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.

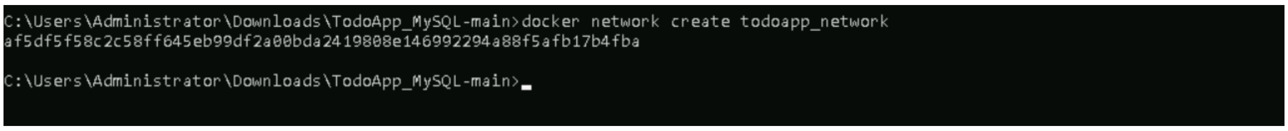
Implementation Steps

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
   ```

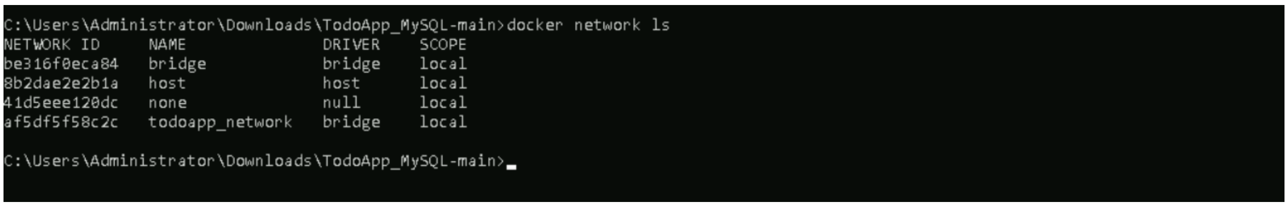
   

   ```
   C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker network create todoapp_network
   af5df5f58c2c58ff645eb99df2a00bda2419808e146992294a88f5afb17b4fba

   C:\Users\Administrator\Downloads\TodoApp_MySQL-main>
   ```

   You can verify that the network was created by running:

   ```
   docker network ls
   ```

   You should see **todoapp_network** listed.

   ```
   C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker network ls
   NETWORK ID     NAME              DRIVER    SCOPE
   be316f0eca84   bridge            bridge    local
   8b2dae2e2b1a   host              host      local
   41d5eee120dc   none              null      local
   af5df5f58c2c   todoapp_network   bridge    local

   C:\Users\Administrator\Downloads\TodoApp_MySQL-main>
   ```

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

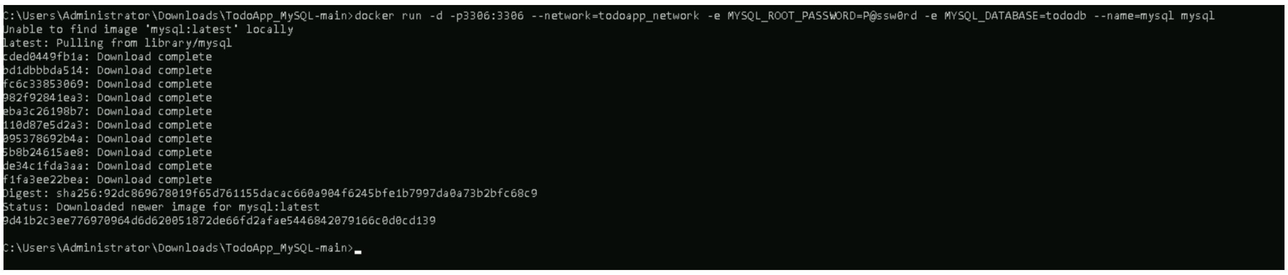
Next, we'll run a MySQL container that will act as the database for the TodoApp.

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.



```
C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker run -d -p3306:3306 --network=todoapp_network -e MYSQL_ROOT_PASSWORD=P@ssw0rd -e MYSQL_DATABASE=tododb --name=mysql mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
cded0449fb1a: Download complete
bd1dbbbda514: Download complete
fc6c33853069: Download complete
982f92841ea3: Download complete
eba3c26198b7: Download complete
110d87e5d2a3: Download complete
095378692b4a: Download complete
5b8b24615ae8: Download complete
de34c1fda3aa: Download complete
f1fa3ee22bea: Download complete
Digest: sha256:92dc869678019f65d761155dacac660a904f6245bfe1b7997da0a73b2bfc68c9
Status: Downloaded newer image for mysql:latest
9d41b2c3ee776970964d6d620051872de66fd2afae5446842079166c0d0cd139

