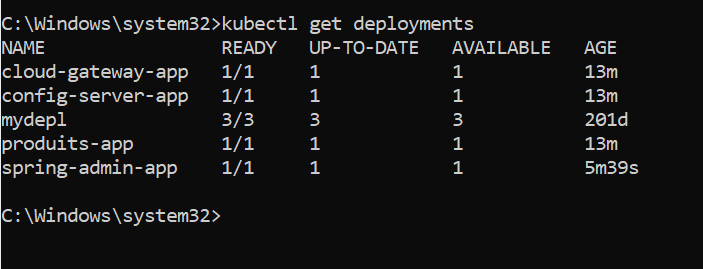
**kubectl apply -f kubernetes.yaml**

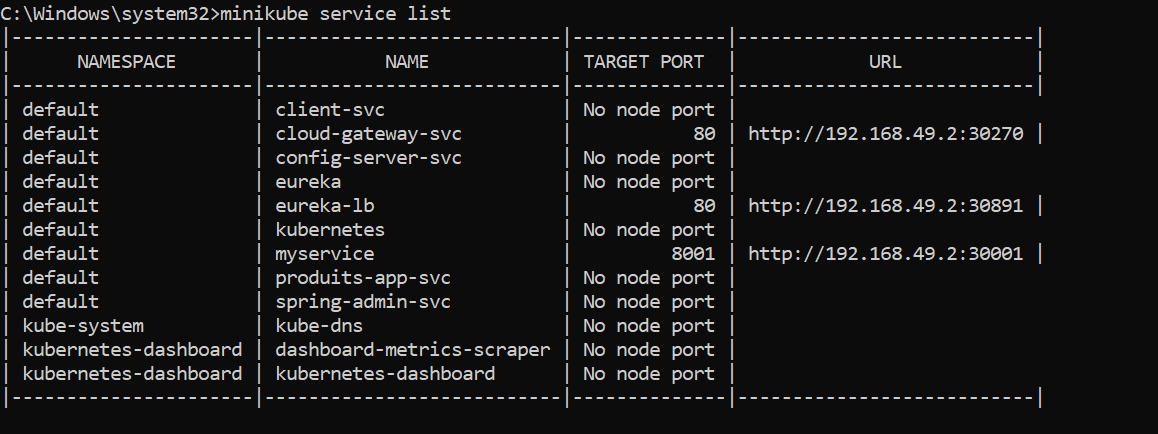




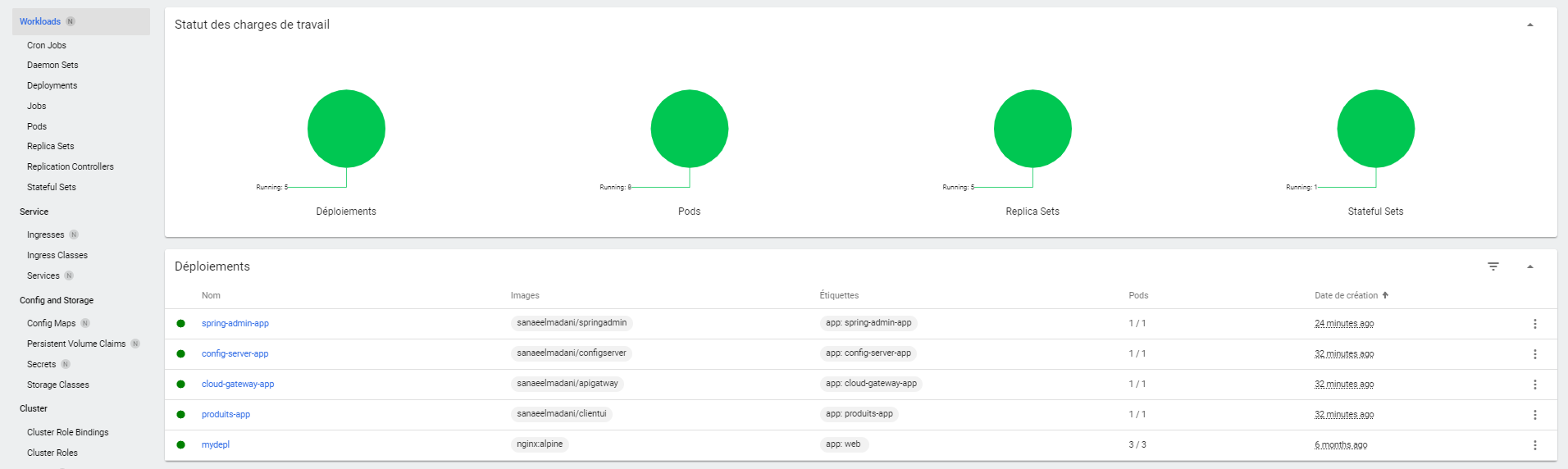


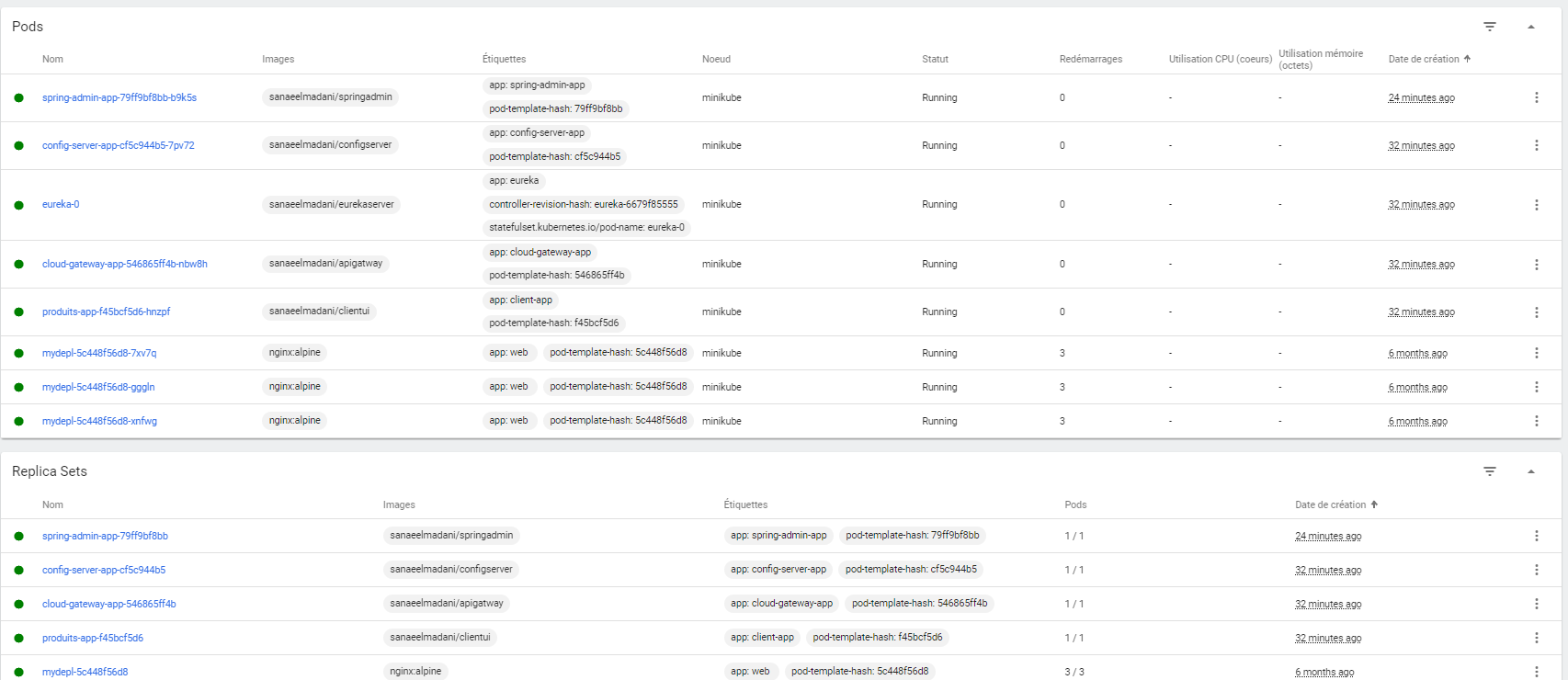






6-Kubernetes Dashboard





**⇒**Voilà nous avons réussi le déploiement des microservices en se servant de Kubernetes.

4. Utilisation d’un Service Mesh :

Un service Mesh est une couche de plate-forme au-dessus de la couche d'infrastructure qui permet une communication gérée, observable et sécurisée entre des services individuels .

Malheureusement on n’a pas réussi cette étape car il demande des ressources de grande capacité (RAM et Stockage).

Le lien github vers le projet après déploiement :

<https://github.com/sanae-elmadani/Microservices-Ecomerce-Project/tree/080b4919e3d828d15625bcd89c7caab64f05169d>

Le lien github vers les fichiers de configuration de Kubernets:

<https://github.com/sanae-elmadani/Files-configuration>