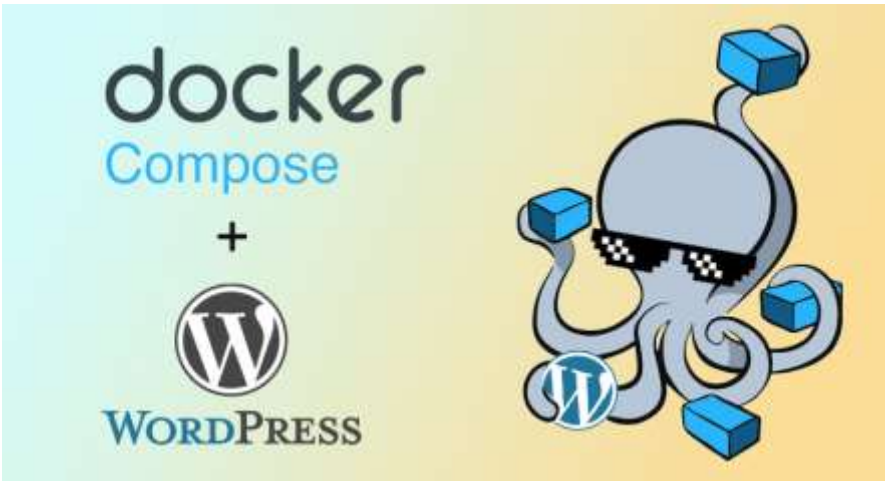


TEST TECHNIQUE

Déploiement d'une instance WordPress avec MySQL en local via Docker Compose

COMPTE RENDU

Le rapport de déploiement

Préparé par : Alcée LOUMOUAMOU

Table des matières

1 Introduction

Dans ce projet, j'ai déployé un site WordPress en local en utilisant Docker Compose. L'objectif était de créer un environnement fonctionnel avec une base de données MySQL, tout en s'assurant que les données soient conservées même après l'arrêt des conteneurs.

Grâce à Docker, j'ai pu simplifier l'installation et la configuration de WordPress sans avoir à tout installer manuellement sur mon ordinateur. Ce rapport explique les étapes que j'ai suivies, les choix que j'ai fait et les problèmes que j'ai rencontrés avec leurs solutions.

2 Vérification des prérequis

Avant de commencer, j'ai voulu m'assurer que Docker et Docker Compose étaient bien installés sur mon ordinateur. Pour ça, j'ai utilisé les commandes suivantes dans l'invite de commande :

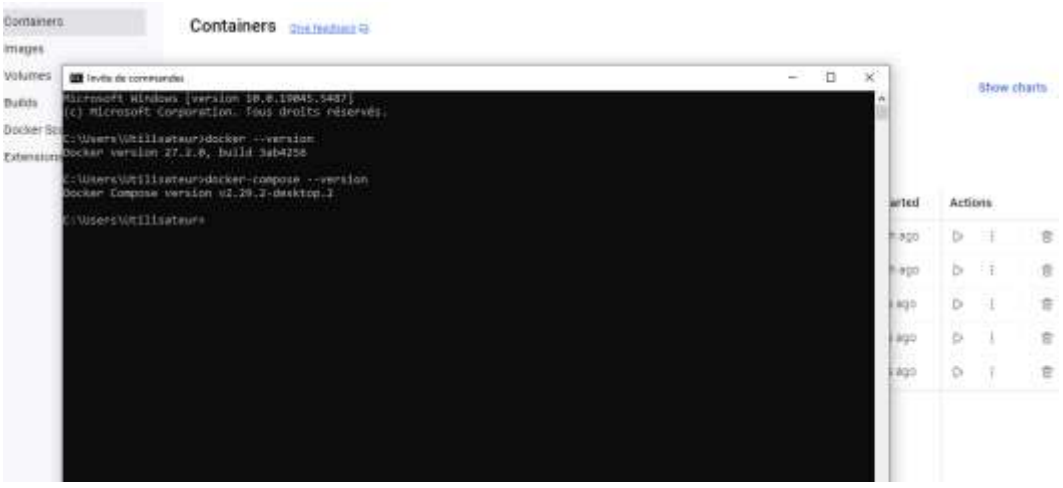
```
docker --version

docker-compose --version
```

Ces commandes permettent de vérifier les versions installées. Si elles renvoient bien un numéro de version, ça signifie que tout est correctement installé. Dans mon cas, les résultats étaient :

Docker : version 27.2.0, build 3ab4256

Docker Compose : version v2.29.2-desktop.2

Figure 1:l'image qui illustre les commandes



3 Création du dossier de travail

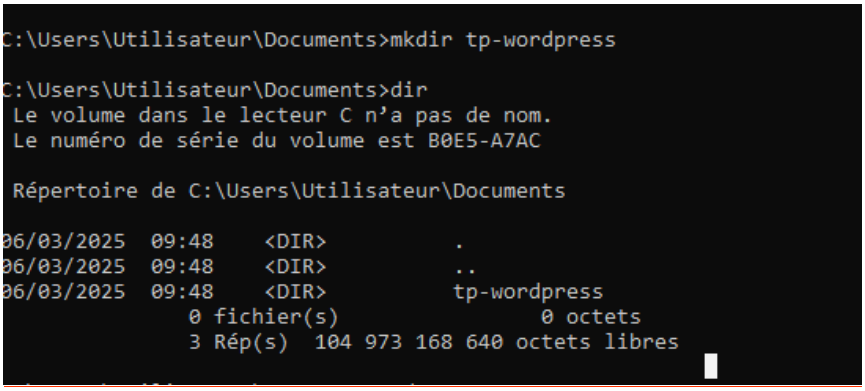
Une fois que j'ai vérifié que Docker était bien installé, j'ai créé un dossier pour stocker tous les fichiers liés à mon projet WordPress. Les commandes utilisées pour un ordinateur Windows sont :

cd Documents : Cette commande m'a permis de me déplacer dans le dossier **Documents** de mon ordinateur. Cela m'a évité de créer mon projet n'importe où et m'a aidé à mieux organiser mes fichiers.

mkdir tp-wordpress : Cette commande a servi à créer un dossier nommé **tp-wordpress** à l'intérieur de **Documents**. Ce dossier contiendra le **fichier docker-compose.yml** ainsi que toutes les données nécessaires au projet.

