Docker : Maîtrisez la conteneurisation applicative de A à Z

Table des matières

Réalisation : Guillaume DELACROIX Formateur AFPA

Docker pour une application Back End

Installation et Utilisation du Conteneur PHP et Apache pour héberger une application PHP

Créer un dossier pour votre projet

Créez un dossier dédié pour votre projet PHP.

Par exemple, créez un dossier appelé **'my-php-app'**.

Faites vos développements jusqu'à ce que tout fonctionne

Créer un fichier Dockerfile

À l'intérieur du dossier my-php-app, créez un fichier appelé Dockerfile sans extension. Vous pouvez utiliser un éditeur de texte pour cela. Ajoutez le contenu suivant dans le fichier Dockerfile :

```
# Utilisez une image PHP Apache
FROM php:apache

# Copiez le contenu de l'application PHP dans le répertoire de travail du conteneu
r (/var/www/html/ est le répertoire par défaut d'Apache)
COPY . /var/www/html/

# Installez les dépendances de PDO pour MySQL
RUN docker-php-ext-install pdo pdo_mysql
```

Construire l'image Docker

Ouvrez un terminal et naviguez jusqu'au dossier **my-php-app** contenant le Dockerfile et votre application. Exécutez la commande suivante pour construire l'image Docker :

```
docker build -t my-php-app .
```

Cela va construire une image Docker nommée **my-php-app**.

Exécution du conteneur

Une fois l'image construite, vous pouvez exécuter un conteneur basé sur cette image en utilisant la commande suivante :

```
docker run -d -p 80:80 my-php-app
```

Cela lancera un conteneur basé sur l'image **my-php-app** en arrière-plan, en mappant le port 80 du conteneur au port 80 de votre système hôte.

Accédez à votre application

Ouvrez votre navigateur web et accédez à **http://localhost**. Vous devriez voir votre application PHP en cours d'exécution dans le conteneur Docker.

Arrêtez et supprimer le conteneur

Pour arrêter le conteneur, récupérez son ID en utilisant la commande docker ps et ensuite utilisez la commande docker stop avec l'ID du conteneur.

Par exemple :

```
docker ps
docker stop CONTAINER_ID
```

Pour supprimer le conteneur, utilisez la commande **docker rm** avec l'ID du conteneur :

```
docker rm CONTAINER_ID
```

Installation et Utilisation du Conteneur MariaDB

MariaDB est une base de données SQL open-source, largement utilisée dans le développement web. Dans ce chapitre, nous verrons comment installer et utiliser MariaDB à l'aide de Docker.

Télécharger l'image MariaDB

La première chose à faire est de télécharger l'image MariaDB depuis Docker Hub. Pour cela, vous pouvez utiliser la commande `docker pull`.

Par exemple :

```
docker pull mariadb
```

Cette commande télécharge la dernière version de MariaDB disponible sur Docker Hub.

Créer un conteneur MariaDB

Pour créer un conteneur MariaDB, vous pouvez utiliser la commande `docker run`. Vous devrez spécifier certains paramètres d'environnement, tels que le mot de passe root, le nom de la base de données à créer, etc.

Par exemple :

```
docker run -d -p 3306:3306 --name my-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -
e MYSQL_DATABASE=mydb -v mariadb_data:/var/lib/mysql mariadb
```

Dans cette commande :

- `-d` signifie que le conteneur doit être exécuté en arrière-plan (mode détaché).

- `-p 3306:3306` signifie que le port 3306 du conteneur doit être exposé sur le port 3306 de l'hôte.

- `--name my-mariadb` donne un nom au conteneur.

- `-e MYSQL_ROOT_PASSWORD=my-secret-pw` définit le mot de passe root de la base de données.

- `-e MYSQL_DATABASE=mydb` crée une nouvelle base de données appelée "mydb".

- `-v mariadb_data:/var/lib/mysql` crée un volume pour stocker les données de la base de données de manière persistante.

Utiliser la base de données MariaDB

Pour interagir avec la base de données MariaDB, vous pouvez utiliser la commande `docker exec` pour exécuter la commande `mysql`.

Par exemple :

```
 docker exec -it my-mariadb mysql -u root -p
```

Cette commande vous invite à entrer le mot de passe root que vous avez défini lors de la création du conteneur, puis ouvre l'interface de ligne de commande MySQL.

Dans le prochain chapitre, nous verrons comment installer et utiliser phpMyAdmin pour gérer votre base de données MariaDB de manière plus conviviale.

Installation et Utilisation du Conteneur PhpMyAdmin

phpMyAdmin est un outil de gestion de base de données MySQL/MariaDB basé sur le web. Il est très populaire pour sa facilité d'utilisation. Voici comment l'installer et l'utiliser avec Docker.

Télécharger l'image phpMyAdmin

Tout d'abord, vous devez télécharger l'image phpMyAdmin depuis Docker Hub. Pour cela, utilisez la commande `docker pull`.

Par exemple :

```
docker pull phpmyadmin
```

Cette commande télécharge la dernière version de phpMyAdmin disponible sur Docker Hub.

Créer un conteneur phpMyAdmin

Ensuite, vous pouvez créer un conteneur phpMyAdmin en utilisant la commande `docker run`. Vous aurez besoin de spécifier quelques variables d'environnement, notamment les détails de connexion à votre base de données MariaDB.

Par exemple :

```
docker run -d -p 8080:80 --name my-phpmyadmin -e PMA_HOST=mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw phpmyadmin
```

Dans cette commande :

- `-d` signifie que le conteneur doit être exécuté en arrière-plan (mode détaché).

- `-p 8080:80` signifie que le port 80 du conteneur (le port par défaut pour le serveur web) doit être exposé sur le port 8080 de l'hôte.

