# Rapport : Découverte de Docker, MySQL, Nginx et Dockerfile Multi-étapes

Ferchichi Safa

## 1 Introduction : Qu'est-ce que Docker ?

Docker est une technologie qui permet d'exécuter des applications dans des **conteneurs**. Un conteneur est une petite boîte isolée contenant un programme, ses outils, sa configuration et les fichiers nécessaires à son fonctionnement. Grâce à Docker, on peut lancer n'importe quel logiciel sans l'installer directement sur son ordinateur, ce qui évite les problèmes de compatibilité et permet de créer, supprimer ou redémarrer des environnements en quelques secondes.

Ce TP a pour objectif de découvrir Docker à travers plusieurs manipulations : créer un conteneur MySQL, tester la persistance des données, partager un volume entre plusieurs conteneurs, lancer un site web avec Nginx et utiliser un Dockerfile multi-étapes pour compiler et exécuter un programme Java.

## 2 Création d'un conteneur MySQL

Nous avons lancé un conteneur MySQL en utilisant l'image `mysql:8`. La commande suivante permet de créer un serveur MySQL fonctionnel avec un volume qui stocke les données de manière persistante :

Listing 1: Création du conteneur MySQL

```
docker run -d --name mysql \
  -e MYSQL_ROOT_PASSWORD=pass \
  -v mysql_data:/var/lib/mysql \
  mysql:8
```

Le volume `mysql_data` garantit la sauvegarde des données même si le conteneur est arrêté ou supprimé.

## 3 Création d'une base de données et insertion de données

Connexion au conteneur :

Listing 2: Connexion au serveur MySQL dans le conteneur

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

Puis création d'une base et d'une table :

Listing 3: Création de la base et de la table `users`

```
CREATE DATABASE testdb;
USE testdb;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50)
);

INSERT INTO users (name) VALUES ('Alice'), ('Bob'), ('Charlie');
SELECT * FROM users;
```

# 4 Test de persistance des données

Nous avons arrêté puis redémarré le conteneur MySQL :

Listing 4: Arrêt et redémarrage du conteneur MySQL

```
docker stop mysql
docker start mysql
```

En se reconnectant à MySQL et en relançant la requête de sélection, nous avons vérifié que les données étaient toujours présentes grâce au volume Docker.

# 5 Partage du volume entre deux conteneurs

Un second conteneur MySQL utilisant le même volume a été lancé :

Listing 5: Deuxième conteneur utilisant le même volume

```
docker run -d --name mysql2 \
  -e MYSQL_ROOT_PASSWORD=pass \
  -v mysql_data:/var/lib/mysql \
  mysql:8
```

En consultant la base depuis `mysql2`, nous retrouvons exactement les mêmes données. Cela montre que les volumes Docker sont indépendants des conteneurs.

# 6 Utilisation de deux images différentes avec un volume partagé

Nous avons ensuite expérimenté la création d'une image MySQL personnalisée très simple :

Listing 6: Dockerfile minimal pour MySQL personnalisé

```
FROM mysql:8
ENV MYSQL_ROOT_PASSWORD=pass
```

Après création d'un volume `shared_data`, deux conteneurs basés sur deux images différentes ont pu lire et écrire dans la même base, tant qu'ils utilisaient ce volume.

# 7 Dockerfile multi-étapes pour une application Java

Programme Java utilisé :

Listing 7: Programme Java `Hello.java`

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Dockerfile multi-étapes :

Listing 8: Dockerfile multi-étapes pour compiler et exécuter Hello.java

```
FROM eclipse-temurin:21-jdk AS build
WORKDIR /TP2
COPY Hello.java .
RUN javac Hello.java

FROM eclipse-temurin:21-jre
WORKDIR /TP2
COPY --from=build /TP2/Hello.class .
CMD ["java", "Hello"]
```

Lancement de l'image :

Listing 9: Construction et exécution de l'image Java

```
docker build -t hello-java .
docker run --rm hello-java
```

# 8 Création d'une page HTML avec Nginx

Lancement du serveur web :

Listing 10: Lancement d'un conteneur Nginx

```
docker run -d -p 8080:80 --name site nginx
```

Puis remplacement de la page par défaut par :

Listing 11: Fichier `index.html` servi par Nginx

```
<!DOCTYPE html>
<html>
<head>
<title>Ma page statique</title>
</head>
<body>
<h1>Bonjour depuis Docker Nginx !</h1>
<p>Ceci est une page statique servie dans un conteneur.</p>
</body>
</html>
```

# 9 Conclusion

Ce TP nous a permis de découvrir Docker de manière progressive et accessible, en comprenant son intérêt principal : fournir des environnements isolés, rapides à déployer et simples à maintenir. Nous avons appris à lancer un serveur MySQL dans un conteneur, à manipuler une base de données, puis à vérifier la persistance des données grâce aux volumes Docker. Nous avons également démontré que plusieurs conteneurs, voire plusieurs images différentes, peuvent partager les mêmes données grâce à ces volumes. La création d'un site web statique avec Nginx nous a montré la facilité avec laquelle Docker permet de déployer des services web, et l'utilisation d'un Dockerfile multi-étapes nous a initiés à des pratiques professionnelles permettant de compiler et d'exécuter une application Java dans des environnements optimisés. Dans l'ensemble, ce TP a clarifié l'intérêt de Docker dans le développement moderne : rapidité, portabilité, reproductibilité et flexibilité.