Après avoir exécuté ces commandes, j'ai bien vérifié que le dossier était créé en listant son contenu avec dir.

Figure 2:l'image sur les commandes décrites



4 Création du fichier docker-compose.yml

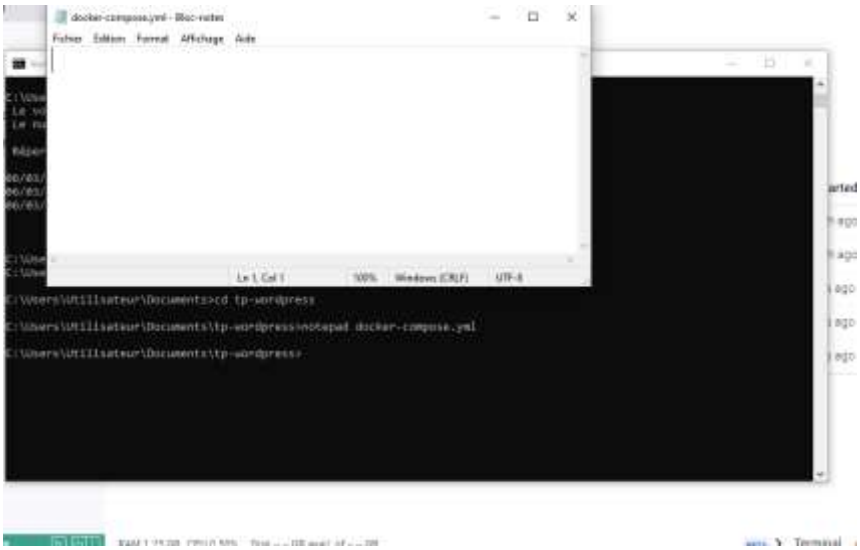
Après avoir créé mon dossier de travail, j'ai dû y ajouter un fichier docker-compose.yml. Ce fichier contient toutes les instructions pour lancer WordPress et MySQL avec Docker.

Les commandes utilisées :

cd tp-wordpress : Cette commande m'a permis d'entrer dans le dossier que j'avais créé précédemment. Ainsi, le fichier docker-compose.yml sera bien placé au bon endroit.

notepad docker-compose.yml : J'ai utilisé cette commande pour créer et ouvrir directement le fichier dans le Bloc-notes. Cela m'a permis d'écrire la configuration nécessaire pour déployer WordPress et MySQL. Une fois le fichier ouvert, j'ai copié le contenu YAML et enregistré le fichier sans l'extension .txt pour éviter tout problème.

Figure 3:l'image qui représente cette étape



Précision importante sur le contenu YAML copié :

L'image copié dans le site : https://hub.docker.com/_/wordpress

Les changements que j'ai fait et pourquoi

Quand j'ai écrit mon fichier docker-compose.yml, j'ai fait quelques ajustements pour qu'il fonctionne mieux et soit plus clair. Ce que j'ai changé et pourquoi :

Ajouter d'un réseau dédié : J'ai créé un réseau spécifique pour que WordPress et MySQL puissent bien communiquer ensemble sans interférences avec d'autres services.

=>Sans ça, Docker utilise son réseau par défaut, ce qui peut poser des problèmes.

Fixer un mot de passe root pour MySQL : Au début, MySQL générait un mot de passe root aléatoire à chaque démarrage. Le problème, c'est qu'on ne peut pas le voir.

=> J’ai mis un mot de passe fixe (rootpassword) pour pouvoir accéder facilement à la base de données si besoin.

Ajout de guillemets autour des ports

J’ai entouré les numéros de port avec des guillemets ("8080 :80") pour éviter d’éventuelles erreurs d’interprétation.

=>C’est juste une bonne pratique pour éviter des bugs.

Passage à MySQL 8.0

À la base, le fichier utilisait MySQL 5.7, mais j’ai choisi MySQL 8.0, qui est plus récent et plus performant.

=> Ça améliore la sécurité et la rapidité de la base de données.

les modifications que j’ai faites sur le fichier docker-compose.yml :

version : '3.8'

services :

wordpress:

image : wordpress

restart : always

ports :

- "8080 :80" **(ajouter des guillemets pour éviter les erreurs YAML)**

environment:

WORDPRESS_DB_HOST: db

WORDPRESS_DB_USER: exampleuser

WORDPRESS_DB_PASSWORD: examplepass

WORDPRESS_DB_NAME: exampledb

volumes:

- wordpress:/var/www/html

networks:

- wp-network **(ajouter un réseau dédié)**

db:

image: mysql:8.0 **(Passage à MySQL 8.0 au lieu de 5.7)**

restart: always

environment:

MYSQL_DATABASE: exampledb

MYSQL_USER: exampleuser

MYSQL_PASSWORD: examplepass

MYSQL_ROOT_PASSWORD: rootpassword **(Remplacement du mot de passe root aléatoire)**

volumes:

- db:/var/lib/mysql

networks:

- wp-network **(ajouter un réseau dédié)**

volumes:

wordpress:

db:

networks:

wp-network: (Définition du réseau)

Après j'enregistre le tp dans le dossier tp-wordpress

de commande

Ensuite, j'ai ouvert l'invite de commande (cmd) et j'ai navigué jusqu'à mon dossier en tapant :

cd %USERPROFILE%\Documents\tp-wordpress.

Cela m'a permis de me placer directement dans le bon dossier pour exécuter les commandes Docker.

Vérification du fichier docker-compose.yml.

Avant de continuer, j'ai vérifié que mon fichier était bien dans le dossier en tapant :

dir

Le fichier docker-compose.yml est bien apparu dans la liste, ce qui signifie qu'il est prêt à être utilisé.

