Automate the deployment of Docker containers using a simple shell script

Table of Contents

Description

This document explains how to automate the deployment of Docker containers using a **simple shell script**. The script will handle tasks such as building Docker images, running containers, and ensuring the containers are up and running with minimal manual intervention.

Problem Statement

Manually deploying Docker containers can become time-consuming and error-prone, especially when handling multiple containers or updating services. Automating the deployment process using a **shell script** ensures consistent and efficient container management, saving time and reducing human error.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-06) is required before proceeding with Lab Guide-07.

Software Required

- **Docker Desktop**: Installed on your Windows machine.
- **Bash (for Windows)**: Git Bash or WSL (Windows Subsystem for Linux) to run shell scripts.
- **TodoAPP_MYSQl**: To download the source folder **click here**

Hardware Requirement

- Minimum of 4 GB RAM

- At least 2 cores in the processor
- 5 GB of free storage space for Docker images and containers

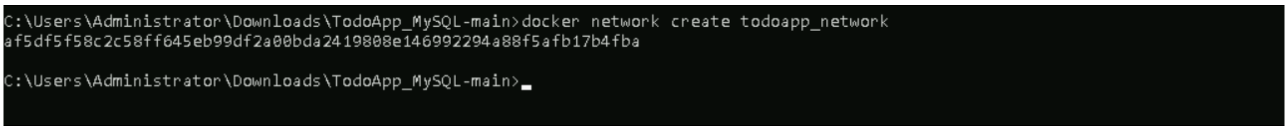
Implementation Steps

Step-1: Create a Custom Docker Network and MySQL Container

1. **Create the Docker Network**:

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

   ```
   docker network create todoapp_network
   ```

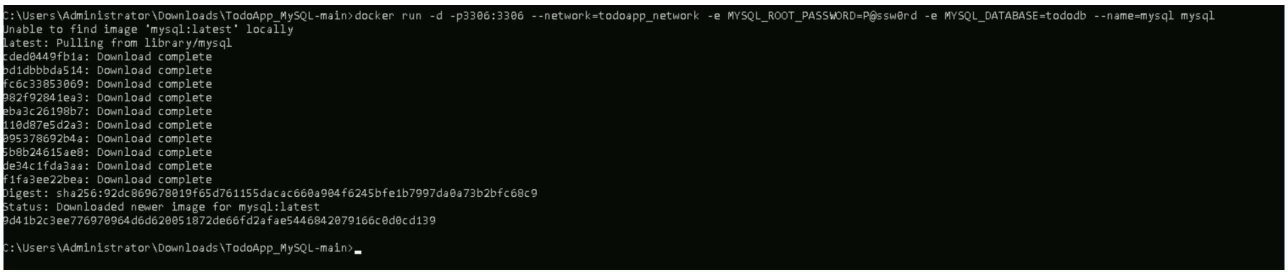
   

   ```
   C:\Users\Administrator\Downloads\TodoApp_MySQL-main>docker network create todoapp_network
   af5df5f58c2c58ff645eb99df2a00bda2419808e146992294a88f5afb17b4fba

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

2. **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
   ```

   ```
   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-2: Define Deployment Requirements

Before automating, define what the script should do. In this example, the following tasks will be automated for the **TodoApp** Docker container:

- **Build** the Docker image.
- **Run** the container.
- **Remove** any old, running container with the same name to ensure a fresh start.
- **Restart** the container in case of failure.

Step-3: Create the Shell Script

Let's create a simple shell script that automates the deployment of a Docker container. The script will:

- Stop and remove any existing container with the same name.

- Build a new Docker image.
- Run the new container.
- Ensure the container is restarted automatically if it stops.

`deploy_todoapp.sh` **- Shell Script**

```
#!/bin/bash

# Set variables
IMAGE_NAME="todoapp_image"
CONTAINER_NAME="todoapp_container"
PORT="8081:8081"

echo "Starting the automated deployment of TodoApp..."

# Step 1: Stop and remove any old running container
if [ $(docker ps -a -q -f name=$CONTAINER_NAME) ]; then
    echo "Stopping and removing existing container: $CONTAINER_NAME..."
    docker stop $CONTAINER_NAME
    docker rm $CONTAINER_NAME
fi

# Step 2: Build Docker image
echo "Building Docker image: $IMAGE_NAME..."
docker build -t $IMAGE_NAME .

# Step 3: Run Docker container
echo "Running Docker container: $CONTAINER_NAME..."
docker run -d -p8081:8081 --name todoapp_container --network=todoapp_network -e
MYSQL_HOST=mysqldb --restart unless-stopped $IMAGE_NAME