C:\Users\Administrator\Downloads\TodoApp_MySQL-main>
```

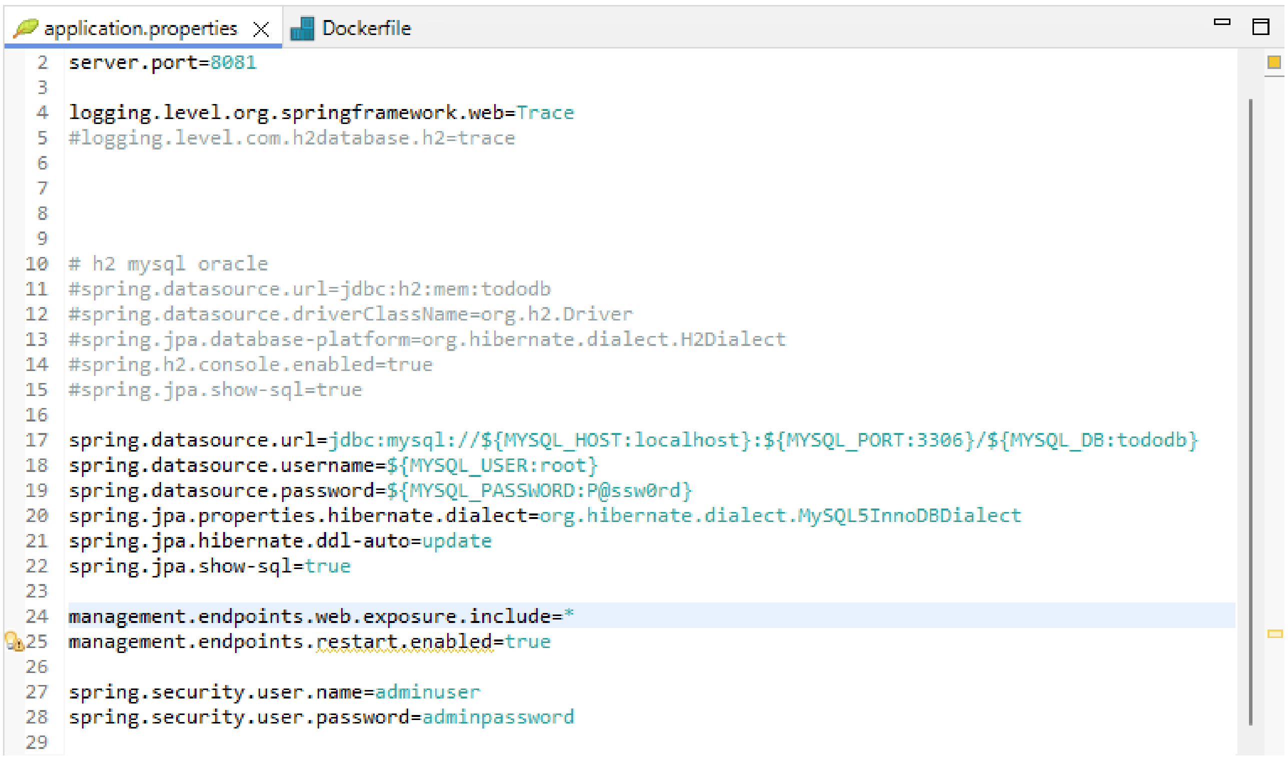
Step-3: Modify TodoApp to Use MySQL

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.MySQL5InnoDBDialect
   spring.jpa.hibernate.ddl-auto=update
   spring.jpa.show-sql=true
   ```



