SAE 5.03 : Orchestrer la conteneurisation d'une application

1. Contexte

Votre équipe est chargée de déployer une application web permettant notamment d'afficher des citations du capitaine Haddock, personnage de la série de bande dessinée « les aventures de Tintin » d'Hergé. Cette application propose plusieurs API permettant d'afficher un juron du capitaine Haddock, ou encore en ajouter/supprimer/modifier.

Nous souhaitons rendre accessible ce développement interne au plus grand nombre en utilisant un hébergement « cloud native » basé sur l'orchestration de conteneurs, afin d'apporter plus de flexibilité et de portabilité.

2. Principes généraux

Afin de pouvoir maîtriser les coûts d'hébergement, nous avons fait le choix de nous tourner vers la conteneurisation de l'application afin de pouvoir disposer de plusieurs plateformes (qualification, production) permettant de tester les changements dans des environnements différents tout en conservant le même comportement de l'application. Votre mission sera axée sur les éléments suivants :

- Création d'un conteneur,
- Déploiement des conteneurs pour rendre l'application fonctionnelle (1 conteneur par point d'accès API),
- Mise en place d'un orchestrateur permettant la montée en charge des services par le lancement de multiples instances du même service

Vous déploierez une plate-forme de qualification et une plate-forme de production.

3. Application des citations du capitaine Haddock

Cette application présente les points d'accès :

- /users : gestion des utilisateurs
  - GET : affiche la liste des utilisateurs
  - POST : ajoute un utilisateur
- /quotes :
  - GET : affiche toutes les citations
  - POST : ajoute une citation
  - DELETE : supprime une citation
- /search :
  - GET : effectue une recherche de citation par mot-clé
- /apidocs : affiche le swagger.

O. Guillossou olivier@guillossou.fr

Certaines opérations nécessitent de fournir un jeton d'authentification dans l'en-tête « Authentification » (paramètre « ADMIN_KEY »).

Vous vous enregistrerez sur Github Classrooms https://bit.ly/iutsm-sae503 .

Vos livrables (code et documents) seront placés dans ce dépôt Github Classrooms. La date limite est positionnée au 02 février 2026 à 12h00, et vous taguerez « sae503 » les fichiers et documents définitifs. L'évaluation de cette SAE se fera sous la forme d'une soutenance.

4. Fonctionnalités attendues

a. Multiples plateformes

- L'application devra pouvoir être instanciée à plusieurs reprises de manière cloisonnée dans le même cluster,
- Chaque plateforme sera nommée selon son type (qualification, production),
- Chaque plateforme exécutera plusieurs images du conteneur applicatif (un par microservice), et les requêtes entrantes seront acheminées par l'intermédiaire d'un *reverse proxy*.
- Chaque plateforme doit être configurée de telle sorte qu'une plateforme « hors production » ne puisse pas accaparer les ressources allouées à la plateforme de production.

b. Stockage

- Vous veillerez à ce que les informations contenues dans la base de données soient bien stockées dans un espace où la persistance des données est assurée.

c. Accès

- La gestion des accès http au cluster seront faits à l'aide de Traefik.
- Vous configurerez un limiteur de requêtes à 10 requêtes par minute.

d. Divers

- Pour accéder au service, utilisez le service https://nip.io pour créer le FQDN propre à chaque plate-forme
- Le déploiement se fait sur une VM unique. Les exigences « cluster » (stockage, montée en charge) sont évaluées sur la mise en œuvre des mécanismes Kubernetes et la capacité à expliciter les limites d'un mono-nœud, ainsi que les choix à faire en environnement multi-nœuds.

O. Guillossou olivier@guillossou.fr

5. Allotissement

Afin de faciliter la décomposition du projet en différentes tâches, l'équipe en charge de la réalisation sera attentive à réaliser un lotissement de son activité, en fournissant pour chaque lot les livrables associés :

- Lot 1 : Définition de l'architecture : vous rédigerez un document d'architecture technique qui définit et documente tout ce qu'il faut faire et mettre en place pour réussir la mise en œuvre de l'architecture, en vue d'atteindre les objectifs et respecter les différentes contraintes.

Livrable : un dossier d'architecture technique.

Evaluation : clarté et complétude du DAT, justification des choix techniques, diagrammes…

- Lot 2 : Mise en place et configuration d'un orchestrateur de conteneurs : vous mettrez en place une solution d'orchestration de conteneurs sur un ensemble de machines virtuelles.

Livrable : un cahier d'installation technique (détaillant l'architecture et la configuration mises en place sur plusieurs axes : matériel, logiciel, schéma de principe, adressage réseau, matrice de flux, etc.).

Evaluation : clarté et complétude du CIT

- Lot 3 : Refactorisation de l'application : vous adapterez le code fourni pour créer 3 microservices selon l'activité : 1/ fonctions concernant les utilisateurs « /users » 2/ fonctions concernant les citations « /quotes », 3/ fonctions de recherche des citations. Vous respecterez les bonnes pratiques des *12-Factors* *https://12factor.net/fr/*

Livrable : un repository Git contenant le code + le Dockerfile de chaque service.

Evaluation : respect de la consigne, Dockerfile fonctionnel. Point bonus : modifier l'authentification pour prendre comme référence l'identifiant/mot de passe des utilisateurs en base de données.

- Lot 4 : Instanciation de l'application dans l'orchestrateur : vous déploierez et rendrez accessible l'application grâce à l'orchestrateur de conteneurs. Idéalement, créer un modèle de déploiement Helm.

Livrable : les manifestes de déploiement dans un repository Git

Evaluation : manifestes conformes, clarté du repository. Bonus pour utilisation de Helm.

- Lot 5 : Sécurité : vous démontrerez que vous avez respecté les exigences minimales de sécurisation de l'application. Toute utilisation d'information sensible doit être en secret Kubernetes. Les instances doivent être séparées de manière logique entre namespace. Les images Docker utilisées doivent être minimales ou distroless, et le SecurityContext ajusté pour empêcher un fonctionnement en *root* par exemple. L'utilisation des tags des images Docker doit être réalisé avec discernement. Vous utiliserez l'outil *Trivy* pour vous assurer de l'absence de vulnérabilités connues ou de mauvaise configuration.

Livrable : dans votre repository, un fichier SECURITY.md contenant la liste des exigences appliquées et la preuve pour chacune, et éventuellement un paragraphe indiquant les exigences que vous n'avez pas pu appliquer et/ou vos observations.

Evaluation : exigences respectées. Bonus pour l'utilisation de *Trivy* dans un opérateur.

O. Guillossou olivier@guillossou.fr