Module SAE 503

Dossier d'Architecture Technique

Orchestrer la conteneurisation
d'une application

Elaboré par

Daniel Aronn

Sérandour Clément

Promotion

FA3A

Année

Universitaire 2024-2025

Lot 1 : Documentation Architecture Technique **3**
1. Introduction 3
2. Objectifs 3
3. Architecture 3
3.1. Composants principaux 3
3.2. Schéma 4
3.3. Gestion des accès 4
4. Sécurité et Observabilité 4
5. Conclusion 4

Lot 1 : Documentation Architecture Technique

1. Introduction

Ce projet consiste à déployer une application web gérant des citations du capitaine Haddock, divisée en microservices, sur une plateforme cloud native avec orchestration de conteneurs. Le but est d'assurer la flexibilité, la scalabilité et l'observabilité de l'application.

2. Objectifs

- **Conteneurisation** : Déployer chaque API dans un conteneur dédié.
- **Orchestration** : Utilisation de Kubernetes pour gérer les conteneurs, assurer la montée en charge et l'isolement des environnements (preprode, production).
- **Scalabilité et haute disponibilité** : Utilisation de Traefik comme reverse proxy et mise à l'échelle automatique.
- **Observabilité** : Suivi des performances et alertes en cas de problèmes.

3. Architecture

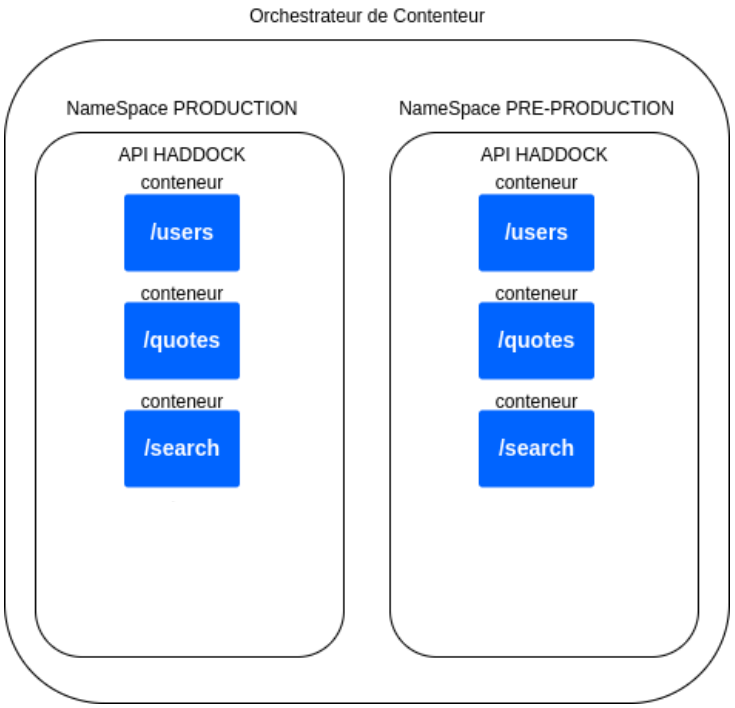
3.1. Composants principaux

- **Microservices** : Trois microservices (utilisateurs, citations, recherche) gérés par Kubernetes.
- **Base de données** : Stockage persistant des données.
- **Reverse Proxy** : Traefik pour acheminer les requêtes HTTP.
- **Orchestrateur de conteneurs** : Kubernetes pour gérer les déploiements et la montée en charge.

On utilisera un environnement de **PROD** qui sera utilisé par les utilisateurs finaux, avec des données réelles.

Ainsi qu’un environnement de **PRÉPROD** qui sera l’ environnement de test qui simule l’environnement de **PROD**, utilisé pour valider les mises à jour avant de les déployer en production.

3.2. Schéma



3.3. Gestion des accès

- **Jeton** ADMIN_KEY : Requis pour certaines opérations (ajouter, supprimer, modifier).
- **Limitation de requêtes** : Maximum 10 requêtes par minute.

4. Justification technique

Grafana : Offrant une visualisation avancée des métriques, il permet de surveiller la performance des services en temps réel.

Kubernetes : Permet de déployer, gérer et mettre à l'échelle des applications conteneurisées de manière efficace.

Traefik : Fournit un reverse proxy dynamique adapté aux architectures cloud natives avec gestion des certificats SSL automatisée.

5. Sécurité et Observabilité

- **Traefik + SSL** : Sécurisation des connexions.
- **Alertes** : Notification en cas de problème de performance ou d'erreurs excessives.
- **Tableau de bord** : Suivi des métriques via grafana et prometheus (temps de réponse, CPU, mémoire, etc.).

6. Conclusion

Cette architecture garantit une gestion optimisée des ressources, une évolutivité facile et une bonne observabilité, tout en répondant aux besoins du projet.