TP2 – Docker et Docker Compose

1 Introduction

Ce TP explore Docker, MySQL et la persistance des données via un volume. Objectifs : exécuter un conteneur MySQL, créer et manipuler une base, tester la persistance, et partager les données entre plusieurs conteneurs.

2 Création du conteneur MySQL

Lancement d'un conteneur avec volume pour persistance :

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

3 Création et insertion de données

Connexion au conteneur :

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

Dans MySQL :

```
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 Tests de persistance

Arrêt et redémarrage :

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

Vérification :

```
docker exec -it mysql mysql -u root -p
USE testdb;
SELECT * FROM users;
```

5 Partage du volume avec un second conteneur

Arrêt du premier conteneur :

```
docker stop mysql
```

Lancement d'un second conteneur sur le même volume :

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

Vérification des données :

```
docker exec -it mysql2 mysql -u root -p
USE testdb;
SELECT * FROM users;
```

6 Deux images partageant le même volume

Création d'un Dockerfile minimal :

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

Construction des images :

```
docker build -t mysql-img1 .
docker build -t mysql-img2 .
```

Création du volume partagé :

```
docker volume create shared_data
```

Premier conteneur :

```
docker run -d --name cont1 \
  -v shared_data:/var/lib/mysql \
  mysql-img1
docker exec -it cont1 mysql -u root -p
```

Dans MySQL :

```
CREATE DATABASE sharedDB;
USE sharedDB;

CREATE TABLE data (id INT AUTO_INCREMENT PRIMARY KEY, value VARCHAR(50));
INSERT INTO data (value) VALUES ('A1','A2');
SELECT * FROM data;
```

Second conteneur :

```
docker run -d --name cont2 \
  -v shared_data:/var/lib/mysql \
  mysql-img2
docker exec -it cont2 mysql -u root -p
```

Vérification :

```
USE sharedDB;
SELECT * FROM data;
```

7 Compilation et exécution d'un programme Java dans un conteneur Docker

Pour illustrer la création d'images personnalisées, nous avons également construit une image Docker permettant de compiler et d'exécuter un simple programme Java. Le programme utilisé est le suivant :

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

Un Dockerfile multi-étapes a été créé afin de séparer la phase de compilation et l'exécution finale :

```
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"]
```

Ce Dockerfile permet :

- de compiler le fichier `Hello.java` dans une première image contenant le JDK ;
- de produire ensuite une image finale plus légère contenant uniquement le JRE ;
- d'exécuter automatiquement le programme Java lors du lancement du conteneur.

Construction de l'image :

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

Exécution du conteneur :

```
docker run --rm hello-java
```

Le conteneur affiche alors :

```
Hello, World!
```

Cette expérimentation montre comment Docker peut être utilisé non seulement pour des services comme MySQL, mais également pour compiler et exécuter des applications simples dans un environnement isolé.

8 Conclusion

Ce TP montre que les volumes Docker assurent la persistance et le partage de données entre conteneurs, indépendamment de l'image ou du conteneur utilisé.