Pierre-Etienne BERNARD

E°4

Compte Rendu du Projet E4 WMD Docker

Table des matières

[Introduction 2](#_Toc181203216)

[1. Importation des Applications 2](#_Toc181203217)

[3. Configuration du Docker Compose 4](#_Toc181203218)

[4. Mise en Place des Réplicas 6](#_Toc181203219)

[5. Configuration du Reverse Proxy 6](#_Toc181203220)

[6. Vérification du Fonctionnement 9](#_Toc181203221)

[8. Publication des Images Docker sur Docker Hub 9](#_Toc181203222)

[Diagramme de l’Architecture 12](#_Toc181203223)

[Questions et Réponses 12](#_Toc181203224)

# Introduction

Dans le cadre du projet E4 WMD Docker, l’objectif était de déployer cinq applications dans un environnement de développement à l’aide de la technologie Docker et d’assurer leur accessibilité externe. Ce rapport présente les démarches suivies, les configurations appliquées, ainsi que les solutions adoptées pour répondre aux exigences spécifiques du client.

**Étapes de Réalisation**

# 1. Importation des Applications

La première étape consistait à importer les applications via la commande git clone. Cela a permis de télécharger les fichiers sources nécessaires pour chaque application dans l’environnement de développement.

Une image contenant texte, capture d’écran, Logiciel multimédia, Police

Description générée automatiquement2. Création et Configuration des Fichiers Docker

Chaque application nécessitait une configuration Docker spécifique. Les modifications suivantes ont été apportées :

* **Dockerfile** : Adaptation des instructions de build pour chaque application afin de garantir une optimisation des images.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

**Appseed.conf** : Ajustement des paramètres de configuration pour répondre aux besoins du projet. Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* **Gunicorn** : Paramétrage du serveur d’application pour optimiser la gestion des processus.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

* Et le **Docker-compose.yaml** de l’appli : Une image contenant texte, capture d’écran, logiciel

  Description générée automatiquement

# 3. Configuration du Docker Compose

Un fichier docker-compose.yml a été créé pour orchestrer le déploiement des cinq applications. Voici les ports qui ont été assignés :

* **App1** : port 80
* **App2** : port 8080
* **App3** : port 5001
* **App4** : port 5086
* **App5** : port 7000

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Une fois le fichier configuré, le déploiement des conteneurs a été lancé via la commande suivante :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

# 4. Mise en Place des Réplicas

Afin de tester la résilience des applications, un système de réplicas a été mis en place pour une des applications (au choix) en ajoutant 5 réplicas dans le fichier docker-compose.yml. Voici la commande utilisée pour déployer les réplicas :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Après cette modification, la commande « *docker compose up -d* » a été relancée pour appliquer les changements et vérifier le bon fonctionnement des réplicas. Une image contenant texte, logiciel, Logiciel multimédia, Police

Description générée automatiquement

# 5. Configuration du Reverse Proxy

Pour permettre un accès centralisé, un reverse proxy NGINX a été mis en place pour les applications suivantes : **App1, App3, App4**, et **App5**. **App2** a été exclue de cette configuration en raison de sa charge élevée liée aux réplicas.

Le fichier docker-compose.yml a été modifié pour inclure le proxy, et un dossier de configuration pour NGINX a été créé afin de gérer le routage.

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Après avoir modifier ce fichier il fallait aussi créer un dossier de configuration nginx :  
Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

# 6. Vérification du Fonctionnement

Après avoir configuré le reverse proxy et les réplicas, la commande docker compose up -d a été exécutée de nouveau pour vérifier le bon fonctionnement des applications. Les tests ont confirmé que les quatre applications utilisant le proxy, ainsi que l’application avec ses cinq réplicas, fonctionnaient correctement.

Une image contenant texte, logiciel, Police, capture d’écran

Description générée automatiquement

**7. Vérification de la Taille des Images**

La taille des images Docker a été vérifiée à l’aide de la commande suivante pour s’assurer qu’elles étaient optimisées :

Une image contenant capture d’écran, texte, Logiciel multimédia, multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Cette étape permet de contrôler et réduire l'empreinte de chaque application déployée.

# 8. Publication des Images Docker sur Docker Hub

Le client a autorisé la publication des images des applications en open source pour en faire bénéficier la communauté. La démarche suivante a été réalisée pour publier les images sur Docker Hub et configurer l’Infrastructure as Code (IaC) en conséquence.

**Étapes de Publication sur Docker Hub**

**Connexion à Docker Hub**  
La première étape consistait à se connecter à Docker Hub en utilisant la commande suivante :

Une image contenant capture d’écran, multimédia, texte, Logiciel multimédia

Description générée automatiquement

**Construction et Push des Images**  
Pour chaque application, les étapes suivantes ont été exécutées :

* **Build de l’image Docker** : Création de l’image à partir des fichiers de l’application.
* **Tag de l’image** : Attribution d’un tag unique correspondant au repository Docker Hub.
* **Push de l’image** : Publication de l’image sur Docker Hub.

Exemple pour **App1** :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Ces mêmes commandes ont été répétées pour les cinq autres applications.

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

**Mise à Jour du Fichier docker-compose.yml**  
Enfin, le fichier docker-compose.yml a été modifié pour utiliser les images hébergées sur Docker Hub. Cette mise à jour permet de lancer les applications directement depuis Docker Hub, garantissant ainsi leur portabilité et leur accessibilité pour d’autres utilisateurs.

Une image contenant texte, capture d’écran, ordinateur, logiciel

Description générée automatiquement

# Diagramme de l’Architecture

Une image contenant diagramme, capture d’écran, Tracé, ligne

Description générée automatiquement

# Questions et Réponses

1. **Pourquoi utiliser Docker dans ce projet ?**

Docker simplifie le déploiement des applications en les isolant dans des conteneurs avec toutes leurs dépendances. Cela garantit la portabilité, la cohérence entre les environnements et réduit les risques d’incompatibilités.

1. **Comment Docker s’inscrit-il dans la méthodologie DevOps ?**

Docker facilite la collaboration entre développement et exploitation en standardisant l’environnement de déploiement. Il permet des déploiements rapides et fiables tout au long du cycle de vie DevOps, soutenant l’intégration et le déploiement continus (CI/CD).

1. **Exemple de workflow pour tester et publier l’application automatiquement :**

* **Commit** : Les développeurs poussent leur code sur Git.
* **Build et Test** : Une nouvelle image Docker est construite et testée automatiquement.
* **Push sur Docker Hub** : L’image validée est versionnée et poussée sur Docker Hub.
* **Déploiement** : L’image est déployée en production via un orchestrateur.
* **Surveillance** : Le fonctionnement en production est surveillé pour feedback et amélioration.