Nantes Université

XMA1IU3042
Année 2023-2024

Projet DevOps Problème 2

Binôme :

ATALLA Salim Alma - TP1, groupe A
BAMOUH Léna Alma - TP1, groupe A

Description de l'application

Laragigs est une application web conçue avec Laravel pour la gestion des offres d'emploi. Elle offre une fonctionnalité d'authentification permettant aux utilisateurs de créer des comptes pour la création et la gestion des offres. Même sans se connecter, les utilisateurs peuvent rechercher des offres d'emploi selon divers critères.

Contexte de développement :

L'application a été développée pour simplifier la gestion des offres d'emploi. Son objectif est de fournir une plateforme pour les utilisateurs afin qu'ils puissent créer, rechercher et gérer des offres d'emploi de manière efficace.

Ce projet a été effectué au cours du stage de dernière année de Licence informatique de Salim ATALLA.

Objectifs et fonctionnalités :

- Système d'authentification utilisateur pour la création de comptes et la gestion des offres.
- Fonctionnalité CRUD (Create, Read, Update, Delete) pour les offres d'emploi.
- Recherche avancée basée sur les titres, descriptions, tags, localisation et entreprise.
- Vue détaillée pour chaque offre d'emploi, fournissant des informations complètes.

Technologies utilisées :

- Laravel : Framework PHP côté serveur pour le développement de l'application.
- MySQL : Base de données utilisée pour stocker les données des offres d'emploi.
- PHPMyAdmin : Outil de gestion de base de données web pour faciliter la gestion des données.
- Nginx : Serveur web utilisé pour gérer les requêtes HTTP.

Test de l'exécution en conteneur :

Pour tester l'exécution en conteneur, vous pouvez suivre les instructions fournies dans les fichiers suivants du projet Laragigs :

1. *build-and-deploy.md* : Ce fichier contient des instructions détaillées sur la construction et le déploiement de l'application en utilisant des conteneurs. Il présente les étapes nécessaires pour mettre en place l'application à l'aide des différents conteneurs (MySQL, PHPMyAdmin, PHP, Nginx).

2. *commands.md* : Ce fichier contient les commandes nécessaires pour exécuter les conteneurs. Il fournit les commandes spécifiques à exécuter pour lancer les différents conteneurs nécessaires au fonctionnement de l'application.

Pour tester l'exécution en conteneur, il faut suivre les étapes décrites dans ces fichiers. Ils fournissent des instructions pas à pas sur la configuration et l'exécution des conteneurs pour déployer l'application Laragigs.

3.1. Phase d'analyse

Question 1

Pour ce projet, nous allons avoir besoin de 4 conteneurs différents. Il y en aura un pour le déploiement de la base de données, un autre pour le déploiement d'un outil de gestion pour la base de données, encore un autre pour l'exécution du code côté serveur et un dernier pour le déploiement du front-end.

Question 2

➢ Pour la base de données

On utilise l'image amd64/mysql comme base pour le conteneur MySQL en raison de sa simplicité et de sa fiabilité. Elle fournit également une configuration de base pour un serveur MySQL.

Le script de construction d'image doit configurer MySQL selon les besoins du projet Laragigs. Les fichiers à inclure sont principalement des scripts SQL pour l'initialisation et la configuration de la base de données, des éventuels fichiers de configuration personnalisée pour MySQL, et un script d'entrée pour démarrer le serveur MySQL.

Pour réduire la taille de l'image, il faut limiter les couches d'images, nettoyer les caches et les fichiers temporaires après l'installation des dépendances. Le point d'entrée de l'image MySQL est le démarrage du serveur MySQL pour rendre la base de données accessible.

Pour utiliser l'image de conteneur, des variables d'environnement peuvent configurer l'application et des montages permettent la persistance des données.

➢ Pour l'outil de gestion de la base de données

L'image qu'on a utilisé phpmyadmin/phpmyadmin car elle fournit un outil préconfiguré pour gérer une base de données MySQL via une interface web (PhpMyAdmin).

