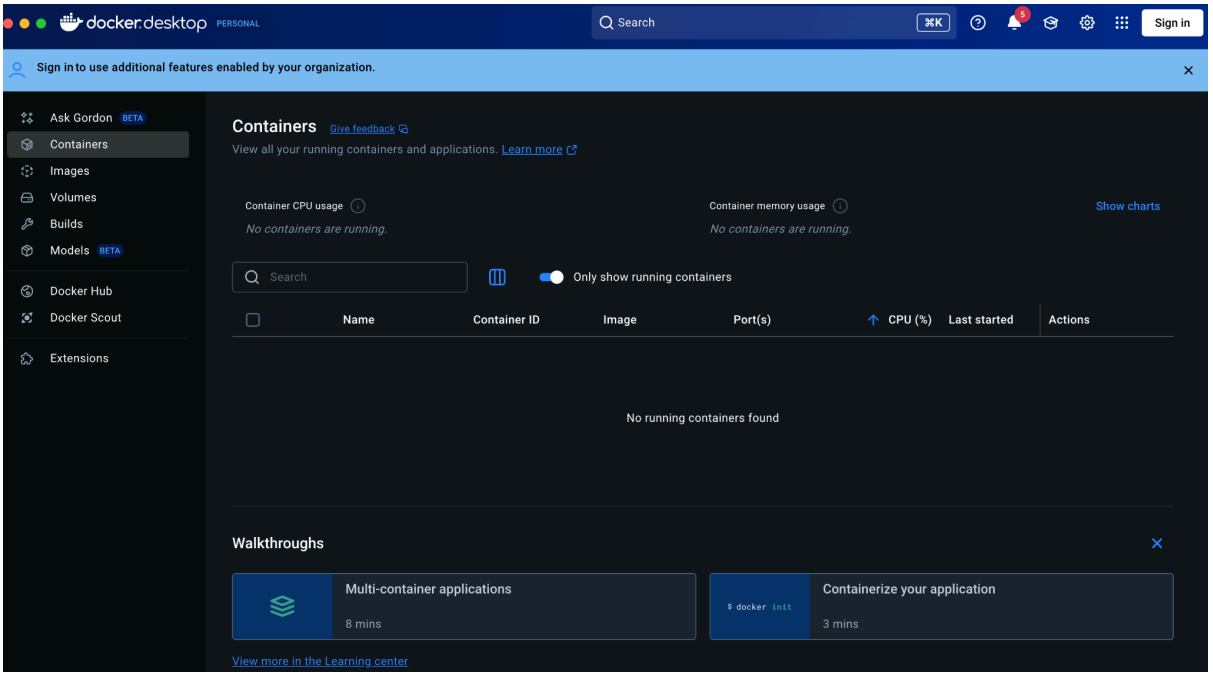
Installation et utilisation de Docker sur MAC.

“Docker permet d'empaqueter une application et ses dépendances dans un conteneur isolé, qui peut être exécuté sur n'importe quel serveur. Il ne s'agit pas de virtualisation, mais de conteneurisation”

1. Installer “Docker Desktop”

Cela permettra de gérer ses conteneurs
https://www.docker.com/products/docker-desktop/

Dès que Docker Desktop est installé, le logiciel doit ressembler à ça :



Mon but est ici de créer un environnement de développement pour un projet Symfony dans le cadre de réalisation de tests de procédure, un environnement d’apprentissage.

