Create a custom Docker network and connect multiple containers

Table of Contents

Description

This section walks through the process of setting up a custom Docker network and connecting multiple containers within that network. For this example, we will use a **Java-based TodoApp** and a **MySQL database** in the same network to simulate an app communicating with its database.

Problem Statement

Running containers in isolation is common, but you often need to connect multiple containers (e.g., an application and its database). Docker networks allow containers to communicate with each other using their service names rather than exposing ports directly to the host.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-03) is required before proceeding with Lab Guide-04.

Software Requirement

- **Docker Desktop**: Installed and running on your Windows system.
- **Java JDK 11 or higher**: For building the Java-based TodoApp.
- **MySQL Docker Image**: Official MySQL image pulled from Docker Hub.
- **TodoApp Docker Image**: Make sure `Docker image` is present for todoapp.
- **TodoAPP_MYSQl**: To download the source folder **click here**

Hardware Requirement

- **CPU**: 64-bit processor with virtualization support.
- **RAM**: 4 GB minimum (8 GB recommended).
- **Disk Space**: 1 GB or more for Docker images and containers.

Implementation Steps

Step-1: Create a Custom Docker Network

1. **Create the Docker Network**:

   First, we'll create a custom network named **todoapp_network**.

   ```
   docker network create todoapp_network
   ```

    CreateNetwork

   You can verify that the network was created by running:

   ```
   docker network ls
   ```

   You should see **todoapp_network** listed.

    VerifyNetwork

2. **Network Configuration**:

   The custom network isolates your containers and allows them to communicate with each other by their container names.

Step-2: Create a MySQL Container

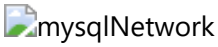
Next, we'll run a MySQL container that will act as the database for the TodoApp.

1. **Run the MySQL Container**:

   Use the following command to create a MySQL container connected to the custom network:

   ```
   docker run -d -p3306:3306 --network=todoapp_network -e
   MYSQL_ROOT_PASSWORD=P@ssw0rd -e MYSQL_DATABASE=tododb --name=mysqldb mysql
   ```

   - **--name mysql_db**: Names the container **mysql_db**.
   - **--network todoapp_network**: Connects the container to the custom network.
   - **-e**: Sets environment variables for MySQL, including root password, database name, and user credentials.



Step-3: Modify TodoApp to Use MySQL

Assume that the TodoApp connects to a MySQL database for storing tasks. Here's how to modify your **application.properties** (for Spring Boot) or the equivalent configuration for your Java app.

1. **Modify Database Connection in application.properties**:

   Add the following configurations to point to the **mysql_db** container:

```
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_PORT:3306
}/${MYSQL_DB:tododb}
spring.datasource.username=${MYSQL_USER:root}
spring.datasource.password=${MYSQL_PASSWORD:P@ssw0rd}
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDi
alect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

2. **Rebuild the TodoApp Image**:

   If you have modified your application, rebuild the Docker image for the TodoApp:

   > Note - Add the Dockerfile before building the image

```
docker build -t todoapp:1.1 .
```

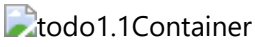
Step-4: Connect TodoApp to the Custom Network

1. **Run the TodoApp Container**:

   Now, run the **TodoApp** container and connect it to the custom network **todoapp_network**:

```
docker run -d -p8081:8081 --name todoapp --network=todoapp_network -e
MYSQL_HOST=mysqldb todoapp:1.1
```

   - **--network todoapp_network**: Connects the container to the custom network so it can communicate with the MySQL container.
   - **-p 8081:8081**: Exposes port 8081 of the container on port 8081 of the host machine.



2. **Verify the Containers are Connected**:

   To verify that both containers are on the same network, run:

   ```
   docker network inspect todoapp_network
   ```

   You should see both **my_todoapp** and **mysql_db** containers listed under the network configuration.

3. **Access the Application**:

   Open a browser and go to **http://localhost:8081/swagger-ui/index.html**. Your TodoApp should be up and running, communicating with the MySQL database.

References

For more information, refer to these official resources:

- Docker Networks: https://docs.docker.com/network/
- MySQL Docker Image: https://hub.docker.com/_/mysql
- Java MySQL Configuration: https://spring.io/guides/gs/accessing-data-mysql/