Si on devait reconstruire l'image, le script pourrait inclure des configurations spécifiques telles que des personnalisations de l'interface, des configurations de sécurité.
L'image PhpMyAdmin étant préconfigurée, elle ne nécessite pas de fichiers supplémentaires pour fonctionner.

Pour limiter la taille de l'image, il est possible de supprimer les fichiers temporaires et les dépendances inutiles après l'installation. Cependant, dans notre cas, puisqu'il s'agit d'une image déjà fournie, cette optimisation peut être limitée.

Le point d'entrée de l'image PhpMyAdmin est le lancement du serveur PhpMyAdmin pour fournir l'interface web.

Pour utiliser l'image de conteneur, des variables d'environnement peuvent configurer l'application et il faut publier le port sur lequel PhpMyAdmin écoute pour permettre l'accès aux autres composants de l'application.

➢ Pour l'exécution du code, côté serveur

L'image de base utilisée est amd64/php:8.3-fpm-alpine3.19 car elle fournit un environnement PHP fonctionnant avec FPM (PHP-FPM) basé sur Alpine Linux, ce qui est souvent léger et adapté aux conteneurs.

Le script doit installer les extensions PHP nécessaires à l'application Laragigs, telles que pdo, pdo_mysql pour interagir avec la base de données, et zip pour potentiellement manipuler des fichiers compressés.

Les fichiers nécessaires sont ceux qui composent l'application Laravel (le code source dans le dossier src) ainsi que les configurations spécifiques du serveur PHP-FPM.

Pour limiter la taille de l'image, il faut nettoyer les caches, supprimer les fichiers temporaires et de minimiser le nombre de couches d'image.

Le point d'entrée serait probablement le démarrage du serveur PHP-FPM pour exécuter le code PHP de l'application Laragigs.

Pour utiliser l'image de conteneur, il faut créer un volume pour stocker le code source de l'application Laravel, ce qui permet une gestion plus flexible du code.

➢ Pour le front-end

L'image de base utilisée est amd64/nginx:stable-alpine. Cette image est choisie car elle fournit un serveur NGINX léger basé sur Alpine Linux, souvent utilisé pour servir des applications web statiques.

Le script doit configurer NGINX pour servir correctement les fichiers statiques du frontend de l'application Laragigs. L'image NGINX inclura le fichier de configuration default.conf nécessaire pour servir l'application frontend.

Pour limiter la taille de l'image NGINX, on peut nettoyer les caches et les fichiers temporaires après la configuration de NGINX.

Le point d'entrée de l'image NGINX est le démarrage du serveur NGINX pour servir les fichiers statiques.

Pour utiliser l'image de conteneur, il faut créer un volume pour fournir les fichiers statiques de l'application Laravel, ce qui permet une gestion plus flexible du code. Il faut également exposer le port NGINX nécessaire pour permettre l'accès au frontend depuis l'extérieur.

Question 3

➢ Pour la base de données

Réseaux virtuels : Connecté au réseau laragigs_network.
Montages de répertoires/fichiers : Montage d'un volume pour la persistance des données de la base de données.
Variables d'environnement :

- DB_CONNECTION=mysql
- DB_HOST=mysql

- DB_PORT=3306
- DB_DATABASE=laragigs
- DB_USERNAME=user
- DB_PASSWORD=password

Publications de port : Aucune publication directe nécessaire.

➢ Pour l'outil de gestion de la base de données

Réseaux virtuels : Connecté au réseau laragigs_network pour permettre la communication avec le conteneur MySQL.

Montages de répertoires/fichiers : Pas de montages nécessaires.

Variables d'environnement :

- PMA_HOST=laragigs_mysql
- PMA_PORT=3306
- MYSQL_USER=user
- MYSQL_PASSWORD=password
- MYSQL_ROOT_PASSWORD=root

Publications de port : Le port 80 (ou un autre port défini) doit être publié pour accéder à l'interface web de PhpMyAdmin.

