Rapport Microservices

Reda Chafik et Al Mastour Salwa

1 Introduction

1.1 Aperçu du Projet

Ce projet vise à développer une application de services météorologiques en utilisant une architecture de microservices. Chaque service fonctionne indépendamment, fournissant des fonctionnalités spécifiques telles que les prévisions météorologiques, la gestion des utilisateurs et les notifications.

1.2 Importance de l'Architecture Microservices

Cette architecture offre une grande flexibilité, permettant une maintenance et des mises à jour plus aisées. Elle favorise également une meilleure scalabilité et une répartition efficace des ressources.

2 Architecture Microservices

2.1 Architecture

Cette architecture offre une grande flexibilité, permettant une maintenance et des mises à jour plus aisées. Elle favorise également une meilleure scalabilité et une répartition efficace des ressources.

2.2 Description des services

- Service Météo (meteo-service) : Responsable de fournir des informations météorologiques actualisées. Il récupère et traite les données météorologiques de diverses sources pour les présenter aux utilisateurs.
- Service Utilisateur (user-service) : Gère les informations des utilisateurs, y compris l'inscription, la gestion des profils, et l'authentification. Il stocke et récupère les données utilisateur de manière sécurisée.
- Service de Notification (notification-service) : Envoie des alertes et des notifications aux utilisateurs. Il peut s'agir d'alertes météo, de mises à jour importantes, ou de messages personnalisés.

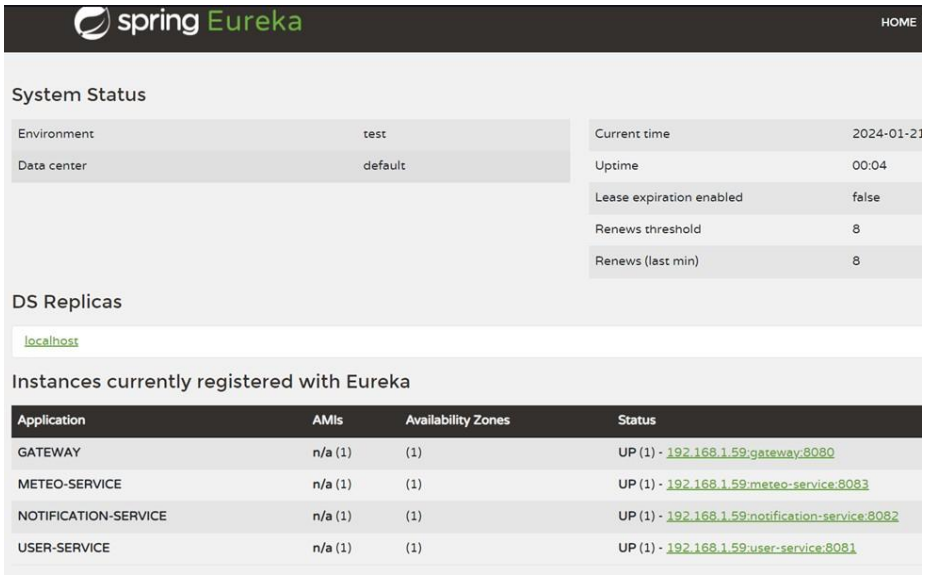

FIGURE 1 – Eureka Dashboard

— Service Eureka Server (eurekaserver) : Agit comme un serveur de découverte pour les microservices. Il permet aux services de s'enregistrer et de découvrir les autres services de l'architecture, facilitant ainsi la communication entre eux.
— Service Gateway (gateway) : Fait office de point d'entrée pour les requêtes externes. Il route les demandes aux services appropriés et gère les aspects liés à la sécurité, comme l'authentification et l'autorisation.

2.3 Mécanismes de communication

Les services communiquent principalement via des API REST, garantissant une interaction efficace et standardisée entre eux.

3 Conception des Microservices

Chaque microservice est conçu pour être autonome, avec sa propre base de données et son propre environnement d'exécution. Cette approche renforce la modularité et la résilience du système.

