SAE 5.03 : Orchestrer la conteneurisation d'une application

1. Contexte

Notre société a mis au point une application permettant de gérer une file d'attente, constituée de trois points de terminaison (« prendre un numéro », « appeler un numéro », « afficher le nombre de personnes qui attendent »). Il est possible de gérer plusieurs files d'attente en exécutant plusieurs instances de l'application. Le stockage se fait dans une base de données Redis. Nous souhaitons désormais 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.

Nous faisons appel à votre équipe pour mettre en place une telle solution.

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 (développement, 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 file d'attente et par plateforme),
- Mise en place d'un orchestrateur permettant la montée en charge des services par le lancement de multiples instances du même service

Nous souhaitons que les files d'attentes suivantes soient créées sur la plate-forme de production :

- Equipe Rubis
- Equipe Saphir
- Equipe Emeraude
- Equipe Diamant

Les points d'accès de chaque file d'attente sera placé dans un chemin mettant en évidence le nom de l'équipe (par exemple : https://fqdn/rubis/list_tickets).

3. Application de gestion de file d'attente

L'application Python de gestion de la file d'attente nécessite la fourniture de trois arguments :

- Nom ou adresse du serveur Redis (par défaut : « 127.0.0.1 »)
- Port TCP utilisé pour se connecter au serveur Redis (par défaut : 6379)
- Nom de la file d'attente (par défaut : « DEFAUT »)

Cette application présente les points d'accès :

- /get_ticket : émet un nouveau ticket
- /call_ticket : appelle le prochain numéro. Détruit le ticket en mode FIFO.
- /list_tickets : liste le contenu de la file d'attente
- /docs : affiche le swagger.

Le code est disponible dans cet entrepôt Github : https://github.com/guilloss/iut-stmalo-sae503

4. Fonctionnalités attendues

a. Multiples plateformes

- L'application devra pouvoir être instanciée à plusieurs reprises de manière cloisonnée,
- Chaque plateforme sera nommée selon son type (développement, qualification, production),
- Chaque plateforme exécutera plusieurs images du conteneur applicatif (un par file d'attente), 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 relatives aux files d'attente soient bien stockées dans un espace où la persistance des données est assurée.

c. Observabilité

- La solution doit mettre à disposition un tableau de bord permettant de suivre les paramètres d'exécution de l'application et de la plateforme d'orchestration.
- Un tableau de bord synthétique devra reprendre les éléments suivants :
  - Temps de réponse de la base de données
  - Evolution du nombre de requêtes vers les microservices
  - Nombre de conteneurs en cours d'exécution
  - Mémoire et CPU utilisés
  - Taille de chaque file d'attente
- Vous devrez mettre en place un mécanisme d'alerte (par mail ou messagerie instantanée) en cas de survenance de certains événements ayant un impact sur la production, par exemple un taux de requêtes en erreur trop important.

5. Lotissement

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.

- Lot 2 : Mise en place d'une solution de stockage externe : cette solution doit permettre d'assurer la persistance des données de l'application.

~~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.).~~

- Lot 3 : 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.).

- Lot 4 : Instanciation de l'application dans l'orchestrateur : vous déploierez et rendrez accessible l'application grâce à l'orchestrateur de conteneurs.

Livrable : les scripts de déploiement

- Lot 5 : Mise en place d'une couche d'observabilité : vous déploierez et configurerez une solution permettant de suivre les performances de l'application déployée.

Livrable : un ou plusieurs tableaux de bord affichant *a minima* les informations demandées

- Lot 6 : sécurisation : vous sécuriserez l'application de telle sorte que l'appel aux API soit authentifié.

Livrable : le plan de remédiation