

Déploiement - WordPress avec Docker

Prérequis

- *Logiciels* : Docker et Docker Compose installés sur votre machine.
- *Connaissances* : Notions de base sur Docker, la gestion des volumes et des réseaux.
- *Matériel* : Terminal, éditeur de texte et navigateur web.

1. Introduction

Ce document présente le processus de déploiement de WordPress en utilisant Docker et Docker Compose. Il inclut les étapes suivies, les choix de configuration, les difficultés rencontrées et des perspectives d'amélioration.

2. Étapes de developpement

Pour commencer, il est essentiel de définir un espace de travail, un dossier dédié à ce projet.

2.1. Création du dossier de travail

```
mkdir tp-wordpress && cd tp-wordpress
```

2.2. Rédaction du fichier docker-compose.yml

- Un fichier **docker-compose.yml** a été créé dans "tp-wordpress", avec la configuration suivante :

```
services:
  wordpress:
    image: wordpress:latest
    container_name: wordpress
    restart: always
    ports:
      - "8082:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress_data:/var/www/html
    depends_on:
      - db
    networks:
      - wp-network

db:
  image: mysql:8.0
  container_name: mysql
  restart: always
  environment:
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpress
    MYSQL_PASSWORD: wordpress
    MYSQL_ROOT_PASSWORD: rootpassword
  volumes:
    - db_data:/var/lib/mysql
```

```

networks:
  - wp-network

phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  container_name: phpmyadmin
  restart: always
  ports:
    - "8081:80"
  environment:
    PMA_HOST: db
    MYSQL_ROOT_PASSWORD: rootpassword
  depends_on:
    - db
  networks:
    - wp-network

volumes:
  wordpress_data:
  db_data:

networks:
  wp-network:
    driver: bridge

```

2.3. Démarrage des conteneurs

- Lancement de l'environnement avec :

```
docker compose up -d
```

- Lister les conteneurs en cours d'exécution:

```
docker ps
```

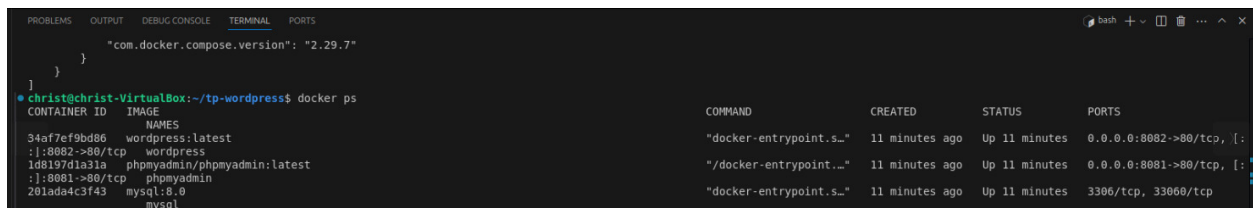


Figure 1: Mes conteneurs

3. Choix de Configuration

3.1. Configuration de WordPress

Ce service exécute un conteneur WordPress avec les paramètres suivants :

- **Image utilisée** : wordpress:6.7 (une version stable de WordPress "Rollins").
- **Nom du conteneur** : wordpress.
- **restart: always** (Redémarrage automatique) -> le conteneur sera relancé en cas de crash.
- **Port exposé** : 8082:80 -> Accès à l'interface via <http://localhost:8082>.

- **Variables d'environnement :**
 - WORDPRESS_DB_HOST: db → Indique que la base de données est hébergée par le service db.
 - WORDPRESS_DB_USER: wordpress → Nom d'utilisateur pour la base de données.
 - WORDPRESS_DB_PASSWORD: wordpress → Mot de passe pour la base de données.
 - WORDPRESS_DB_NAME: wordpress → Nom de la base de données WordPress.
- **Volume monté :**
 - wordpress_data:/var/www/html → Permet de sauvegarder les fichiers WordPress en dehors du conteneur pour une persistance des données.
- **Dépendance :**
 - depends_on: db → WordPress ne démarrera qu'après le démarrage de la base de données MySQL.
- **Réseau :**
 - Connecté au réseau wp-network.

3.2. Configuration de MySQL(db)

Ce service exécute un conteneur MySQL 8.0, qui sert de base de données pour WordPress.