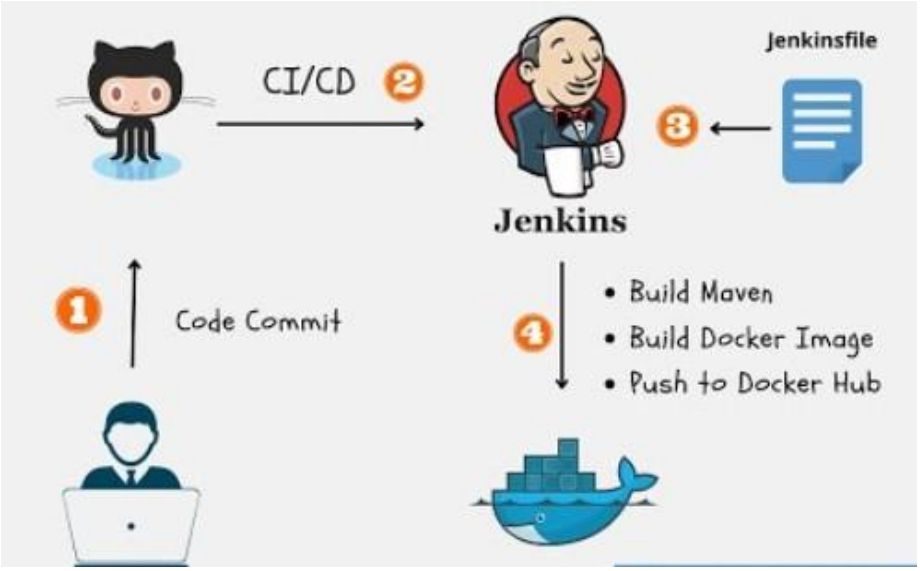

FIGURE 2 – Architecture

4 Conteneurisation avec Docker

4.1 Implémentation

Docker est utilisé pour empaqueter chaque microservice dans un conteneur distinct. Ces conteneurs peuvent être déployés et exécutés de manière isolée, garantissant la cohérence entre les environnements de développement et de production.