- `--name my-phpmyadmin` donne un nom au conteneur.

- `-e PMA_HOST=mariadb` définit l'hôte de la base de données que phpMyAdmin doit gérer. Cela devrait correspondre au nom du conteneur MariaDB que vous avez créé précédemment.

- `-e MYSQL_ROOT_PASSWORD=my-secret-pw` définit le mot de passe root de la base de données.

Accéder à phpMyAdmin

Une fois le conteneur phpMyAdmin en cours d'exécution, vous pouvez accéder à l'interface phpMyAdmin en ouvrant votre navigateur web et en naviguant vers `http://localhost:8080`.

Vous devriez être en mesure de vous connecter en utilisant le nom d'utilisateur "root" et le mot de passe que vous avez défini lors de la création du conteneur MariaDB.

Comment faire interagir ensemble tous ces conteneurs ?

Nous avons déjà configuré le Dockerfile pour notre application PHP.

Créez le script de création de la base de données MySQL

Maintenant, créez un script SQL (par exemple, **'createdb.sql'**) contenant les commandes pour créer votre base de données et vos tables, ainsi que pour les alimenter en données.

Exemple :

```
CREATE DATABASE mydatabase;
USE mydatabase;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Placez ce fichier (**createdb.sql**) dans le même dossier que votre **Dockerfile**.

Créez le fichier docker-compose.yml

Maintenant, nous allons créer un fichier **docker-compose.yml** pour définir le service PHP et le service MySQL.

Créez un fichier appelé **docker-compose.yml** à la racine de votre projet et ajoutez le contenu suivant :

```
version: '3.8'

services:
  app:
    build:
      context: ./app
      dockerfile: Dockerfile  # Spécifiez ici le chemin vers votre Dockerfile
personnalisé
    ports:
      - "80:80"
    volumes:
      - ./app:/var/www/html
    depends_on:
      - db

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: app_db
      MYSQL_USER: app_user
      MYSQL_PASSWORD: app_password
```

```
    volumes:
      - db_data:/var/lib/mysql
      - ./createdb.sql:/docker-entrypoint-initdb.d/createdb.sql  # Montez le
script SQL

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
      - "8080:80"
    environment:
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: root_password

volumes:
  db_data:
```

Assurez-vous de remplacer les valeurs des variables d'environnement (comme les mots de passe) par celles que vous souhaitez utiliser. Ce fichier Docker Compose crée trois services :

1 - **app**: Un conteneur PHP avec Apache où vous pouvez placer votre code PHP dans le dossier ./app et qui s'appuie sur le Dockerfile créé précédemment.

2 - **db**: Un conteneur MySQL pour la base de données. Les variables d'environnement définies permettent de configurer le mot de passe root, le nom de la base de données et les informations d'authentification pour l'utilisateur de l'application.

3 - **phpmyadmin**: Un conteneur PhpMyAdmin pour gérer facilement la base de données. Il est lié au conteneur MySQL (db) et utilise les mêmes informations d'authentification.

N'oubliez pas de remplacer **your_root_password** par le mot de passe root souhaité pour votre base de données.

Assurez-vous que le Dockerfile personnalisé pour l'application PHP se trouve dans un sous-dossier nommé "app" à côté de votre fichier Docker Compose, ou mettez à jour le chemin dans la section `build` pour correspondre à l'emplacement réel de votre Dockerfile personnalisé. Cette configuration permettra à Docker Compose de construire l'image de l'application PHP à partir du Dockerfile spécifié lors de l'exécution de `docker-compose up`.

Exécutez l'application dans des conteneurs Docker

Ouvrez une invite de commande dans le répertoire de votre projet et exécutez la commande suivante pour construire les images Docker et démarrer les conteneurs :

```
docker-compose up -d
```

Cela va créer les images Docker pour l'application PHP et MySQL, lancer les conteneurs et lier le port 80 de votre machine hôte au port 80 du conteneur pour accéder à l'application.

Votre application PHP devrait maintenant être accessible à l'adresse **http://localhost** dans votre navigateur.

Le script **createdb.sql** sera exécuté automatiquement lors du lancement du conteneur MySQL, créant ainsi la base de données et les tables nécessaires.

PHPMyAdmin sera disponible sur le port 8080 à l'adresse **http://localhost:8080.**

C'est tout! Vous avez maintenant une application PHP connectée à une base de données MySQL, le tout hébergé dans des conteneurs Docker.

Attention, cela dit…. L'hôte de votre BDD est db dans votre conteneur.
En dev, on a tendance à utiliser une base locale, donc on a comme hôte dans notre code php *localhost*.
Avant de monter votre conteneur, il faudra modifier en *db* partout dans votre code.

Le mieux étant d'avoir trois fichiers d'environnements et d'initialiser l'info $host à partir de cequ'il y'a dans le fichier .env.
Un fichier *.env.dev* qui contient les infos d'environnement en dev
Un fichier *.env* qui contient les infos d'environnement dans le « stack » actuel (en dev, ce sera une copie du fichier précédent
Un fichier *.env.prod* qui contient les infos d'environnement du conteneur.

On rajoutera dans notre docker.ignore ceci.

.dockerignore | dockerfile

app > .dockerignore

```
# .dockerignore

# Exclude the 'scripts' dire
scripts/
# Exclude the env files
.dev.env
.env
.prod.env


```

Et dans notre dockerfile :

.dockerignore | dockerfile

app > dockerfile > ...

```
#utilisons une image PHP apache
FROM php:apache
#copy de toute l'appl dans le répertoire du travail du conteneur
COPY . /var/www/html
#copy de l'env prod en le renommant en env
COPY .env.prod /var/www/html/.env
# installons les dépendances de PDO pour MySQL
RUN docker-php-ext-install pdo pdo_mysql
```