➢ Pour l'exécution du code, côté serveur

Réseaux virtuels : Connecté au réseau laragigs_network pour permettre la communication avec les autres composants du déploiement.

Montages de répertoires/fichiers : Le volume laragigs_volume:/var/www/html monté pour stocker le code source de l'application Laravel.

Variables d'environnement :

- Aucune variable d'environnement spécifique n'est configurée dans cet exemple, mais des variables pour les paramètres PHP peuvent être définies au besoin.

Publications de port : Aucune publication directe nécessaire.

➢ Pour le front-end

Réseaux virtuels : Connecté au réseau laragigs_network pour permettre la communication avec les autres composants du déploiement.

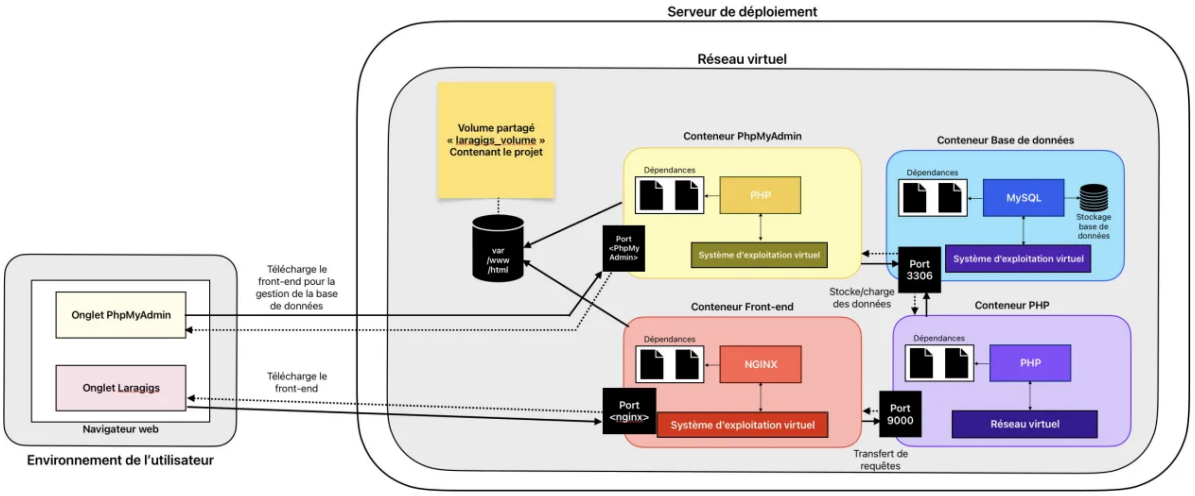
Montages de répertoires/fichiers : Le volume laragigs_volume:/var/www/html est monté pour servir les fichiers statiques de l'application Laragigs.

Variables d'environnement :

- Aucune variable d'environnement spécifique n'est configurée dans cet exemple.

Publications de port : Le port 80 (ou un autre port défini) doit être publié pour permettre l'accès au frontend de l'application Laragigs depuis l'extérieur.

Question 4



3.2. Phase technique

➢ <u>Image de la base de données, MySQL</u>

Fichiers Nécessaires pour Créer l'Image :

- Script Dockerfile: Ce fichier décrit les étapes de construction de l'image. Il contient les commandes Docker pour installer MySQL, configurer les paramètres nécessaires, et éventuellement d'autres dépendances ou configurations spécifiques.

Documentation Technique :

- Utilisation : Il faut monter les volumes pour la persistance des données et définir les variables d'environnement pour la configuration de la base de données (nom de la base de données, nom d'utilisateur, mot de passe).
- Paramétrage : il faut définir MYSQL_ROOT_PASSWORD et MYSQL_DATABASE pour la configuration initiale de la base de données.