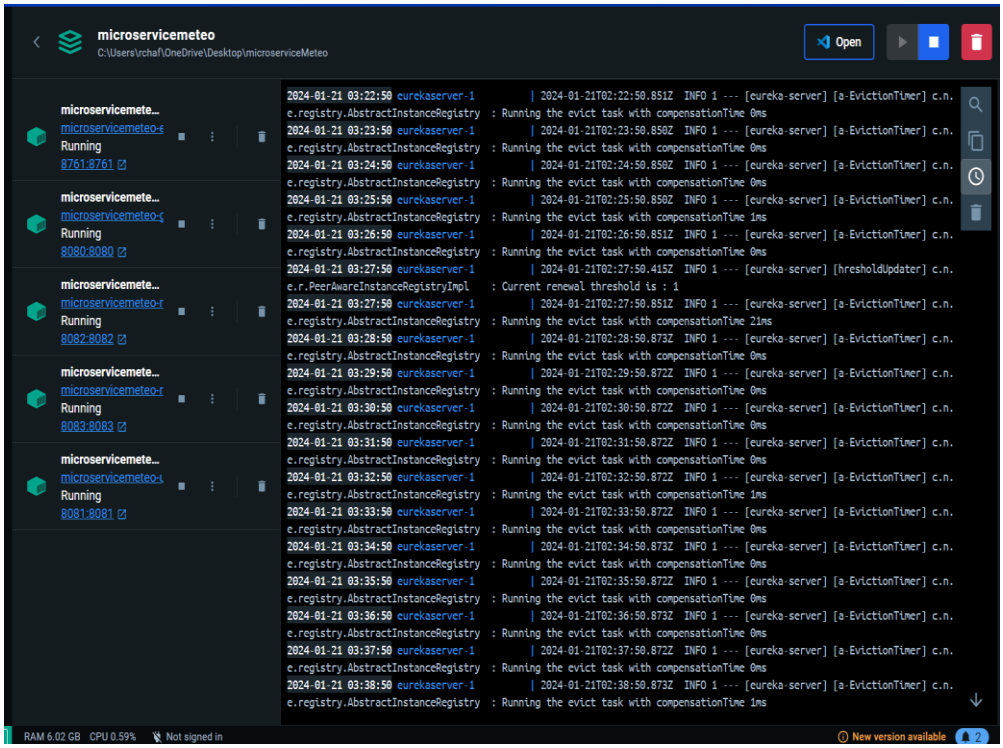

Figure 3 – Conteneurs Docker

4.2 Avantages

La conteneurisation facilite le déploiement, la mise à l'échelle et la gestion des microservices, tout en assurant un niveau élevé d'isolation et de sécurité.

5 CI/CD avec Jenkins

5.1 Processus

Jenkins orchestre le processus d'intégration et de déploiement continus (CI/CD), automatisant la construction des images Docker et leur déploiement.

5.2 Configuration

Un Jenkinsfile est utilisé pour définir les étapes du pipeline, y compris la construction, le test et le déploiement des microservices.

