

## Introduction à Docker avec un Projet SQLite

### Objectif du tutoriel

- Comprendre les bases de Docker : conteneur, image, volumes.
- Lancer une application Python avec Docker.
- Manipuler une base de données SQLite dans un conteneur.
- Utiliser un projet concret pour apprendre.

### Introduction → Lexique

Image (La recette)

Modèle figé contenant l'application, toutes ses **dépendances** (bibliothèques, packages, etc.), la **configuration système** nécessaire, les **commandes d'exécution**.

Les fichiers nécessaires pour l'image docker sont: Dockerfile (obligatoire), requirements.txt, et le code source (script.py)

Conteneur (Le plat prêt à consommer)

Instance en cours d'exécution d'une image Docker sur lequel on peut exécuter des commandes

Ex: Docker ps

Permet de voir les conteneurs en cours d'exécution

Volumes

servent à **définir et gérer le stockage persistant** de manière claire et déclarative.

Les volumes sont déclarés dans le fichier docker-compose.yml

### **1. Créer des fichiers Docker:**

- Requirement.txt (Install nécessaire pour l'exécution du script)
- Dockerfile (Script pour créer une image Docker personnalisée)
- Docker-compose.yml (Gérer plusieurs conteneurs ensemble)
- Readme.md (Markdown ☐ Explication du projet)

Requirements.txt

A compléter

Dockerfile

```
FROM python:3.9-slim

# Labels pour les métadonnées
LABEL maintainer=" "
LABEL version=" "
LABEL description=" "

#Création d'un dossier app
A compléter

# Ajout de sqlite3 CLI
RUN apt update && apt install -y sqlite3

#Copie des sources de travail et les .csv dans le conteneur
COPY ./OuJeVeuxLeMettre /DoùCaVient
COPY requirements.txt .

#Install de requirement.txt
RUN pip install --no-cache-dir -r requirements.txt

#Lancement du script Python (SRC/script.py)
A compléter
```

Docker-compose.yml

```
services:
  sqlite_base:
    build: .
    container_name: sqlite_base
    volumes:
      - ./DATA:/app/DATA
      - ./SRC:/app/SRC
    command: Commande permettant que le conteneur reste ouvert pour du dev interactif
  csv-to-sqlite:
    build: .
    depends_on:
      - sqlite_base
    volumes:
      A compléter
    command: python SRC/script.py
```

## Schéma explicatif des Volumes dans Docker-compose.yml

+-----+			+-----+		
	Conteneur			Conteneur	
	csv-to-sqlite			sqlite_base	
	-----			-----	
	script.py			SQLite CLI	
	Charge les .csv		---> volume --->		Lit pme.db
	Crée pme.db			dans /app/DATA	
+-----+			+-----+		

Readme.md

### # Tuto Docker – Chargement de CSV vers SQLite avec Docker

#### ## Objectif du projet

- Lire un ou plusieurs fichiers **CSV**.
- Générer une base **SQLite** contenant les données.
- Permettre des **requêtes SQL interactives** sur la base via Docker.

---

#### ## Architecture (Docker)

Ce projet utilise **Deux services Docker** définis dans `docker-compose.yml` :

Service	Description
-----	
<b>sqlite_base</b>	Conteneur contenant la base de données SQLite.
<b>csv-to-sqlite</b>	Transforme et charge les fichiers CSV dans la base de données.

---

#### ## Arborescence du projet

...

```
|— DATA/
|   |— magasins.csv      # CSV source des magasins
|   |— produits.csv     # CSV source des produits
|   └─ ventes.csv       # CSV source des ventes
```

```

├── SRC/
│   └── script.py          # Script Python de transformation CSV → SQLite
├── docker-compose.yml     # Définition des services
├── Dockerfile             # Image Docker (Python + SQLite)
├── Readme.md              # Documentation du projet
└── requirements.txt       # Dépendances Python
...
---
| Terme      | Définition |
|-----|-----|
| **Image** | Modèle figé contenant tout ce qu'il faut pour exécuter une app (code, dépendances, etc.) |
|
| **Conteneur** | Instance d'une image en cours d'exécution, isolée du reste du système |
| **Volume** | Dossier partagé entre ton PC et le conteneur, permet de **garder les données** |
| **Dockerfile** | Fichier de configuration pour construire une image Docker personnalisée |
| **docker-compose.yml** | Fichier YAML qui décrit et orchestre plusieurs services Docker |
|
|-----|-----|

```

## 1<sup>ère</sup> exécution Docker :

### **## Comment exécuter l'application avec Docker**

#### **1. Installer Docker\*\* si ce n'est pas déjà fait :**

<https://www.docker.com/products/docker-desktop>

#### **2. Ouvrir un terminal dans le dossier du projet\*\***

#### **3. Construction des images Docker:\*\***

docker compose build

#### **4. Lancer le service de transformation CSV → SQLite:**

Pour charger les fichiers CSV dans la base SQLite, exécute la commande suivante :

docker compose run --rm csv-to-sqlite

#### **5. Pour arrêter les conteneurs après utilisation, tape :**

docker compose down

## 2. Pour aller plus loin (Requêtes SQL dans Docker)

Si tu souhaites effectuer des requêtes SQL directement dans la base de données SQLite, voici quelques étapes :

1. **Pour construire et démarrer les conteneurs, tape :**

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

2. **Vérifier que les conteneurs sont en fonctionnement :**

```
docker ps
```

3. **Pour accéder à SQLite et interagir avec la base de données, exécute :**

```
docker exec -it sqlite_base bash
```

```
sqlite3 /app/DATA/pme.db
```

4. **Voici quelques commandes utiles une fois dans SQLite :**

```
.tables          -- Voir les tables disponibles
.schema Ventes   -- Voir la structure de la table "Ventes"
SELECT * FROM Ventes;    -- Voir les ventes
Afficher le chiffre d'affaire total
SELECT SUM(V.quantite * P.prix) AS chiffre_affaires_total
FROM Ventes V
JOIN Produits P ON V.id_produit = P.id_produit;
```

5. **Pour quitter l'interface SQLite, tape :**

```
.quit
```

6. **Pour quitter le conteneur, tape:**

```
exit
```

7. **Pour arrêter les conteneurs après utilisation, tape :**

```
docker compose down
```

### 3. Partie Analyse :

Créer un nouveau service dans le docker-compose.yml pour exécuter le fichier analyse.py

Pour exécuter analyse.py

```
docker compose build  
docker compose run --rm analyse  
docker compose down
```

### Cheat Sheet Docker

```
docker r build -t mon_image .      # Construire une image  
docker run -it mon_image           # Lancer un conteneur  
docker compose up                  # Lancer tous les services définis  
docker compose down                # Arrêter les conteneurs  
docker ps                          # Voir les conteneurs en cours  
docker exec -it <nom> bash         # Entrer dans un conteneur  
docker compose run --rm <service> # Lancer une commande ponctuelle
```