

Rapport CQRS microservices Elasticsearch-MongoDb-Kafka



INSAT
2021-2022

Travail réalisé par:



Ghassen Daoud



Mehdi Ben Chikha



Mouheb Ben Chiekh

Introduction

Dans ce projet, on va présenter un projet microservices, basé sur l'architecture microservice basée sur le modèle CQRS.

Le modèle CQRS (Command and Query Responsibility Segregation) sépare les opérations de lecture et d'écriture pour un data store. Les lectures et les écritures peuvent emprunter des chemins totalement différents dans l'application et peuvent être appliquées à différents data stores.

Notre travail consiste en 6 projets en total:

1. Un microservice pour écriture. (MongoDb)
2. Un microservice pour lire. (Elasticsearch)
3. Un microservice pour le service regisrty.
4. Un API Gateway.
5. Un projet frontend (Angular)
6. Un projet de test. (JMeter)

Principe général

MongoDb est une base de données mieux dédiée pour l'écriture. Et Elasticsearch est un logiciel pour l'indexation et la recherche de données. Il fournit un moteur de recherche distribué et multi-entité à travers une interface REST.

La communication entre les deux microservices (Mongo et Elasticsearch) se fait à travers Kafka qui est un projet à code source ouvert d'agent de messages développé par l'Apache Software Foundation et écrit en Scala. Le projet vise à fournir un système unifié, en temps réel à latence faible pour la manipulation de flux de données.

Le projet est basé sur l'architecture REST.

REST est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Les services web conformes au style d'architecture REST, aussi appelés services web RESTful, établissent une interopérabilité entre les ordinateurs sur Internet.

Présentation de l'architecture:

Pour bénéficier des avantages des technologies en lecture (Elasticsearch) et en écriture (MongoDB) aussi bien des microservices (en scalabilité, resilience, ...), on a choisi une architecture microservice RESTFUL suivant le modèle CQRS. Dans ce projet on a suivi l'approche DDD pour définir les bounded context de chaque microservice.

Description du métier

Le projet est une plateforme où l'utilisateur peut ajouter/lister la liste des produits dans un site E-commerce.

Les produits sont suivant le modèle suivant:

```
public class Product{
    @Id
    private String id;

    private String name;

    private double price;

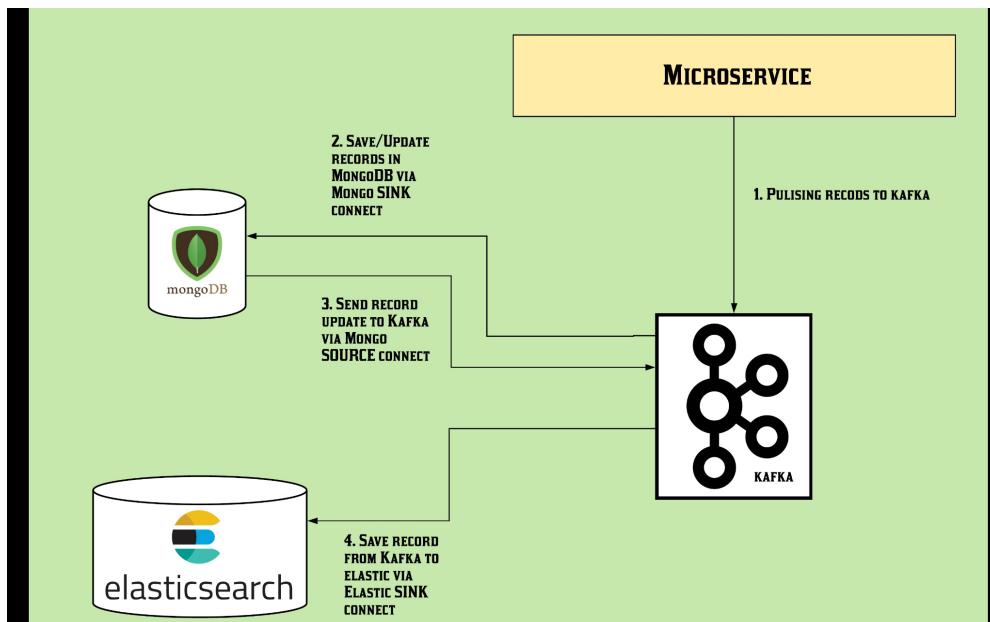
    private String category;

    private String description;

    private String image;
```