- **Image utilisée :** mysql:8.0. **Nom du conteneur :** mysql.
- Redémarrage automatique : restart: always.
- **Variables d'environnement :**
 - MYSQL_DATABASE: wordpress → Création automatique d'une base de données wordpress.
 - MYSQL_USER: wordpress → Nom d'utilisateur wordpress.
 - MYSQL_PASSWORD: wordpress → Mot de passe utilisateur wordpress.
 - MYSQL_ROOT_PASSWORD: rootpassword → Mot de passe administrateur (root).
- **Volume monté :**
 - db_data:/var/lib/mysql → Stocke les données MySQL de manière persistante.
- **Réseau :** Connecté au réseau wp-network.

3.3. Configuration de phpmyadmin (*Interface Graphique*)

Ce service installe PhpMyAdmin, une interface web pour gérer la base de données MySQL.

- **Image utilisée :** phpmyadmin/phpmyadmin:latest.
- **Nom du conteneur :** phpmyadmin. Redémarrage automatique : restart: always.
- **Port exposé :** 8081:80 (accessible via http://localhost:8081).
- **Variables d'environnement :**
 - PMA_HOST: db → Indique que la base de données cible est db.
 - MYSQL_ROOT_PASSWORD: rootpassword → Définit le mot de passe root de MySQL.
- **Dépendance :**
 - depends_on: db → PhpMyAdmin ne démarre qu'après la base de données.
- **Réseau :** Connecté au réseau wp-network.

Définition du Réseau

- En utilisant un réseau bridge, les conteneurs peuvent communiquer entre eux via leurs noms de service (db, wordpress, phpmyadmin) sans exposer leurs ports à tout le réseau de l'hôte.
- **Configuration du site**
 - Accéder à l'interface WordPress via localhost:
 - Choix de la langue → suivre les étapes

Résultat:

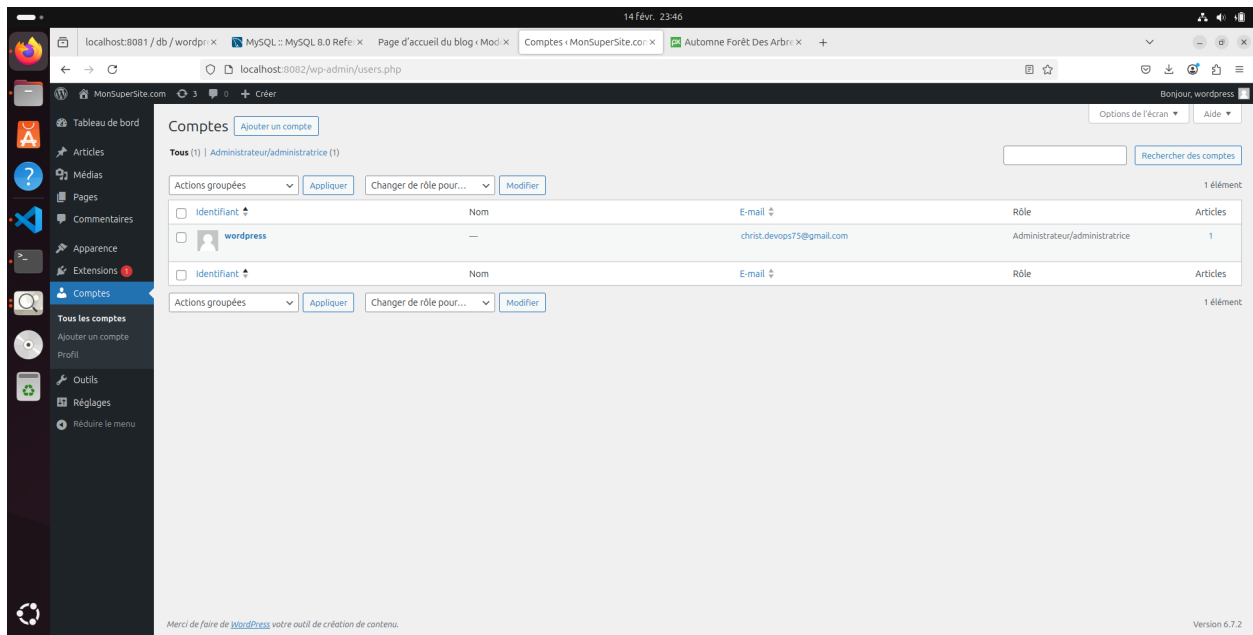


Figure 2: interface admin

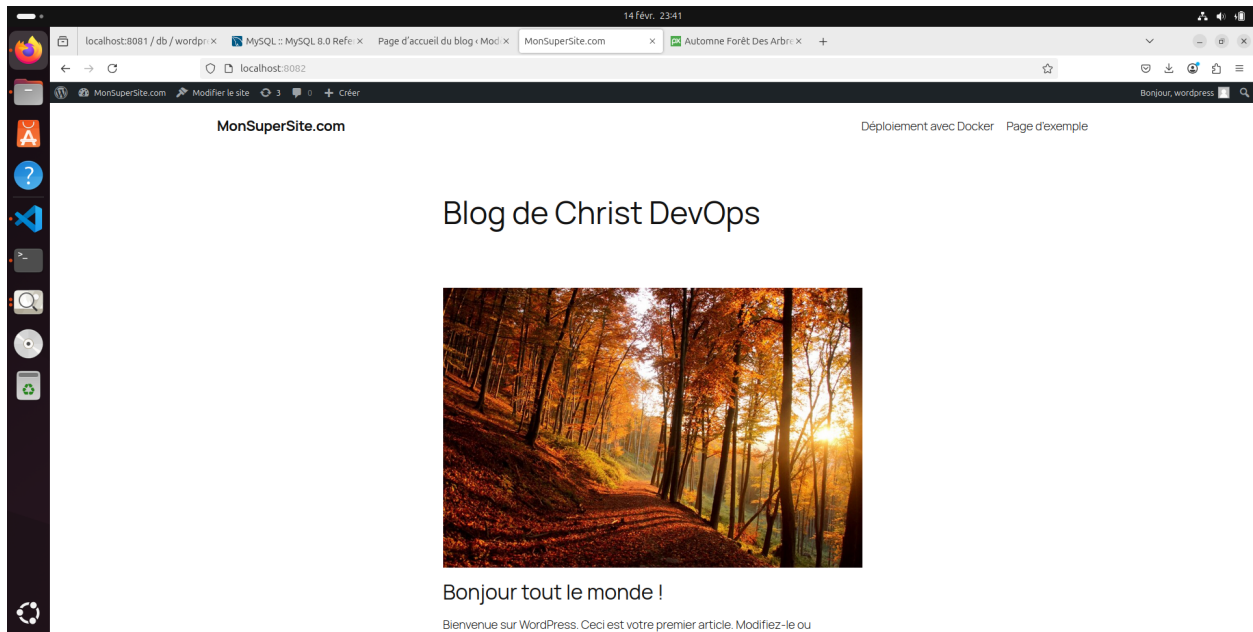


Figure 3: Test de création de pages et d'ajout de médias dans Mon site

3.4. Résumé du Fonctionnement **MySQL (db)** démarre et crée une base de données wordpress. **WordPress** qui permet de créer un site web, démarre et se connecte à la base de données MySQL via db. Accessible sur <http://localhost:8082>
PhpMyAdmin démarre et permet d'accéder à la base de données via <http://localhost:8081>.

4. Difficultés Rencontrées et Solutions

Problème	Cause	Solution
Erreur : Port 8080 déjà utilisé	Un autre service occupait ce port	Changer le port WordPress 8080:80 en 8081:80 dans docker-compose.yml
Problème de connexion entre WordPress et MySQL	Mauvaise configuration des variables d'environnement	Vérification des valeurs et redémarrage des conteneurs
Base de données non persistante après redémarrag	Typo dans la configuration de volume pour MySQL	Correction de typo /var/libn/mysql -> volume db_data:/var/lib/mysql

5. Conclusions et Perspectives

Ce déploiement permet une mise en place rapide et efficace d'un environnement complet pour héberger WordPress avec une base de données MySQL et une interface d'administration PhpMyAdmin, dans un environnement conteneurisé.

Perspectives d'améliorations possibles

- Sécurisation en utilisant des mots de passe plus robustes et des fichiers .env.
- Utilisation de Traefik, apache2 ou Nginx pour un meilleur reverse-proxy et gestion SSL.
- Automatisation avec Ansible pour déployer WordPress sur plusieurs serveurs.
- Automatiser les sauvegardes en ajoutant un conteneur de backup pour MySQL
- Mettre en place un load_balancer et le monitoring pour le site en production.
- Utiliser Redis pour activer la mise en cache de WordPress (améliorer les performances).