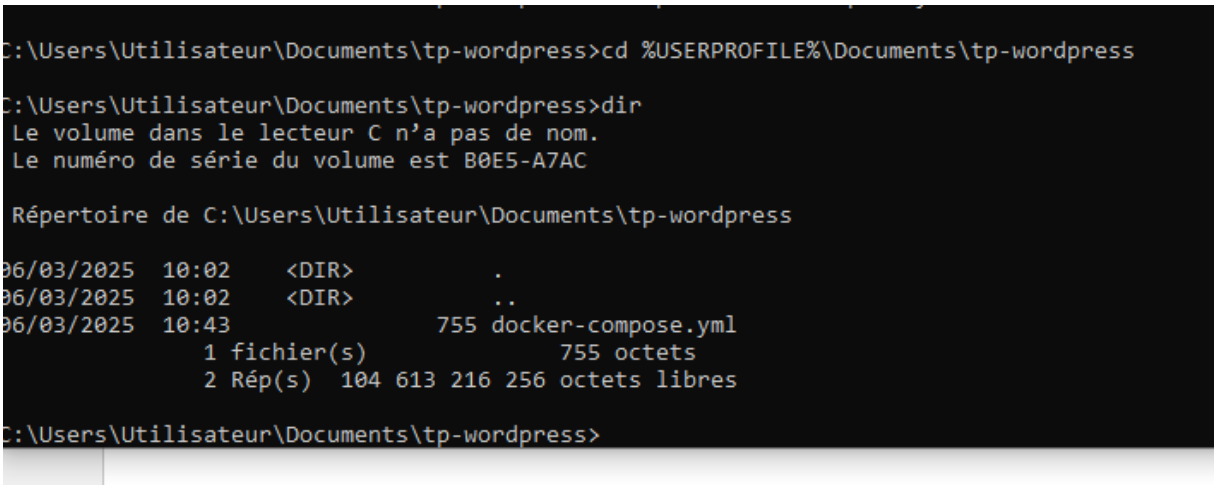
Configurer les volumes pour stocker les données

Mettre en place un réseau interne pour que les services puissent communiquer

Pourquoi j'ai fait tout ça ?

L’objectif était de déployer WordPress avec une base de données MySQL en local grâce à Docker. Avec cette configuration, tout est automatisé et je peux lancer mon site en quelques secondes seulement.

Figure 4:L’image qui représente les commandes utilisées



5 Lancement des conteneurs Docker

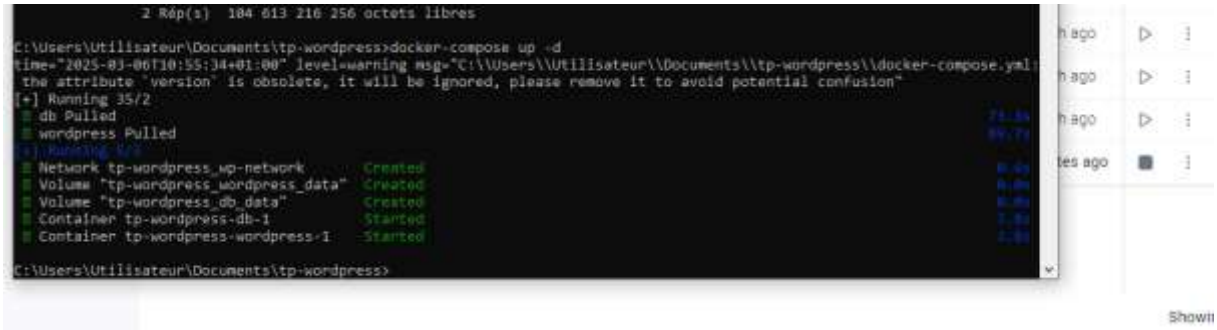
Pour démarrer WordPress et MySQL, j’ai tapé cette commande :

docker-compose up -d

Ce que ça fait : Télécharger WordPress et MySQL si je ne les ai pas déjà, crée et lance les conteneurs ; garde les fichiers et la base de données même si j’arrête tout, fait en sorte que WordPress et MySQL puissent communiquer entre eux, tourne en arrière-plan, donc je peux continuer à utiliser mon terminal. En gros, cette commande installe et démarre mon site automatiquement sans que j’aie besoin de tout configurer à la main.

En dessous une image qui montre les commandes et leurs résultats :

Figure 5:une image qui montre les commandes et leurs résultats



6 Vérification des conteneurs avec docker ps

Après avoir lancé mon environnement avec la commande docker-compose up -d, j'ai voulu vérifier que mes conteneurs étaient bien en cours d'exécution. Pour cela, j'ai utilisé la commande suivante : docker ps

À quoi ça sert ?

Cette commande permet d'afficher la liste des conteneurs qui tournent actuellement sur mon ordinateur. Grâce à elle, j'ai pu voir que :

- ✓ Un conteneur WordPress était bien actif, avec le port 8080 redirigé vers le port 80 du conteneur.
- ✓ Un conteneur MySQL fonctionnait aussi, prêt à gérer la base de données.
- ✓ Confirmer que mon environnement était bien lancé sans erreur.

Pour plus de précision, voici les résultats détaillés :

CONTAINER ID : L'identifiant unique du conteneur.

IMAGE : L'image utilisée (ici wordpress:latest et mysql:5.7).

COMMAND : La commande exécutée au démarrage.

CREATED : Le temps écoulé depuis la création du conteneur.

STATUS : L'état du conteneur (Up 10 minutes signifie qu'il tourne bien).

PORTS : Les ports utilisés (WordPress est accessible sur localhost :8080).

NAMES : Les noms automatiques des conteneurs.

