Master 1 GIL - Document technique - Environnement Docker d'Agora

Groupe 1 Agora V3-1 : Partie bibliothèque de jeux

12 mai 2024

Version	1
Date	12 mai 2024
Rédigé par	BINGINOT Etienne

Mises à jour du document

Version	Date	Modification réalisée
1	12 mai 2024	Création du document

Table des matières

1 Pourquoi Dockeriser Agora

Agora est une application complexe, nécessitant de nombreux outils annexes pour fonctionner. En effet, en plus de l'application Agora en elle-même, une base de données, un serveur mail, un serveur Mercure et un reverse-proxy. Ces composants, ayant chacun une configuration spécifique et des ports à utiliser, ne sont pas facilement synchronisables.

Ainsi, Docker permet de répondre à ce problème, en centralisant les configurations et en permettant de facilement synchroniser les différents serveurs que l'application utilise.

2 Les différents Dockerfile

Deux Dockerfile sont utilisés dans l'application : celui de l'application de base mais également un Dockerfile utilisé dans la CI/CD.

2.1 Dockerfile de l'application

Ce premier Dockerfile se base sur l'image Frankenphp. Cette image contient un serveur Caddy (serveur web) optimisé pour PHP. Ainsi, le compilateur PHP et le serveur Caddy sont réunis dans la même image et donc le même conteneur, permettant une communication entre les deux quasi instantanée. Ce serveur PHP utilise un système de Worker pour charger en mémoire la totalité de l'application dans différents processus, pour permettre par la suite de répondre de façon très rapide aux requêtes.

La suite du Dockerfile se base sur l'installation des dépendances, notamment Symfony puis des dépendances Symfony via l'utilisation de composer (gestionnaire de dépendances du projet). La fin du Dockerfile ouvre le port 7070, le port que le conteneur Agora utilise.

2.2 Dockerfile de la CI/CD

Ce Dockerfile est une version minimaliste du Dockerfile précédent. Celui-ci utilise une image PHP plus légère (php fpm), et ne va installer en dépendance que Symfony et Composer. Les autres dépendances seront installés lors du lancement de la CI/CD.

3 Les différents Docker-compose

De nombreux fichiers compose sont disponibles pour l'application. Il y a tout d'abord une base, compose.yaml, définissant tous les concepts globaux à tous les autres fichiers docker compose. Ainsi, tous les autres fichiers ne seront que des "précisions" de ce fichier de base.

3.1 compose.yaml

Ce fichier définit les 3 conteneurs de base :

— reverse-proxy : point d'entrée de l'application, son but est d'optimiser les accès à Agora en réalisant un système de cache. De plus, de nombreuses métriques sont récoltées pour analyser les données de l'application. Il configure également l'HTTPS

— agora : il s'agit du conteneur de base de Agora. Celui-ci se concentre donc sur la compilation du Dockerfile, puis du lancement de cette image avec les configurations choisies. La configuration Mercure est également présente ici, le serveur Mercure étant intégré dans l'image frankenphp.

— database : définit la base de données du projet, en utilisant mariaDB. Un volume est utilisé pour permettre de configurer la base de données de test. Un healthcheck est également utilisé pour assurer que le conteneur Agora ne se lance uniquement quand la base de données est initialisé

3.2 compose.override.yaml

Ce fichier sert à lancer l'application dans un environnement de développement de base. Ainsi, les bases de données de base et de test sont initialisés, tailwind est compilé en mode écoute (chaque modification CSS recompilera Tailwind) et le serveur php est lancé. De plus, un serveur mail est lancé.

3.3 compose.staging.yaml

Ce fichier a pour but de lancer l'application dans un environnement de test. Ainsi, un environnement de prod est simulé mais avec l'initialisation de la base de données de test et un serveur mail en plus. Les nombreux tests peuvent ainsi être exécutés ici sans impacter l'application en production.

3.4 compose.prod.yaml

Enfin, ce fichier compose permet de lancer l'application de production dans un environnement minimal. Ainsi, la base de données de test n'est pas initialisé et aucun serveur mail n'est lancé (un vrai serveur mail doit être utilisé).

3.5 Monitoring de l'application

A la place du fichier compose.yaml, deux autres fichiers sont disponibles : compose_monitor.yaml et compose_monitor_light.yaml.

3.5.1 compose_monitor_light.yaml

Ce fichier rajoute 3 conteneurs supplémentaires pour monitorer l'application :
— Jaeger : permet d'analyser les appels qui sont réalisés sur l'application, et les différentes routes qui sont appelées.
— Prometheus : analyse les temps de réponses de l'application, les différents code de retour HTTP, ...
— Grafana : permet de visualiser les données de Prometheus avec une interface graphique agréable

3.5.2 compose_monitor.yaml

Ce fichier rajoute en plus des 3 conteneurs précédents une stack ELK complète. Ainsi, les logs de l'application Agora mais également des autres conteneurs comme Traefik sont centralisés dans l'outil, permettant de visualiser facilement les erreurs dans l'application et son état.

De plus, des outils comme metricbeat permettent de visualiser l'état du serveur, comme l'utilisation du CPU ou de la mémoire du serveur. Cet outil peut analyser l'application Docker afin de récupérer ces différentes informations.