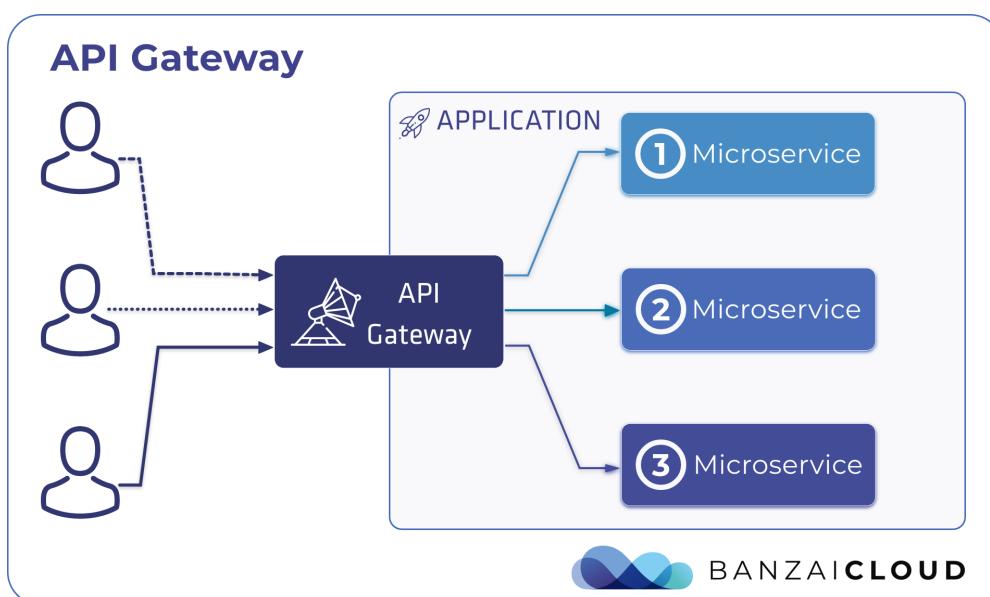
Mise en Oeuvre

On veut profiter des avantages de MongoDB dans le stockage de données et des avantages de ElasticSearch dans la recherche de données.

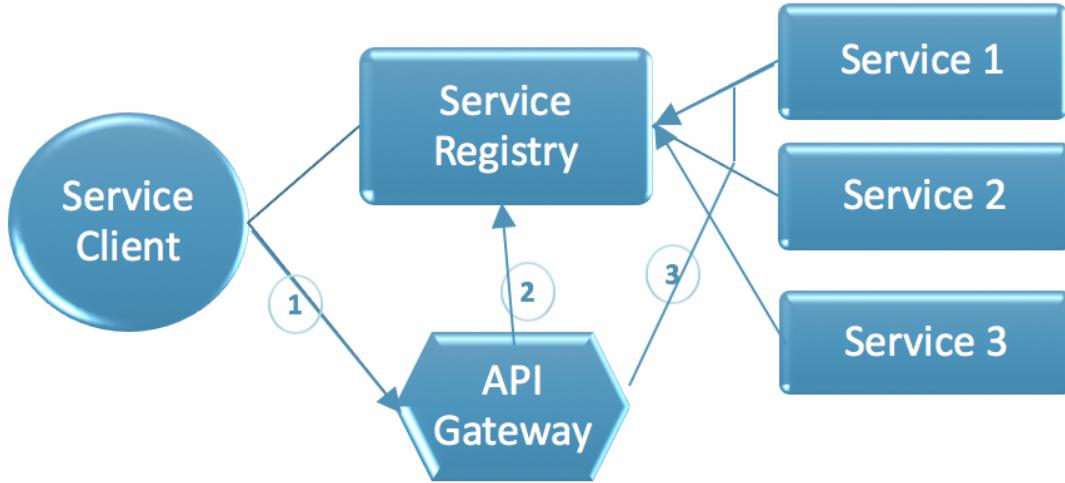


Les deux microservices sont registrés dans un registry service.