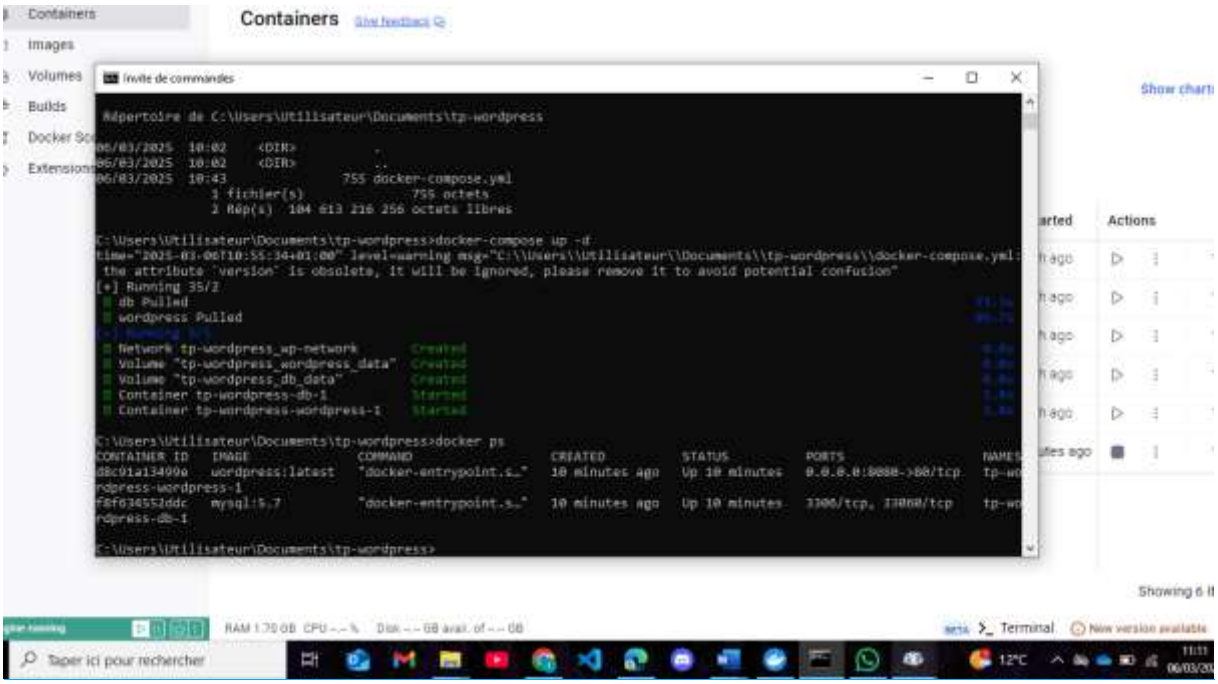
Résultat obtenu :

Le conteneur WordPress (tp-wordpress-wordpress-1) est bien démarré et écoute sur le port 8080.

Le conteneur MySQL (tp-wordpress-db-1) est aussi en cours d'exécution.

=>Tout fonctionne comme prévu

Figure 6:Une capture du résultat

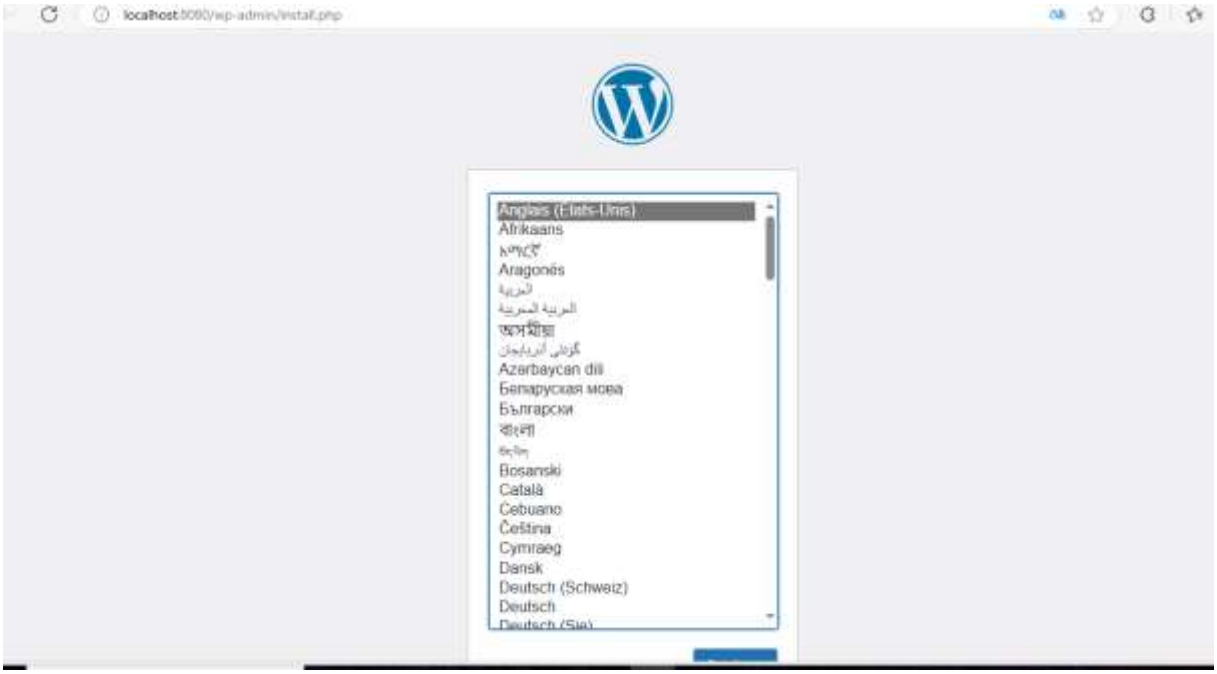


Je peux maintenant accéder à WordPress via mon navigateur à l'adresse :

http://localhost:8080

On demande de choisir une langue

Figure 7:Image de l'interface



7 Installation et configuration de WordPress

Après avoir vérifié que mes conteneurs fonctionnaient correctement, je suis passé à l'installation de WordPress.

Étapes suivies :

1. Accès à l'interface d'installation

J'ai ouvert mon navigateur et entré l'adresse suivante :

http://localhost:8080

ça m'a redirigé vers l'écran d'installation de WordPress.

2. Choix de la langue

J'ai sélectionné Français (Belgique) comme langue d'installation.

3. Configuration du site

Titre du site : Site Web Test

Identifiant : Alcée

Mot de passe : (j'ai choisi un mot de passe sécurisé)

Adresse mail : (j'ai renseigné une adresse mail pour la récupération)

J'ai décoché l'option de visibilité sur les moteurs de recherche, car c'est un test en local.