Commandes de Construction et de Publication :

```
podman image build --tag laragigs:mysql -f Containerfile_mysql

podman image push laragigs:mysql docker-registry.univ-nantes.fr/e191350p/laragigs:mysql
```

➢ Image de l'outil de gestion de la base de données, PhpMyAdmin

Fichiers nécessaires pour créer l'image de conteneur :

- Fichier de configuration de PhpMyAdmin : Un fichier de configuration adapté à l'application Laragigs doit être inclus pour que PhpMyAdmin puisse se connecter correctement à la base de données MySQL.

Documentation technique pour l'utilisation de l'image :

- Paramétrage : il faut définir PMA_HOST, PMA_PORT, MYSQL_USER, MYSQL_PASSWORD, MYSQL_ROOT_PASSWORD pour la connexion à la base de données.

Commandes de Construction et de Publication :

```
podman image build --tag laragigs:phpmyadmin -f Containerfile_phpmyadmin .

podman image push laragigs:phpmyadmin
docker-registry.univ-nantes.fr/e191350p/laragigs:phpmyadmin
```

➢ Image pour l'exécution du code côté serveur, Php

Fichiers nécessaires pour créer l'image de conteneur :

- Dossier source de l'application Laravel : Les fichiers PHP et les dépendances de l'application Laravel doivent être inclus. Ils seront exécutés par le serveur PHP-FPM.

Documentation technique pour l'utilisation de l'image :

- Utilisation : Il faut monter les volumes pour exécuter le code source de l'application Laravel à partir de l'extérieur du conteneur.

Commandes de Construction et de Publication :

```
podman image build --tag laragigs:php -f Containerfile_php .

podman image push laragigs:php docker-registry.univ-nantes.fr/e191350p/laragigs:php
```

➢ Image du front-end, NGINX

Fichiers Nécessaires pour Créer l'Image :

- Script Dockerfile : Définit la configuration NGINX.
- Volume commun : Utilise les fichiers partagés avec le conteneur PHP pour savoir ce qui doit être affiché

Documentation Technique :

- Utilisation : Il faut monter les volumes pour servir les fichiers statiques du front-end depuis l'extérieur du conteneur.

Commandes de Construction et de Publication :

```
podman image build --tag laragigs:nginx -f Containerfile_nginx .

podman image push laragigs:nginx docker-registry.univ-nantes.fr/e191350p/laragigs:nginx
```

Commandes Nécessaires

```
# Création d'un réseau virtuel
podman network create laragigs_network

# Création du volume commun
podman volume create laragigs_volume

# Lancement du conteneur pour la base de données
podman run -d  --name laragigs_mysql  --network laragigs_network \
 -v <path to mysql data folder>:/var/lib/mysql  -e MYSQL_ROOT_PASSWORD=root \
 -e MYSQL_DATABASE=laragigs  -e MYSQL_USER=user \
 -e MYSQL_PASSWORD=password  --userns=keep-id \
  docker-registry.univ-nantes.fr/e191350p/laragigs:mysql

# Lancement du conteneur pour PhpMyAdmin
podman run -d  --name laragigs_phpmyadmin --network laragigs_network \
  -e PMA_HOST=laragigs_mysql -e PMA_PORT=3306 -e MYSQL_USER=user \
  -e MYSQL_PASSWORD=password -e MYSQL_ROOT_PASSWORD=root \
  -p <port for phpmyadmin>:80 \
  docker-registry.univ-nantes.fr/e191350p/laragigs:phpmyadmin

# Lancement du conteneur Php
podman run -d --name laragigs_php --network laragigs_network \
  -v laragigs_volume:/var/www/html docker-registry.univ-nantes.fr/e191350p/laragigs:php

# Lancement du conteneur pour le front-end
podman run -d --name laragigs_nginx --network laragigs_network \
  -v laragigs_volume:/var/www/html -p <port for the app>:80 \
  docker-registry.univ-nantes.fr/e191350p/laragigs:nginx
```