La communication entre le client (le end-user) et les microservices se fait à travers un API gateway.

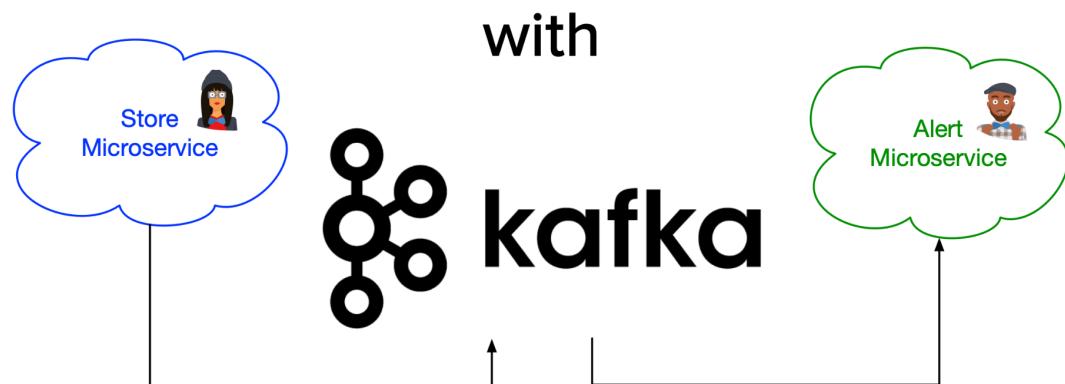


On a implémenté la registration des différents microservices dans un projet séparé service registry



On a fait la communication entre les différents microservices à travers kafka.

Communicate Between Microservices



Et enfin, on a fait le frontend à travers Angular. En utilisant la librairie Nebular.

Technologies utilisées



elasticsearch



{ REST }

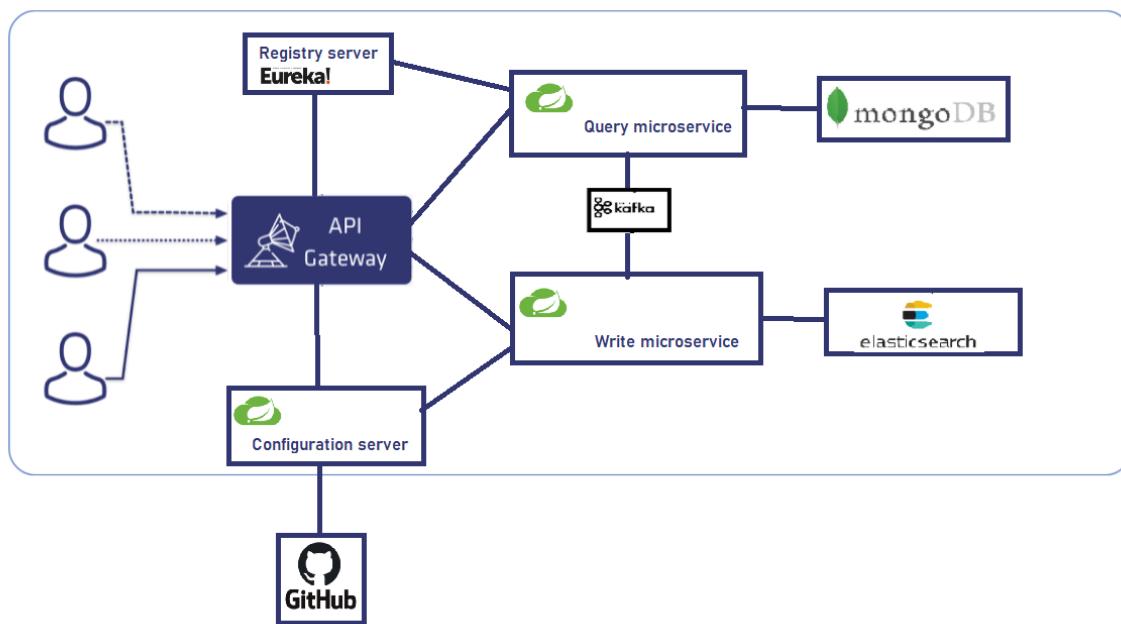


Eureka!