4. Installation de WordPress

J'ai cliqué sur Installer WordPress, et après quelques secondes, l'installation a été complétée.

J'ai ensuite été redirigé vers la page de connexion où j'ai entré mon identifiant (Alcée) et mon mot de passe pour accéder au tableau de bord de WordPress.

Figure 8:Image pour mettre les informations

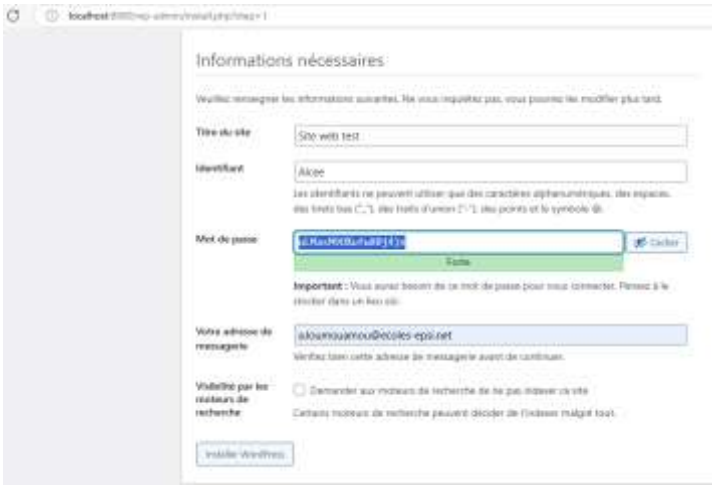


Figure 9:Image de l'interface de connexion

8 Présentation de l'interface du tableau de bord WordPress

Une fois connecté avec mon identifiant (Alcée) et mon mot de passe, j'ai accédé au tableau de bord WordPress.

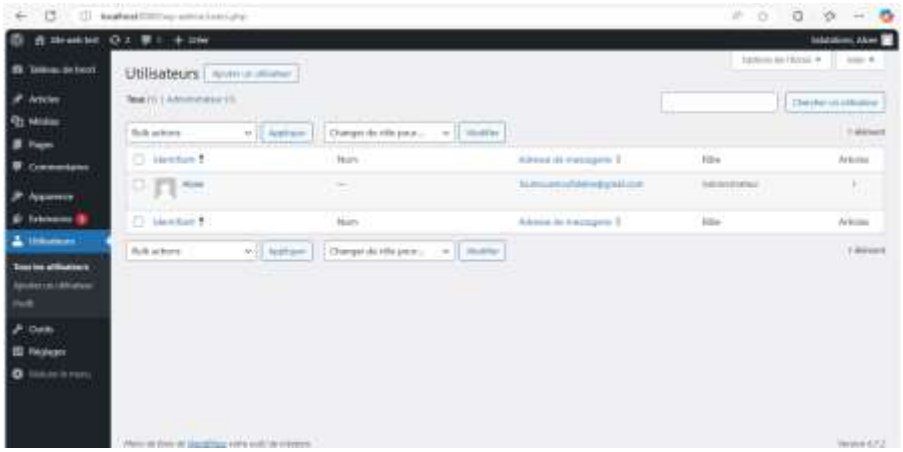
Qu'est-ce que le tableau de bord WordPress ?

C'est l'interface principale qui permet de gérer et personnaliser mon site. Il est composé de plusieurs sections importantes :

- ✓ **Accueil :** Un aperçu rapide du site avec des raccourcis vers les actions principales (écrire un article, gérer les mises à jour, .).
- ✓ **Articles :** Permet de créer, modifier et gérer les articles du blog.
- ✓ **Médias :** Sert à ajouter et gérer les images, vidéos et autres fichiers utilisés sur le site.
- ✓ **Pages :** Section pour créer et éditer des pages statiques comme : **Accueil, À propos, Services, Contact,**
- ✓ **Commentaires :** Affiche les commentaires laissés par les visiteurs, avec la possibilité de les approuver, répondre ou les supprimer.
- ✓ **Apparence :** Permet de modifier l'apparence du site en choisissant un thème, en personnalisant le menu et les widgets.
- ✓ **7. Extensions (Plugins) :** Ajouter des fonctionnalités supplémentaires comme un formulaire de contact, une optimisation SEO, ou encore une boutique en ligne.
- ✓ **Utilisateurs :** Gère les comptes des administrateurs, éditeurs, auteurs et abonnés du site.
- ✓ **Réglages :**Contient les paramètres généraux du site (titre, langue, format de date,).

Figure 10:Capture du tableau de bord wordpress

Figure 11:Image de l'interface utilisateur



9 L’ajout d’un thème WordPress

Pour personnaliser l’apparence de mon site, j’ai ajouté un nouveau thème. Voici les étapes que j’ai suivies :

1. Accéder au tableau de bord : Je me suis connecté à l’interface d’administration via http://localhost:8080/wp-admin.

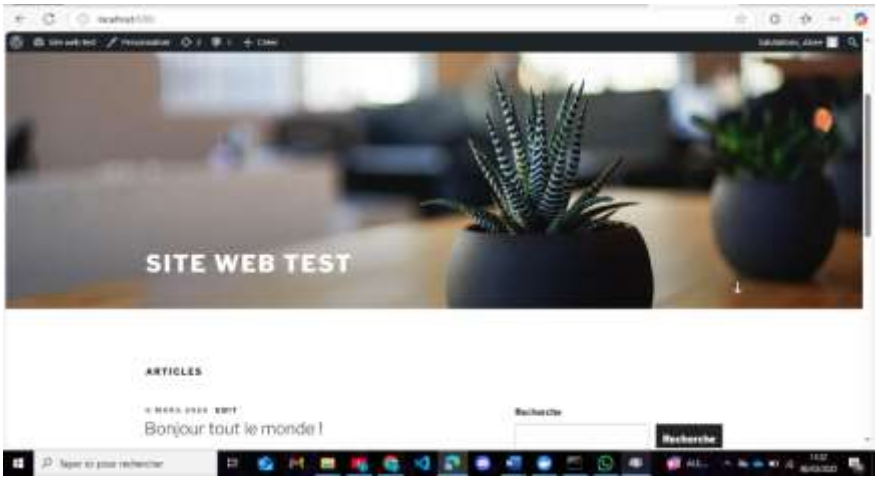
2. Aller dans la section **Apparence** : Dans le menu latéral, j’ai cliqué sur **Apparence > Thèmes**.