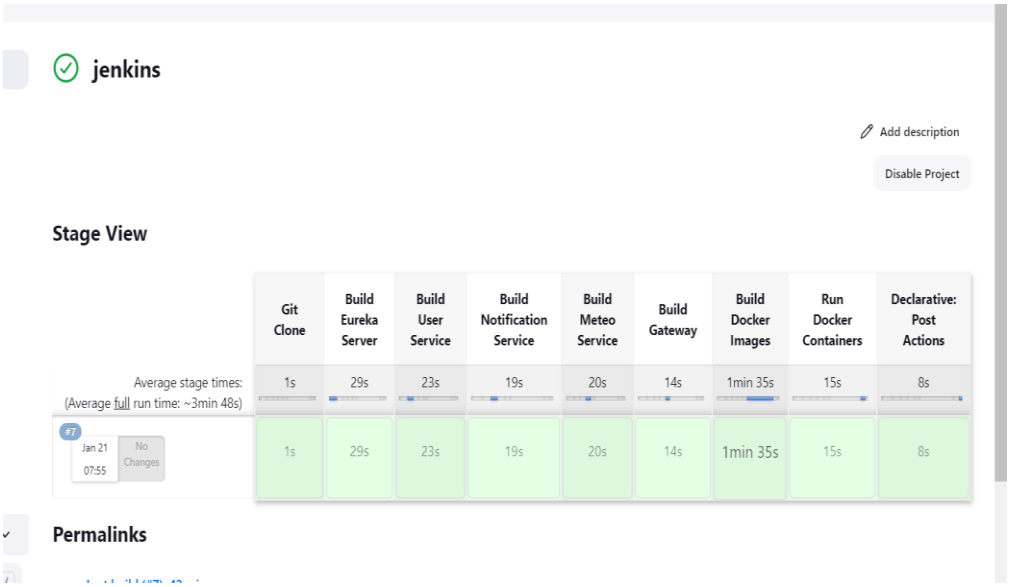

Figure 4 – Stage View Jenkins

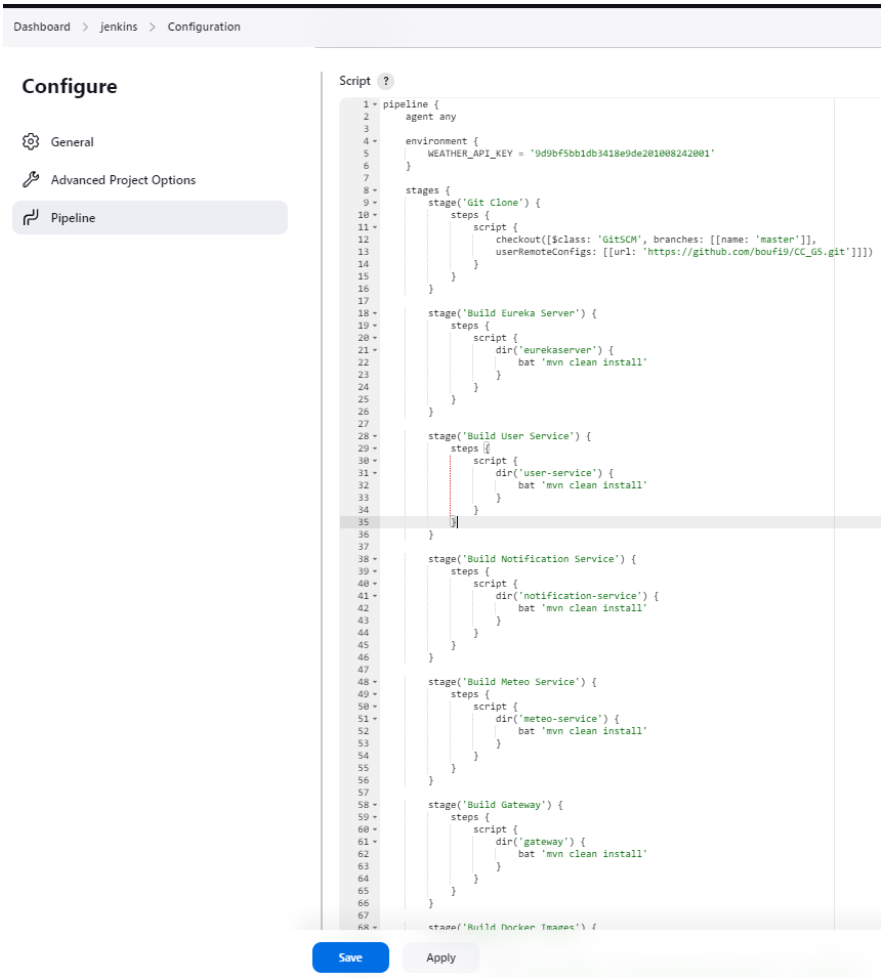

Figure 5 - Configuration pipeline

6 Déploiement Automatique

Ngrok permet d'exposer localement les services pour des tests et des démonstrations, en les rendant accessibles via Internet.