Implémentation

On a créé deux microservices pour la gestion des produits:

- un microservice: Query Products
- un microservice: Write Products
- On a utilisé Kafka pour synchroniser les deux microservice (communication asynchrone) afin d'assurer une éventuelle consistance
- Gateway API: un projet spring cloud qui redirige les requêtes
- Configuration server: qui se connecte à un repository github pour la configuration communes entres les différent microservices
- Registry server: we used eureka framework (projet Netflix) to create the server



Le code source du projet se trouve dans le lien GitHub suivant:

<https://github.com/Insat-Musketeers>

Service Registry

Le service Registry est implémenté à travers Eureka.

The screenshot shows the Eureka UI interface at localhost:8761. At the top, there are configuration settings for lease expiration:

Lease expiration enabled	True
Renews threshold	8
Renews (last min)	16

Below this is the "DS Replicas" section, which lists the instances currently registered with Eureka:

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - DESKTOP-K08VFFN-API-GATEWAY:9191
CONFIGURATION-SERVER	n/a (1)	(1)	UP (1) - DESKTOP-K08VFFN-CONFIGURATION-SERVER:9292
READ-PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-K08VFFN-READ-PRODUCT-SERVICE:8001
WRITE-PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-K08VFFN-WRITE-PRODUCT-SERVICE:8002

Under "General Info", there is a table of system metrics:

Name	Value
total-avail-memory	388mb
num-of-cpus	12
current-memory-usage	85mb (21%)
server-upptime	01:18

The taskbar at the bottom shows various pinned applications and system status indicators.

Frontend

Activities Google Chrome ▾ Jan 31 23:46

localhost:4200/products

E-Product

ADD A PRODUCT

Clavier HP

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Rem libero saepe voluptatem, praesentium unde eveniet motesla dolorem provident dolore quidem commodi repudianae molitia sunt suscipit totam perspicias, alias esse repellendum?

Price: 980dt

PC HP

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Rem libero saepe voluptatem, praesentium unde eveniet motesla dolorem provident dolore quidem commodi repudianae molitia sunt suscipit totam perspicias, alias esse repellendum?

Price: 1890dt

Souris Gamer

Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Rem libero saepe voluptatem, praesentium unde eveniet motesla dolorem provident dolore quidem commodi repudianae molitia sunt suscipit totam perspicias, alias esse repellendum?

Price: 50dt

PC Bureau DELL

Description:

Price: 2500dt

Activities Google Chrome ▾ Jan 31 23:24

localhost:4200/addProduct

E-Product

ADD A PRODUCT

Name:

Description:

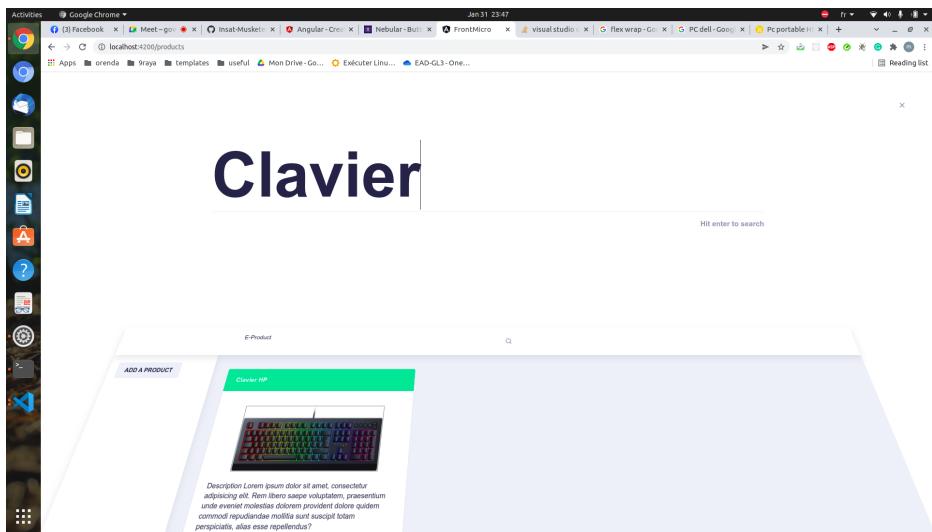
Price:

Category:

Choose File: No file chosen

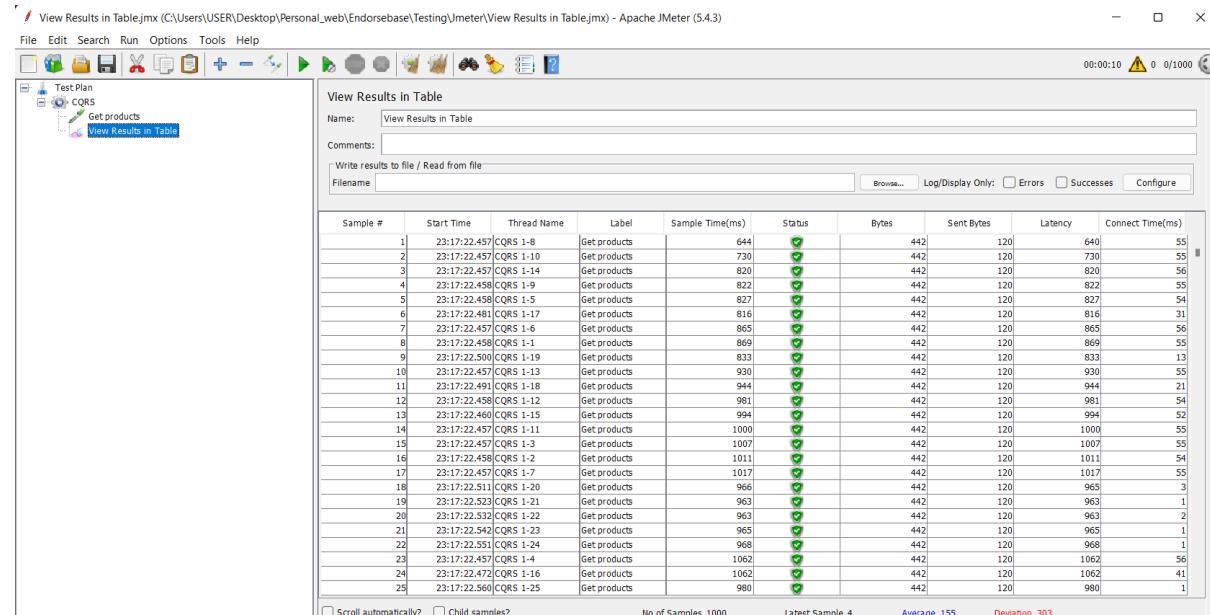
SUBMIT CANCEL

The image shows two screenshots of a web application titled 'E-Product'. The top screenshot displays a product listing page with four items: 'Clavier HP', 'PC HP', 'Souris Gamer', and 'PC Bureau DELL'. Each item has a thumbnail, a brief description, and a price. The bottom screenshot shows an 'Add Product' form with fields for Name, Description, Price, Category, and a file upload input. Both screenshots are taken from a Linux desktop environment with a dark theme.



Testing

On a implémenté un test de performance (stress test) en utilisant JMeter.
En simulant 1000 utilisateurs simultanés.



Le résultat du test est à 100% succès.

Conclusion

Les microservices sont un style d'architecture utilisé par de nombreuses organisations pour le développement de logiciels. Par le passé, l'industrie informatique utilisait des solutions monolithiques ou basées sur l'architecture

orientée services (ou SOA pour Service-Oriented Architecture) comme standard.

Cependant, le manque d'évolutivité dynamique de ce genre de système architectural n'était plus adapté à la complexité croissante des infrastructures actuelles. C'est là qu'intervient le microservice qui est finalement une évolution logique du système SOA conçu pour atteindre un degré élevé d'agilité, de distribution rapide et d'évolutivité.