3. Ajouter **un nouveau thème** : J’ai sélectionné "Ajouter un thème", puis parcouru la bibliothèque de thèmes proposés par WordPress.

4. Installation du thème : J’ai cliqué sur **Installer** sur le thème choisi, puis sur **Activer** pour l’appliquer à mon site.

Après cette modification, mon site a changé d’apparence selon le design du thème sélectionné. Cela m’a permis d’obtenir un rendu plus professionnel et adapté à mon contenu.

Figure 12:L’image du thème ajouter



10Création d'un menu de navigation

Pour améliorer la navigation de mon site WordPress, j'ai créé un menu principal contenant les éléments : **Accueil, À propos, Services et Contact**. Voici comment j'ai procédé :

Création des pages nécessaires :

Dans le menu latéral, j'ai cliqué sur Pages, puis sur Ajouter.

J'ai créé une page pour chaque section : **Accueil, À propos, Services et Contact.**

Accès à la section des menus :

Dans le menu latéral, j'ai sélectionné Apparence puis Menus

Création d'un nouveau menu :

J'ai saisi Menu Principal comme nom du menu.

J'ai cliqué sur Créer le menu

Ajouter des pages au menu :

Dans la section **Ajouter des éléments au menu**, j'ai coché les cases des pages **Accueil, À propos, Services et Contact.**

J'ai cliqué sur **Ajouter au menu**.

Organisation des éléments du menu :

J'ai réorganisé les éléments en les faisant glisser pour obtenir l'ordre souhaité.

Définition de l'emplacement du menu :

Dans la section **Emplacements du menu**, j'ai coché **Menu principal** pour que le menu apparaisse en haut de mon site.

Enregistrement du menu :

J'ai cliqué sur **Enregistrer le menu** pour appliquer les modifications

Figure 13:Image du site avec les menus ajoutés

11 Modification des réglages de lecture

Pour améliorer l'apparence de mon site WordPress et supprimer le message par défaut affiché sur la page d'accueil, j'ai modifié les réglages de lecture et défini une page statique comme page d'accueil.

Étapes suivies :

1. Accès aux réglages de lecture

Je me suis rendu dans le tableau de bord WordPress.

J'ai cliqué sur **Réglages**, puis sur **Lecture**.

2. Définition d'une page statique comme page d'accueil

Dans la section **Votre page d'accueil affiche**, j'ai sélectionné **Une page statique** au lieu des articles récents.

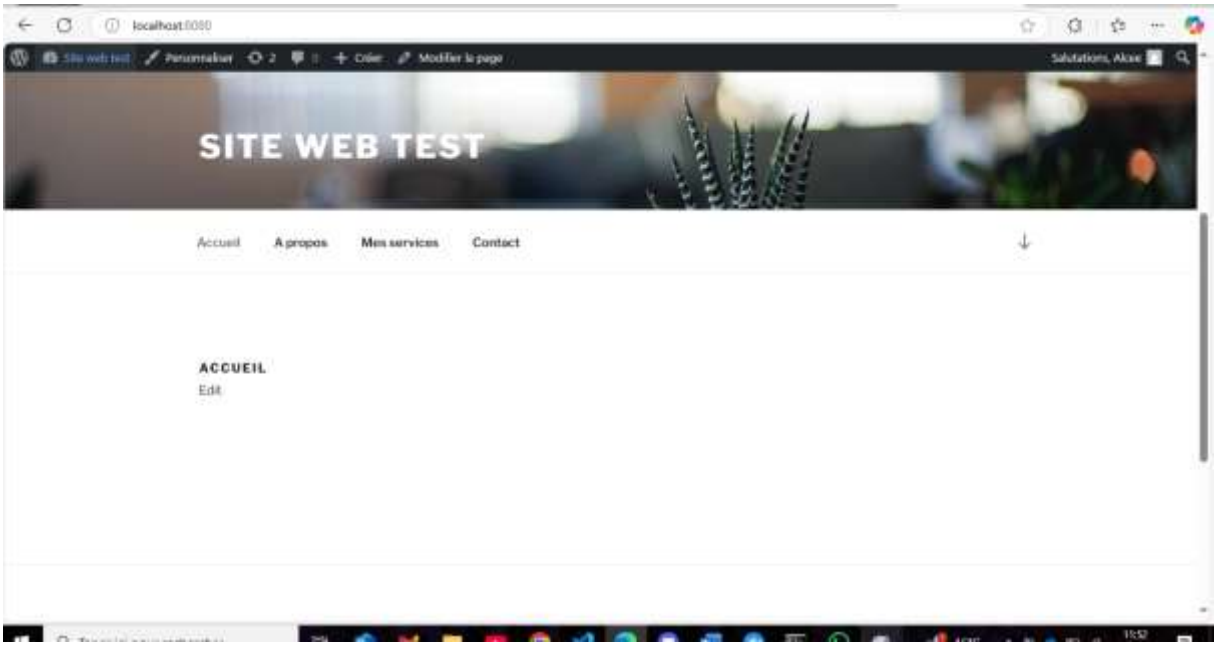
J'ai choisi **Accueil** dans le menu déroulant pour que cette page soit la page d'accueil du site.

3. Enregistrement des modifications

J'ai cliqué sur **Enregistrer les modifications** en bas de la page.

N.B : Grâce à ces réglages, mon site affiche désormais une page d'accueil personnalisée sans le message par défaut de WordPress, rendant l'expérience utilisateur plus propre et professionnelle.

Figure 14:Image de la page d'accueil du site



12 Modification de la page d’accueil ajout d’ une image, un titre et d’un bouton

Pour rendre mon site plus attractif et interactif, j’ai ajouté une image, un titre et un bouton d’action sur ma page d’accueil.

