TP Docker : Déploiement de WordPress avec Docker Compose

Introduction

Dans ce TP (disponible ici), nous allons déployer localement WordPress sur votre ordinateur en utilisant des conteneurs et un fichier `docker-compose.yml`. Ce cas est une véritable utilisation de Docker : avant de déployer un logiciel dans une organisation, on peut le tester en local pour vérifier qu'il répond bien aux besoins.

Qu'est-ce que WordPress ?

WordPress est un *CMS* (Content Management System) open-source, utilisé pour créer des sites web et des blogs sans coder.

- Interface conviviale, thèmes et plugins.
- Gestion flexible du contenu.
- Optimisé pour le référencement.
- Largement adopté (43% des sites web en 2023).

Objectifs de l'architecture

D'après la documentation, il nous faudra :

- Un conteneur `WordPress`.
- Un conteneur pour la base de données (MariaDB).
- Des volumes pour stocker :
  - Les plugins, thèmes et fichiers WordPress.
  - Les données des utilisateurs (comptes, contenu, etc.).
- Des variables d'environnement pour connecter WordPress à la base.
- Une politique de redémarrage en cas de plantage.

Étape 1 : Préparation du projet

Commencez par créer un **répertoire de travail** dédié à ce TP, par exemple `mon-wordpress`. Placez-vous ensuite dans ce répertoire, puis créez un fichier `docker-compose.yml` qui contiendra la description de l'architecture multi-conteneurs.

Au départ, définissez simplement deux services vides :

```
services:
  db:
  wordpress:
```

Étape 2 : Choix des images

```
services:
  db:
    image: mariadb:10.6.4-focal
  wordpress:
    image: wordpress:latest
```

Étape 3 : Communication entre conteneurs

```
services:
  db:
    image: mariadb:10.6.4-focal
    expose:
      - 3306
      - 33060
  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
```

`expose` : communication interne entre conteneurs. `ports` : accès externe à WordPress depuis l'hôte.

Étape 4 : Ajout de volumes

```
services:
  db:
    image: mariadb:10.6.4-focal
    expose:
      - 3306
      - 33060
    volumes:
      - db_data:/var/lib/mysql
  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    volumes:
      - wp_data:/var/www/html

volumes:
  db_data:
  wp_data:
```

Étape 5 : Variables d'environnement

```
services:
  db:
    image: mariadb:10.6.4-focal
    expose:
      - 3306
      - 33060
    volumes:
      - db_data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=wordpress
      - MYSQL_PASSWORD=wordpress
  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    volumes:
      - wp_data:/var/www/html
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=wordpress
      - WORDPRESS_DB_PASSWORD=wordpress
      - WORDPRESS_DB_NAME=wordpress

volumes:
  db_data:
  wp_data:
```

Étape 6 : Politique de redémarrage

Il est recommandé de préciser une politique de redémarrage pour chaque service. Ajoutez la ligne suivante dans la définition des services `db` et `wordpress` :

```
restart: always
```

Étape 7 : Paramètre d'authentification MariaDB

Par défaut, MariaDB peut utiliser différents plugins d'authentification pour gérer les utilisateurs. Certaines versions récentes activent le plugin `caching_sha2_password`, qui n'est pas toujours compatible avec WordPress et peut entraîner des erreurs de connexion (par exemple : `"authentication method unknown to the client"`).

Pour garantir la compatibilité, nous allons forcer MariaDB à utiliser le plugin d'authentification classique `mysql_native_password`. Cela se fait grâce à la directive `command:`, qui permet de modifier la commande de démarrage du conteneur.

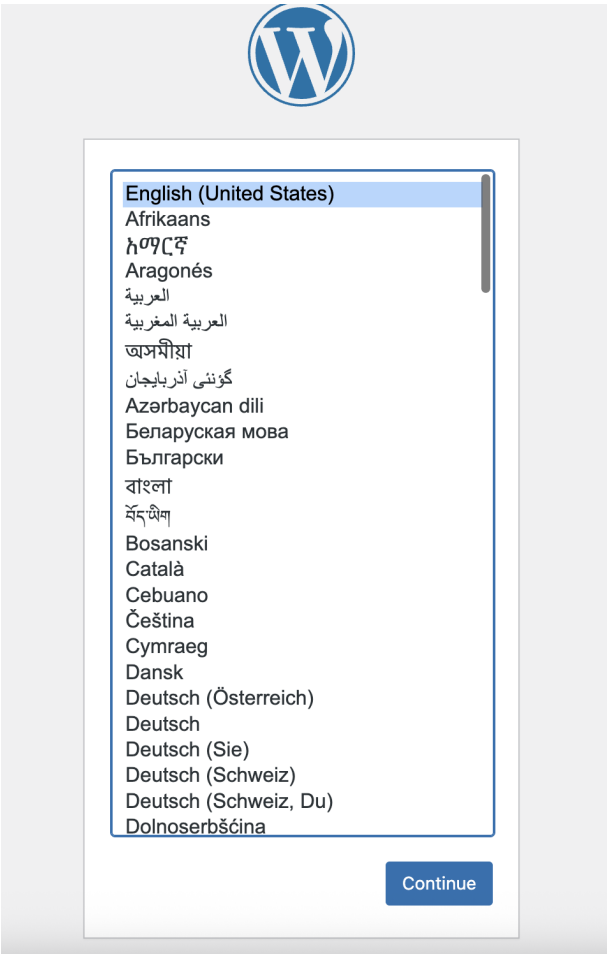
Ajoutez donc la ligne suivante dans la configuration du service `db` :

```
command: '--default-authentication-plugin=mysql_native_password'
```

Étape 8 : Lancement et test

```
docker compose up -d
```

Aller sur `http://localhost` pour configurer WordPress.



Étape 9 : Nettoyage

Lister les volumes :

```
docker volume ls
```

Supprimer les volumes créés :

```
docker volume rm db_data
docker volume rm wp_data
```