Protocole d’expérimentation - FDI-LEEX

Période d'expérimentation :
11/07/2025 au 18/07/2025

Membres du groupe :
- Elyesse BEN HADJ
- Aurélien GUENEBEM KOSSI
- Beviryon ISSANGA NGOULOU
- Luca BRUNET
- Mattéo TOLLA
- Sandro BAKURADZE

Contexte

Ce projet s’inscrit dans la communauté Cloud & Microservices. Nous disposons d’une application en architecture microservices composer de trois services :
- user-service (authentication)
- file-service (partage de fichiers)
- messaging-service (système de messagerie).

L’objectif de cette expérimentation est de :
- Comparer différentes technologies d’API Gateway en les intégrant à notre architecture
- Comparer différents orchestrateurs de conteneurs (Docker Compose, Kubernetes, Docker Swarm)

Cette expérimentation permettra une analyse comparative à la fois technique et liée à l’expérience développeur, en environnement local uniquement.

Objectif du protocole

Déployer et comparer plusieurs API Gateways dans une architecture microservices, puis observer leur intégration au sein de différents orchestrateurs conteneurisés.

Chaque binôme du groupe prendra en charge une technologie d’API Gateway à tester et intégrera cette gateway aux trois services existants. Les orchestrateurs ciblés seront Docker Compose, Kubernetes (via Minikube) et Docker Swarm.

Architecture cible

- Trois services : user-service, file-service, messages-service (backend uniquement)
 - 1 base de données MySQL pour chaque service.
 - API Gateway pour exposer les services via une seule entrée
 - Déploiement local via Compose, Kubernetes et Swarm

API Gateways envisagées

 - Traefik
 - NGINX
 - Envoy Proxy (via Kubernetes)
 - Ocelot (via .NET)
 - Kong (si temps disponible)

Orchestrateurs à comparer

- Docker Compose
- Kubernetes (Minikube)
- Docker Swarm

Outils

- Docker / Docker Compose
- Kubernetes (Minikube, kubectl)
- Docker Swarm
- API Gateways : Traefik, NGINX, etc.
- Postman, curl, k6 pour les tests
- Git pour le versionnement

Déroulé de l’expérimentation

1. Définir les routes, schéma et accès de chaque service
2. Configurer chaque API Gateway et l’intégrer dans l’architecture existante
3. Dockeriser la configuration complète
4. Déployer avec Compose, Kubernetes, puis Swarm
5. Réaliser des tests (fonctionnels, perf, intégration)
6. Remplir la grille de comparaison
7. Consolidation des résultats pour rendu final

Critères de comparaison - API Gateway

Critère	Éléments observés
Facilité d’intégration	Temps de mise en place, documentation
Performance	Temps de réponse, latence
Simplicité de configuration	Clarté et taille des fichiers de conf
Logs et monitoring	Disponibilité et lisibilité des logs
Expérience développeur	Courbe d’apprentissage, ergonomie

Critères de comparaison - Orchestrateurs

Critère	Éléments observés
Facilité de configuration	Complexité des fichiers YAML
Temps de déploiement	Durée nécessaire à l'exécution
Scalabilité	Ajout de réplicas, montée en charge
Résilience	Capacité à relancer un service défaillant
Observabilité	Logs, surveillance, statut des services
Expérience développeur	Prise en main, ergonomie

Perspectives

- Ajout d’un monitoring (Prometheus, Grafana)
- Pipeline CI/CD (GitHub Actions)
- Test de montée en charge
- Intégration d’un frontend minimal (bonus)
- Déploiement sur un cloud public (optionnel, hors périmètre principal)

Groupe	API Gateway	Orchestrateur
A	**Spring Cloud Gateway**	**Docker Compose**
B	**Traefik**	**Docker Swarm**
C	**Kong Gateway OSS**	**Kubernetes (Kind)**

A = Luca
B = Aurélien
C = Sandro