Étapes suivies :

1. Ajouter une imager

Je suis allé sur la page d’accueil en mode édition.

J’ai cliqué sur **Ajouter un bloc**, puis sélectionné **Image**.

J’ai téléchargé une image depuis mon ordinateur et l’ai insérée dans la page.

2. Ajout du titre

J’ai inséré un bloc **Titre** pour structurer mon contenu.

J’ai rédigé un titre principal accrocheur pour capter l’attention des visiteurs.

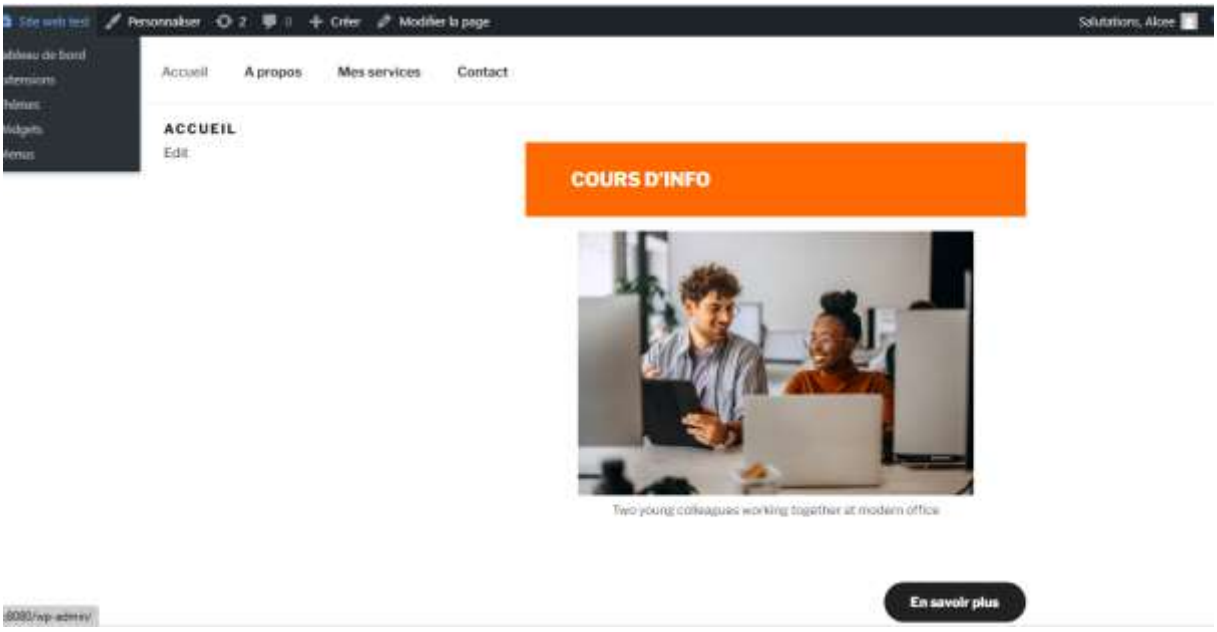
3. Ajout d’un bouton d’action

J’ai ajouté un bloc **Bouton** et personnalisé son texte,en écrivant **En savoir plus**

4. Enregistrement des modifications

Une fois satisfait du design, j’ai cliqué sur **Mettre à jour** pour enregistrer les changements.

Figure 15: Image de la modification de la page d'accueil



13 Connexion à l'administration WordPress via le navigateur

Pour accéder à l'interface d'administration de mon site WordPress, j'utilise mon navigateur web en entrant l'URL suivante :

http://localhost:8080/wp-admin

Étapes de connexion :

1. J'ouvre un autre navigateur pour me connecter Google

2. Dans la barre d'adresse, je tape localhost :8080/wp-admin et j'appuie sur Entrée.

3. La page de connexion WordPress apparaît.

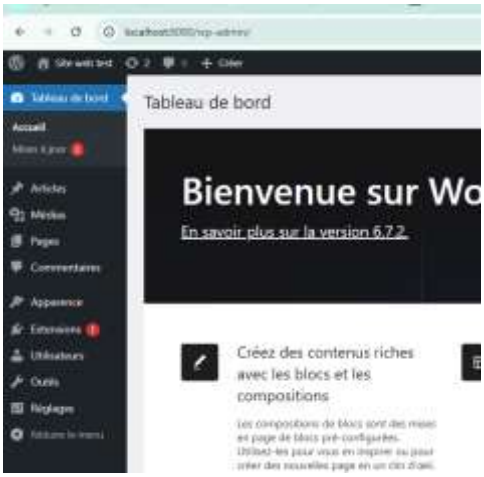
4. Je saisis :

Mon identifiant (Alcée)

Mon mot de passe (défini lors de l'installation)

5. Je clique sur **Se connecter.**

Figure 16:Image du tableau de bord après connexion



Dans mon terminal, j'ai tapé la commande :

docker ps

Cette commande me permet de voir les conteneurs en cours d'exécution. Voici ce que j'obtiens :

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d8c91a13499e wordpress:latest "docker-entrypoint.s..." 2 hours ago Up 2 hours 0.0.0.0:8080->80/tcp tp-wordpress-wordpress-1

f8f634552ddc mysql:5.7 "docker-entrypoint.s..." 2 hours ago Up 2 hours 3306/tcp, 33060/tcp tp-wordpress-db-1

L'explication de chaque colonne :

CONTAINER ID : Identifiant unique du conteneur.

IMAGE : L'image utilisée pour le conteneur (ici, wordpress:latest et mysql:5.7).

COMMAND : La commande exécutée au démarrage du conteneur.