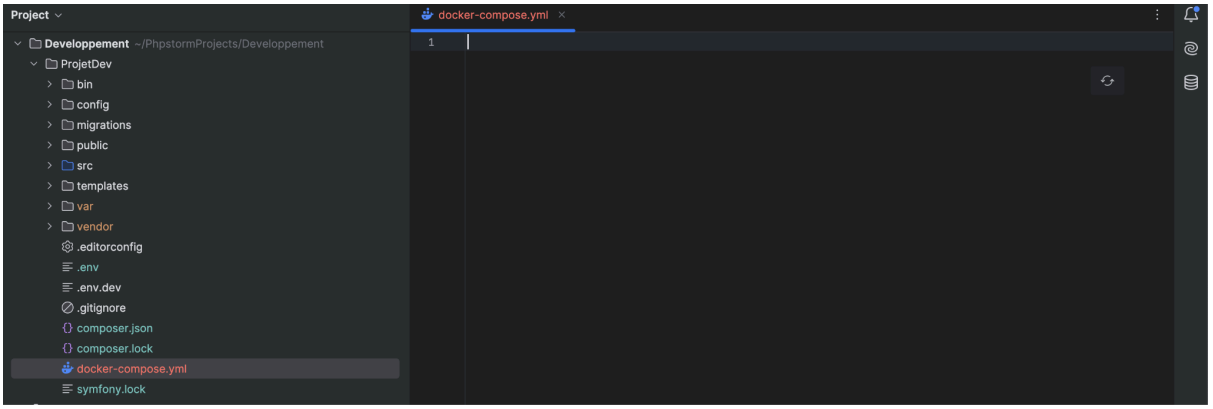
Il est possible de télécharger des “Images” directement depuis le site “docker” ou nous pouvons nous même créer les images.
https://hub.docker.com/search

2. Pour un projet Symfony, nous allons créer nous même les conteneurs avec nos besoins

- Un conteneur “MariaDB” pour gérer les bases de données
- Un conteneur “PHPMyAdmin” pour l’interface graphique
- Un conteneur “Apache”/Docker afin de contenir notre projet

Pour se faire, il faut tout d'abord un projet Symfony avec les composer require de base (Doctrine, Form, Maker, Twig, security, validator, csrf).

Dès que cela est fait, on crée un fichier "docker-compose.yml" à la racine du projet, il contiendra tous les détails de nos conteneurs.



Il faut alors définir les détails de nos services :

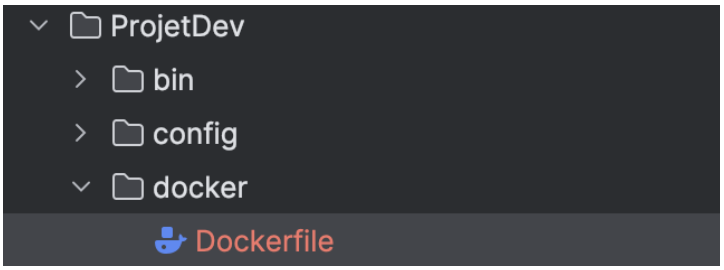
```yaml
version: '3.8'

services:

   projetdev:
       build:
           context: ./docker
       volumes:
           - ./:/var/www/html
       ports:
           - "8083:80"

   projetdev-db:
       image: mariadb
       environment:
           MYSQL_ROOT_PASSWORD: root
           MYSQL_DATABASE: projetdevbdd
       ports:
           - "3336:3306"
       volumes:
           - mariadb_data:/var/lib/mysql

   phpmyadmin:
       image: phpmyadmin
       environment:
           PMA_HOST: projetdev-db
           PMA_PORT: "3306"
       ports:
           - "8890:80"

volumes:
   mariadb_data:
```

Le Context va permettre de définir l'endroit où sera installé le Dockerfile (l'image). Pour une question d'organisation, je crée un dossier "Docker" pour mettre ce fichier afin d'ajouter aussi un fichier vhost.conf.
On décide des ports, et ressources pour chaque conteneur (ici je choisi des ports "non-conventionnels" , *8083 au lieu de 8000,* puisque j'ai déjà plusieurs conteneurs existants, ils ne doivent pas entrer en conflit). On décide par ailleurs, des mots de passe de la base de données (ici root suffit pour un projet en local).

