

## Explications sur l'image fournie

Docker est une plateforme logicielle qui permet de créer, déployer et exécuter des applications de manière isolée, appelée *conteneurisation*. Les conteneurs sont des environnements légers, portables et reproductibles qui contiennent tout le nécessaire pour faire fonctionner une application : le code, les dépendances, les bibliothèques et le système d'exploitation minimal requis. Docker est déjà installé sur les ordinateurs de l'ISTIC.

Fichier `docker-compose.yml`

```
version: '3.8'

services:
  app:
    build: . # Utilise le Dockerfile dans le même répertoire
    container_name: symfony_app
    working_dir: /var/www
    volumes:
      - ./var/www
    depends_on:
      - db

  webserver:
    image: nginx:alpine
    container_name: symfony_web
    ports:
      - "8000:80"
    volumes:
      - ./var/www
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - app

  db:
    image: mysql:latest
    container_name: symfony_db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: symfony_db
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    volumes:
      - db_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin:latest
    restart: always
    ports:
      - "8081:80" # Accède à phpMyAdmin via http://localhost:8081
```

Petit rappel : *Il faut d'abord faire CTRL+D pour sortir du shell ...ou CTRL+C pour arrêter le serveur qui tourne dans le terminal...*

```
environment:
  PMA_HOST: db # Indique le nom du service de base de données
  PMA_USER: user # Utilisateur MySQL
  PMA_PASSWORD: password # Mot de passe MySQL

volumes:
  db_data:
```

Le fichier docker-compose.yml définit quatre services : app, webserver, db et phpmyadmin. Chacun de ces services correspond à un conteneur Docker qui contient différentes applications ou composants nécessaires au bon fonctionnement de ton projet Symfony.

Les conteneurs contiennent les éléments suivants :

### Conteneur app (PHP)

- **Image** : `php:8.1-fpm`
  - Ce conteneur utilise l'image officielle de PHP version 8.1 qui est adapté pour exécuter des applications PHP.
- **Working Directory** : `/var/www`
  - Ce répertoire est le dossier de travail à l'intérieur du conteneur où l'application Symfony sera montée.
- **Volumes** :
  - `./var/www` : Le répertoire actuel (où se trouve ton `docker-compose.yml`) est monté à l'intérieur du conteneur. Cela signifie que tout fichier que tu modifies dans ce répertoire local sera accessible et modifiable à l'intérieur du conteneur.

### Conteneur webserver (Nginx)

- **Image** : `nginx:alpine`
  - Ce conteneur utilise une image légère d'Nginx basée sur Alpine Linux.
- **Ports** :
  - `8000:80` : Cela signifie que le port 80 du conteneur (où Nginx écoute) est mappé sur le port 8000 de ta machine hôte. Tu pourras accéder à ton application en ouvrant `http://localhost:8000`.
- **Volumes** :
  - `./var/www` : Comme pour le conteneur app, cela monte le répertoire actuel dans le conteneur.
  - `./nginx.conf:/etc/nginx/conf.d/default.conf` : Cela remplace la configuration par défaut d'Nginx avec ton fichier de configuration personnalisé, ce qui permet de spécifier comment Nginx doit servir ton application Symfony.

### Conteneur db (MySQL)

- **Image** : `mysql:latest`
  - Ce conteneur utilise la dernière image officielle de MySQL.

Petit rappel : *Il faut d'abord faire CTRL+D pour sortir du shell ...ou CTRL+C pour arrêter le serveur qui tourne dans le terminal...*

- **Environnement :**

- `MYSQL_ROOT_PASSWORD: root` : Définit le mot de passe de l'utilisateur `root` de MySQL.
- `MYSQL_DATABASE: symfony_db` : Crée une base de données nommée `symfony_db` au démarrage.
- `MYSQL_USER: user` et `MYSQL_PASSWORD: password` : Crée un utilisateur MySQL avec le nom d'utilisateur `user` et le mot de passe `password`.

- **Volumes :**

- `db_data:/var/lib/mysql` : Cela crée un volume persistant pour stocker les données de la base de données, ce qui permet de conserver les données même si le conteneur est arrêté ou supprimé.

Et de même, il existe un conteneur pour phpmyadmin...

## En résumé

- **Conteneur** `app` : Exécute PHP pour gérer la logique de l'application Symfony.
- **Conteneur** `webserver` : Gère les requêtes HTTP avec Nginx et sert les fichiers de l'application.
- **Conteneur** `db` : Gère la base de données MySQL utilisée par l'application Symfony.
- **Conteneur** `phpmyadmin` : Gère PhpMyAdmin utilisé par l'application Symfony.

Tous ces conteneurs interagissent ensemble pour fournir l'environnement nécessaire au développement de ton application Symfony

## Fichier de configuration Nginx

Le fichier `nginx.conf` est un fichier de configuration pour le serveur web Nginx. Il définit comment Nginx doit traiter les requêtes HTTP, comment il doit servir les fichiers de ton application, et comment il interagit avec d'autres services.

```
server {
    listen 80;
    server_name localhost;

    root /var/www/projet_miage/public; # Cela doit pointer vers le répertoire public

    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php$is_args$args; # Redirige vers index.php pour
Symfony
    }

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass app:9000; # Correspond à PHP-FPM
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }

    location ~ /\.ht {
```

Petit rappel : *Il faut d'abord faire CTRL+D pour sortir du shell ...ou CTRL+C pour arrêter le serveur qui tourne dans le terminal...*

```
    deny all;
}}
```

## Fichier Dockerfile

Voila ce que va contenir ce Dockerfile :

```
# Utiliser l'image PHP 8.1-FPM comme base
FROM php:8.1-fpm

# Installer les dépendances nécessaires
RUN apt-get update && apt-get install -y \
    curl \
    git \
    unzip \
    && rm -rf /var/lib/apt/lists/*

# Installer Symfony CLI
RUN curl -sS https://get.symfony.com/cli/installer | bash && \
    mv ~/.symfony*/bin/symfony /usr/local/bin/symfony

# Installer Composer
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --
filename=composer
```

Petit rappel : *Il faut d'abord faire CTRL+D pour sortir du shell ...ou CTRL+C pour arrêter le serveur qui tourne dans le terminal...*