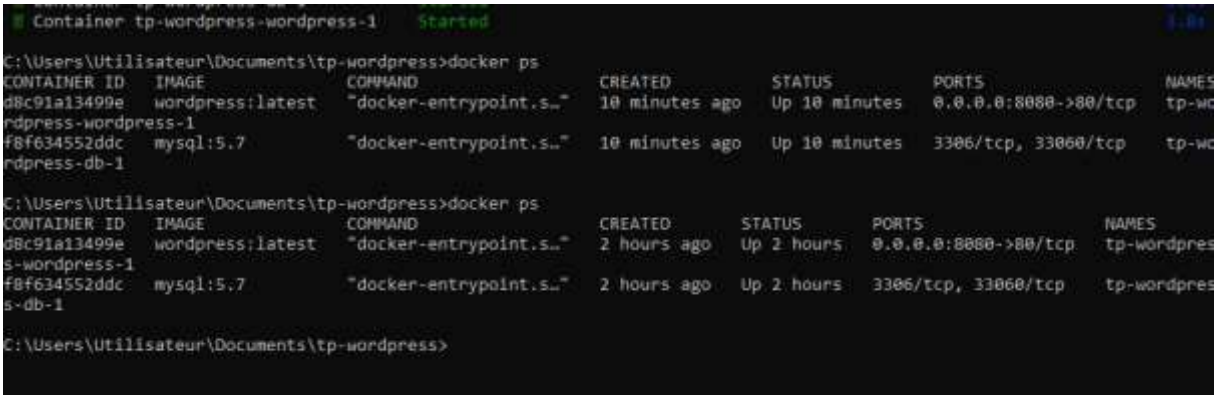
CREATED : Le temps écoulé depuis la création du conteneur.

STATUS : Indique si le conteneur est actif (ici, **Up 2 hours** signifie qu'il tourne depuis 2 heures).

PORTS : Montre les ports utilisés (WordPress est accessible sur localhost:8080).

NAMES : Le nom donné automatiquement aux conteneurs.

Figure 17:Résultats de la commande Docker ps avec cet image



14 Explication des choix de configuration

1. **Ports :**

J'ai défini le port 8080 pour WordPress :

ports :

- 8080 :80

Pourquoi ? Cela permet d'accéder à WordPress via localhost :8080 dans le navigateur.

Avantage : Évite les conflits avec d'autres services qui pourraient déjà utiliser le port 80.

2. Volumes

J'ai créé deux volumes pour garantir la persistance des données :

volumes:

wordpress:

db:

Volume WordPress (wordpress) : Il stocke les fichiers du site (thèmes, plugins, images...).

Volume MySQL (db) : Il sauvegarde la base de données, évitant la perte des données si le conteneur MySQL redémarre.

3. Variables d'environnement

Elles permettent de configurer WordPress et MySQL sans modifier directement les fichiers.

WordPress :

environnement :

WORDPRESS_DB_HOST: db

WORDPRESS_DB_USER: exampleuser

WORDPRESS_DB_PASSWORD: examplepass

WORDPRESS_DB_NAME: exampledb

WORDPRESS_DB_HOST: db => ça indique à WordPress où trouver la base de données.

WORDPRESS_DB_USER, WORDPRESS_DB_PASSWORD, WORDPRESS_DB_NAME → Identifiants de connexion à MySQL.

MySQL :

environnement :

MYSQL_DATABASE: exampledb

MYSQL_USER: exampleuser

MYSQL_PASSWORD: examplepass

MYSQL_RANDOM_ROOT_PASSWORD: '1'

MYSQL_DATABASE = > permet de créer automatiquement la base de données exampledb.

MYSQL_USER et MYSQL_PASSWORD => va définir un utilisateur sécurisé pour la base.

MYSQL_RANDOM_ROOT_PASSWORD: '1' =>va Générer un mot de passe root aléatoire (pour la sécurité).

4. Réseau

J'ai défini un réseau personnalisé wp-network pour isoler et organiser les communications entre les conteneurs :

networks :

wp-network :

Chaque service est connecté à ce réseau :

networks :

wp-network:

driver: bridge

Pourquoi ?

- ✓ Pour une meilleure isolation du projet.
- ✓ Facilite la communication entre WordPress et MySQL sans utiliser d'adresses IP.
- ✓ Améliore la sécurité et l'organisation des conteneurs.

Pour être plus clair en gros c'est :

Port 8080 : Accéder à WordPress sans conflit avec d'autres services.

Volumes : Sauvegarder les fichiers WordPress et la base MySQL.

Variables d'environnement : une configuration dynamique et sécurisée.

Réseau wp-network : Organisation et communication efficace entre les conteneurs.

15 Les difficultés rencontrées et solutions

Dans l'ensemble, l'installation et la configuration de WordPress avec Docker se sont bien déroulées. Cependant, j'ai rencontré une difficulté au moment de créer mon dossier de travail.

Problème : Impossible de créer le dossier tp-wordpress

Au début, j'ai utilisé la commande suivante pour me déplacer dans Documents et créer mon dossier :

cd Documents

mkdir tp-wordpress

Mais le dossier ne s'est pas créé ,

Solutions apportées :

Il fallait vérifier l'emplacement. J'ai utilisé la commande suivante pour voir les dossiers présents dans **Documents** et vérifier si j'étais bien au bon endroit : **dir**

Création du dossier

Une fois certain d'être dans Documents, j'ai relancé la commande :

mkdir tp-wordpress

Ensuite, j'ai pu entrer dans le dossier avec :

cd tp-wordpress

Après cette correction, j'ai pu continuer le TP normalement et tout a fonctionné sans problème.

16 Conclusion :

Ce projet m'a permis de mettre en pratique l'installation et la configuration d'un environnement WordPress en local à l'aide de Docker et Docker Compose. J'ai appris à créer et gérer des conteneurs pour WordPress et MySQL, en assurant la persistance des données grâce aux volumes.

J'ai également pu personnaliser le site en ajoutant un thème, un menu, des pages et des images, ainsi qu'en modifiant certains paramètres comme l'affichage de la page d'accueil. L'accès à l'administration via localhost :8080/wp-admin a confirmé que le site était fonctionnel.

Ce TP m'a permis de mieux comprendre le fonctionnement des conteneurs, la gestion des réseaux et la persistance des données. Pour aller plus loin, il serait intéressant d'ajouter des services comme phpMyAdmin pour gérer la base de données graphiquement.

En somme cette expérience a renforcé mes compétences en déploiement et gestion de conteneurs avec Docker, un outil essentiel pour l'administration, la coordination et la gestion d'applications web.