3. Création du Dockerfile

Le dockerfile va gérer les détails de l'image (aussi téléchargeable depuis le hub docker)

Je crée un dossier /docker puis le fichier :



Dans ce fichier :

```
FROM php:8.4-apache
RUN apt-get update && apt-get install -y libicu-dev zip unzip
RUN docker-php-ext-install intl pdo pdo_mysql
COPY vhost.conf /etc/apache2/sites-available/000-default.conf
RUN a2enmod rewrite
RUN sed -i "s/128M/1024M/gi"
/usr/local/etc/php/php.ini-development
RUN mv /usr/local/etc/php/php.ini-development
/usr/local/etc/php/php.ini
```

Ensuite je crée un fichier "vhost.conf" dans ce même dossier afin de préparer le détail du virtualhost

```
<VirtualHost *:80>
    DocumentRoot /var/www/html/public

    <Directory /var/www/html/public>
        AllowOverride All
        Require all granted
        FallbackResource /index.php
    </Directory>
```

```
</VirtualHost>
```

Dès que cela est fait, je peux construire l'image "build" :
docker-compose up --build

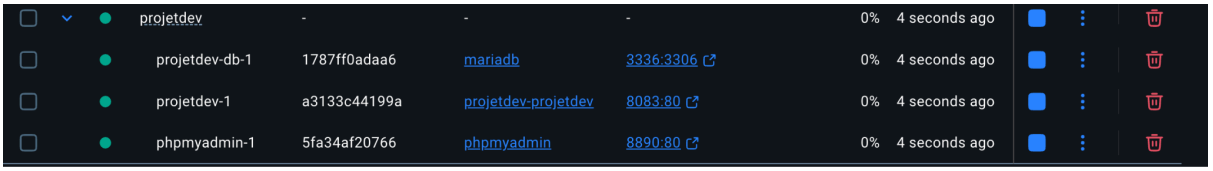
La construction de l'image est lancée

```
[+] Building 68.2s (6/11)
 => => extracting sha256:5c9b3af69d7172d9f2a6f7ab2a9933e475bb3519750408e3b65158e2ee202f17
 => => extracting sha256:a25559cd550c9e4fb3abdc8cb3a193f99a90b5905b7a9082ea7ab0a0c5ec4d9b
 => => extracting sha256:dc14b68ad18b17822fa52d92f24ffb06a0941134712e7d7bce71a3aa383b8359
 => => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1
 => [projetdev internal] load build context
 => => transferring context: 293B
 => [projetdev 2/7] RUN apt-get update && apt-get install -y libicu-dev zip unzip
 => [projetdev 3/7] RUN docker-php-ext-install intl pdo pdo_mysql
```

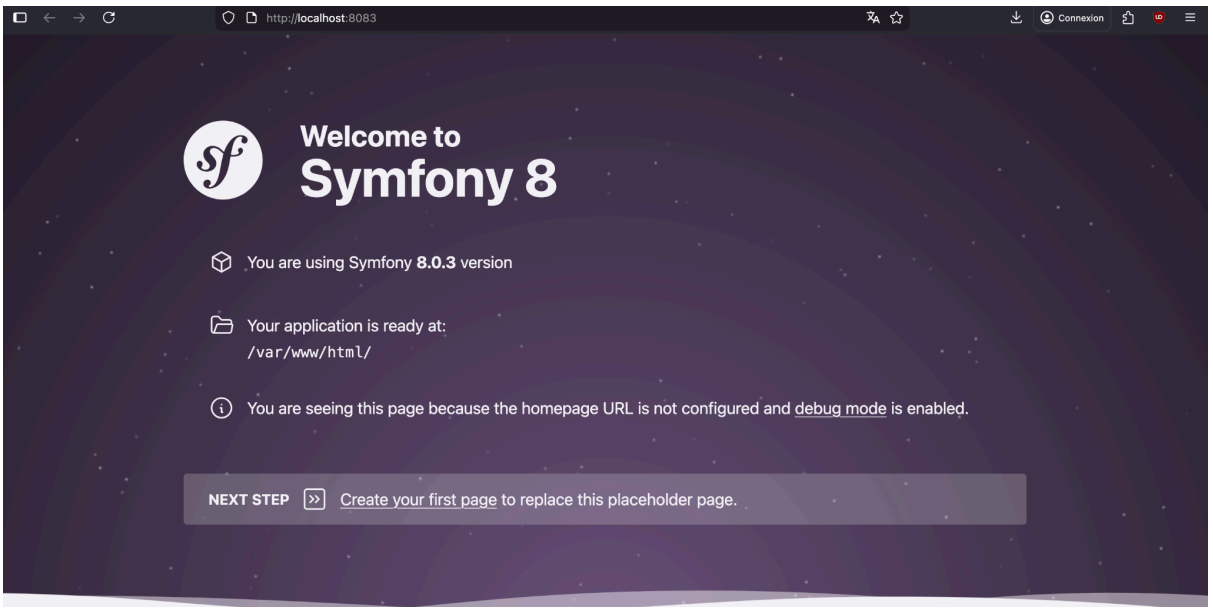
```
[+] Running 6/6
 ✔ projetdev                          Built
 ✔ Network projetdev_default          Created
 ✔ Volume "projetdev_mariadb_data"    Created
 ✔ Container projetdev-gsb-bdd-1      Created
 ✔ Container projetdev-projetdev-1    Created
 ✔ Container projetdev-phpmyadmin-1   Created
```