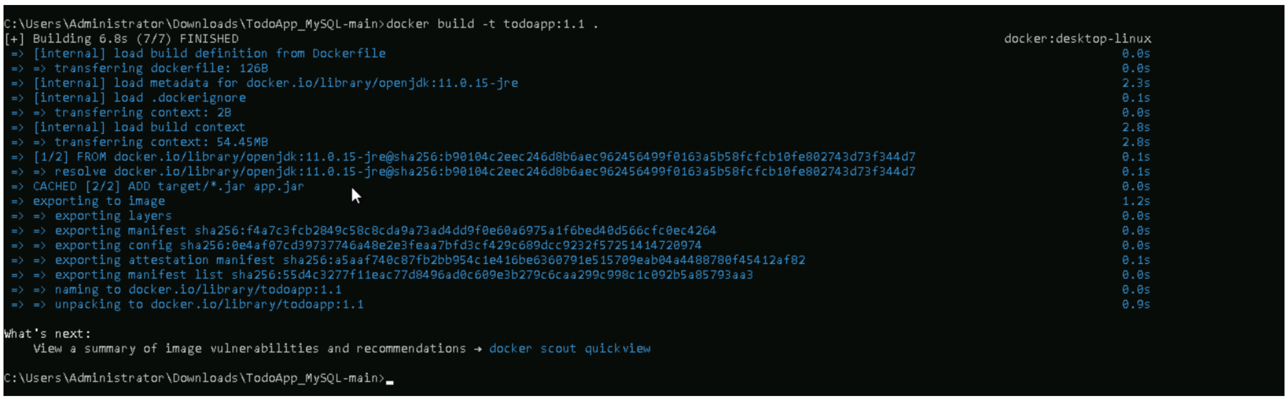
```
server.port=8081

logging.level.org.springframework.web=Trace
#logging.level.com.h2database.h2=trace




# h2 mysql oracle
#spring.datasource.url=jdbc:h2:mem:tododb
#spring.datasource.driverClassName=org.h2.Driver
#spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
#spring.h2.console.enabled=true
#spring.jpa.show-sql=true

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.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

management.endpoints.web.exposure.include=*
management.endpoints.restart.enabled=true

spring.security.user.name=adminuser
spring.security.user.password=adminpassword