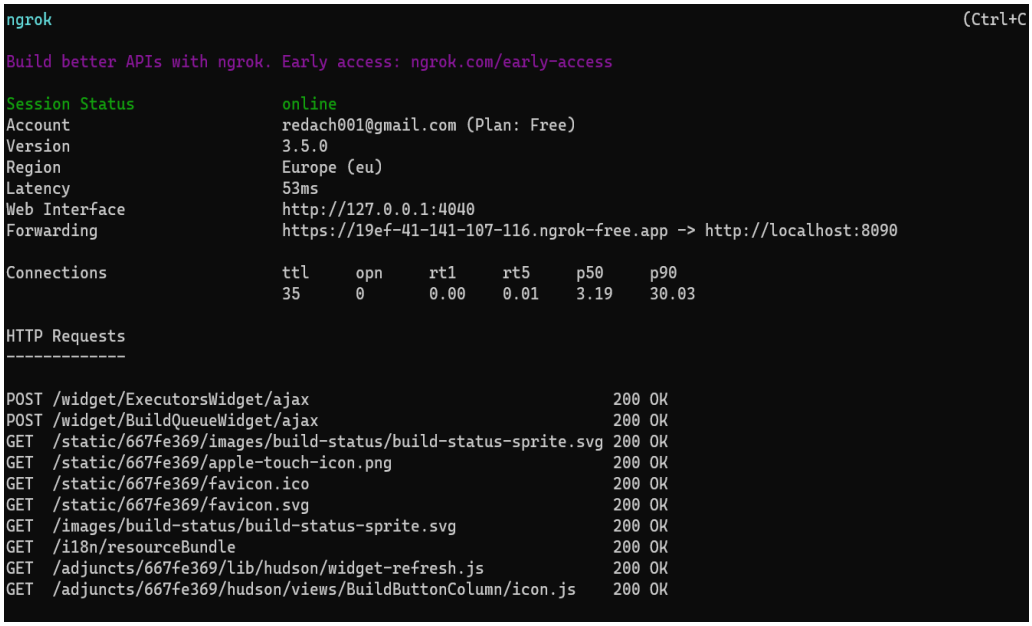

Figure 6 – Connexion ngrok

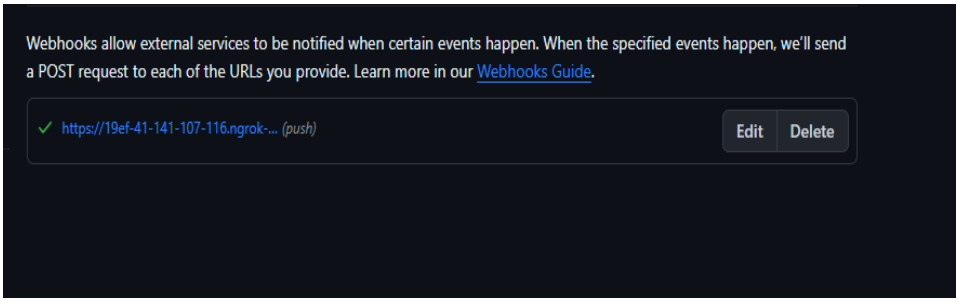

Figure 7 – Connexion avec Webhook

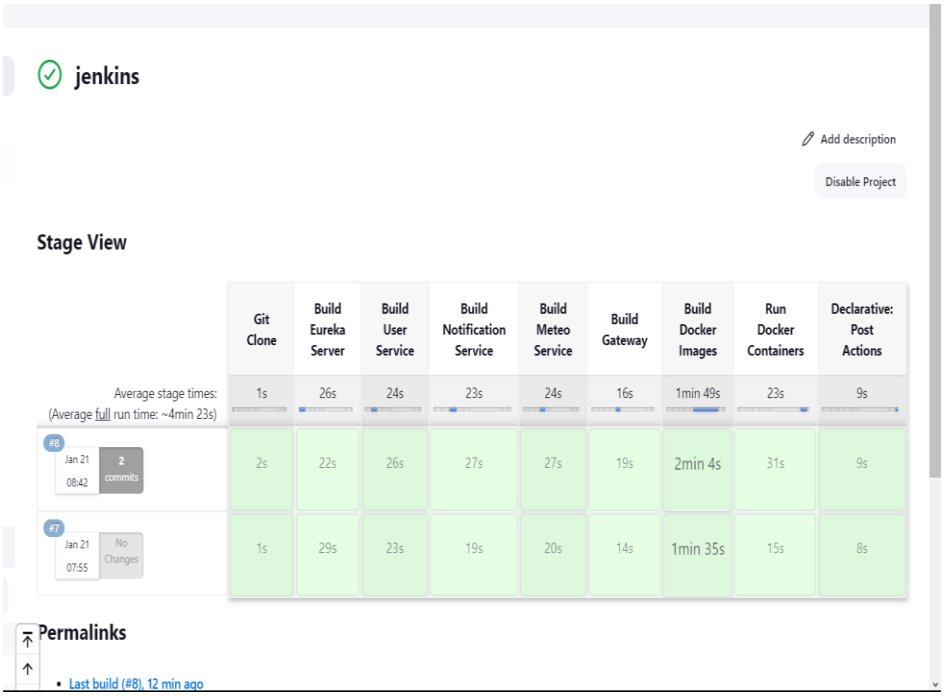

Figure 8 – Stage view avec automatisation

7 Conclusion

Le projet a réussi à mettre en place une architecture robuste et flexible, capable de répondre aux exigences modernes du développement logiciel.

Perspectives Futures : Exploration de nouvelles fonctionnalités, améliorations de l'infrastructure existante et intégration de technologies émergentes pour renfor- cer l'application.