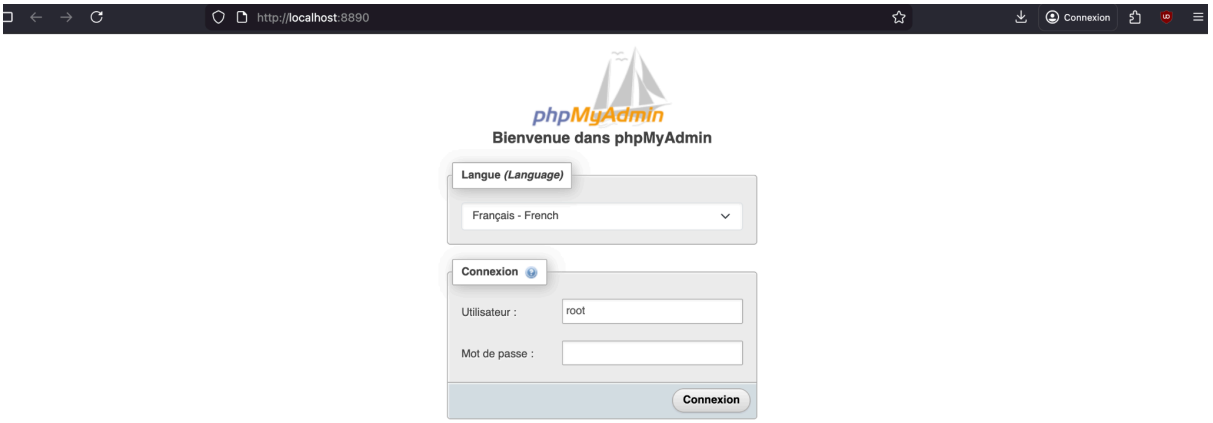
Avec la commande "up", les conteneurs se lancent automatiquement. Il est possible de les arrêter avec un "Ctrl + C" puisque nous pouvons les gérer manuellement depuis Docker Desktop.

Tout est affiché depuis Docker Destop avec la possibilité de cliquer pour lancer les conteneurs.



En cliquant directement sur le port, nous sommes redirigés vers les images souhaités :





4. Ajouter les hosts

Dans le fichier /etc/hosts il est nécessaire d'ajouter le nom de nos conteneurs (Apache et Base de données) afin de finir la configuration.

nano /etc/hosts

5. Lier la base de données avec Docker

Dans son .env.local il suffit d'ajouter le nom du service ainsi que le login et le nom de la base de données (qui sera créé automatiquement lors du build, sinon faire : php bin/console doctrine:database:create)

```
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data_%kernel.environment%.db"
# DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8.0.32&charset=utf8
DATABASE_URL="mysql://root:root@projetdev-db:3336/projetdevbdd"
# DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=16&charset=utf
###< doctrine/doctrine-bundle ###
```

Désormais dès qu'une entité ou une migration sera effectuée elle sera liée au conteneur Docker et sa BDD.