```

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 .
   ```



```
C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker build -t todoapp:1.1 .
[+] Building 6.8s (7/7) FINISHED                                                                         docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                     0.0s
 => => transferring dockerfile: 126B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/openjdk:11.0.15-jre                                                   2.3s
 => [internal] load .dockerignore                                                                                        0.1s
 => => transferring context: 2B                                                                                          0.0s
 => [internal] load build context                                                                                        2.8s
 => => transferring context: 54.45MB                                                                                     2.8s
 => [1/2] FROM docker.io/library/openjdk:11.0.15-jre@sha256:b90104c2eec246d8b6aec962456499f0163a5b58fcfcb10fe802743d73f344d7  0.1s
 => => resolve docker.io/library/openjdk:11.0.15-jre@sha256:b90104c2eec246d8b6aec962456499f0163a5b58fcfcb10fe802743d73f344d7  0.1s
 => CACHED [2/2] ADD target/*.jar app.jar                                                                                0.0s
 => exporting to image                                                                                                   1.2s
 => => exporting layers                                                                                                  0.0s
 => => exporting manifest sha256:f4a7c3fcb2849c58c8cda9a73ad4dd9f0e60a6975a1f6bed40d566cfc0ec4264                        0.0s
 => => exporting config sha256:0e4af07cd39737746a48e2e3feaa7bfd3cf429c689dcc9232f57251414720974                          0.0s
 => => exporting attestation manifest sha256:a5aaf740c87fb2bb954c1e416be6360791e515709eab04a4488780f45412af82            0.1s
 => => exporting manifest list sha256:55d4c3277f11eac77d8496ad0c609e3b279c6caa299c998c1c092b5a85793aa3                   0.0s
 => => naming to docker.io/library/todoapp:1.1                                                                           0.0s
 => => unpacking to docker.io/library/todoapp:1.1                                                                        0.9s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Users\Administrator\Downloads\TodoApp_MySQL-main>
```

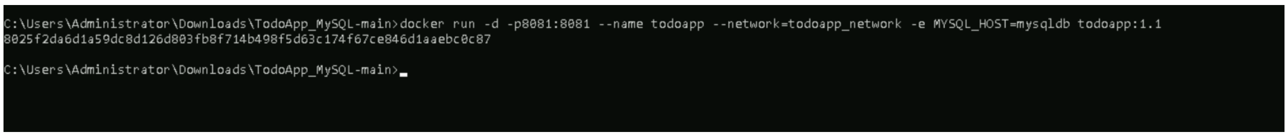
Step-4: Connect TodoApp to the Custom Network

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.



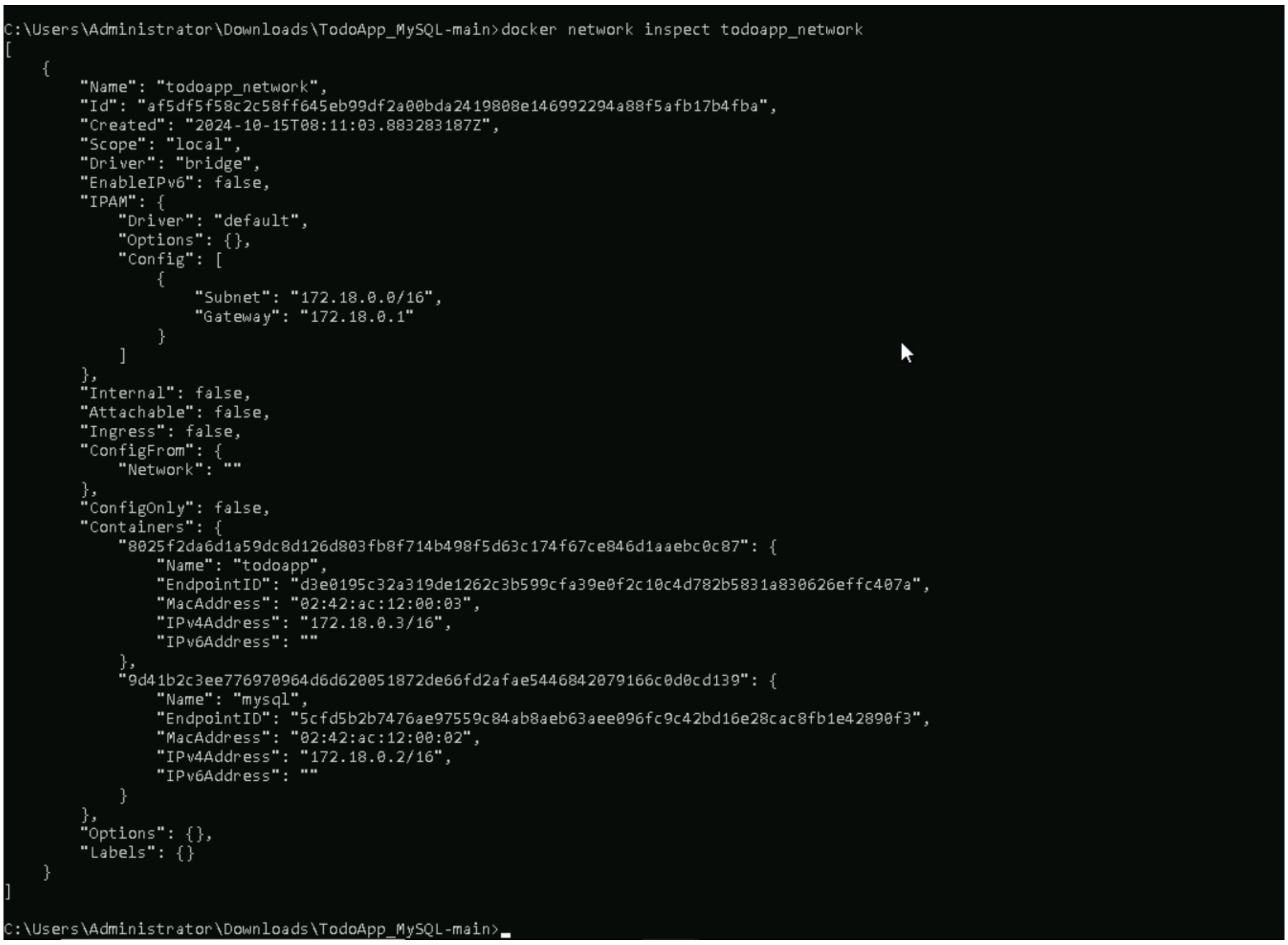
```
C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker run -d -p8081:8081 --name todoapp --network=todoapp_network -e MYSQL_HOST=mysqldb todoapp:1.1
8025f2da6d1a59dc8d126d803fb8f714b498f5d63c174f67ce846d1aaebc0c87

C:\Users\Administrator\Downloads\TodoApp_MySQL-main>
```

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.



```
C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker network inspect todoapp_network
[
    {
        "Name": "todoapp_network",
        "Id": "af5df5f58c2c58ff645eb99df2a00bda2419808e146992294a88f5afb17b4fba",
        "Created": "2024-10-15T08:11:03.883283187Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "8025f2da6d1a59dc8d126d803fb8f714b498f5d63c174f67ce846d1aaebc0c87": {
                "Name": "todoapp",
                "EndpointID": "d3e0195c32a319de1262c3b599cfa39e0f2c10c4d782b5831a830626effc407a",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "9d41b2c3ee776970964d6d620051872de66fd2afae5446842079166c0d0cd139": {
                "Name": "mysql",
                "EndpointID": "5cfd5b2b7476ae97559c84ab8aeb63aee096fc9c42bd16e28cac8fb1e42890f3",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]

C:\Users\Administrator\Downloads\TodoApp_MySQL-main>
```

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/