# Step 4: Verify the container is running
if [ $(docker ps -q -f name=$CONTAINER_NAME) ]; then
    echo "TodoApp is running successfully at http://localhost:8081/swagger-
ui/index.html"
else
    echo "Failed to start TodoApp. Check the Docker logs for more details."
fi
```

Step-4: Automate Docker Container Deployment

3.1 Make the Script Executable

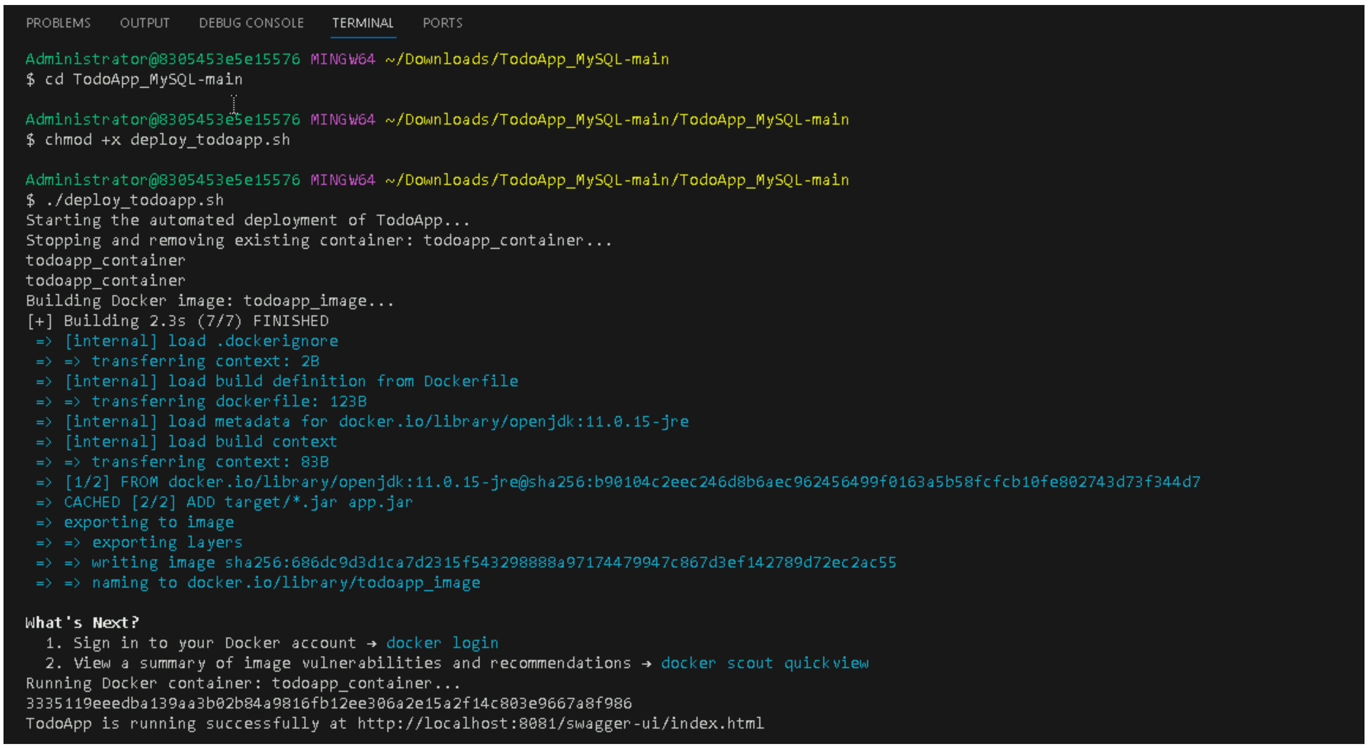
Make the script executable by running the following command:

```
chmod +x deploy_todoapp.sh
```

3.2 Run the Script

Execute the script to automate the deployment of the **TodoApp** Docker container:

```
./deploy_todoapp.sh
```



```
Administrator@8305453e5e15576 MINGW64 ~/Downloads/TodoApp_MySQL-main
$ cd TodoApp_MySQL-main

Administrator@8305453e5e15576 MINGW64 ~/Downloads/TodoApp_MySQL-main/TodoApp_MySQL-main
$ chmod +x deploy_todoapp.sh

Administrator@8305453e5e15576 MINGW64 ~/Downloads/TodoApp_MySQL-main/TodoApp_MySQL-main
$ ./deploy_todoapp.sh
Starting the automated deployment of TodoApp...
Stopping and removing existing container: todoapp_container...
todoapp_container
todoapp_container
Building Docker image: todoapp_image...
[+] Building 2.3s (7/7) FINISHED
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 123B
 => [internal] load metadata for docker.io/library/openjdk:11.0.15-jre
 => [internal] load build context
 => => transferring context: 83B
 => [1/2] FROM docker.io/library/openjdk:11.0.15-jre@sha256:b90104c2eec246d8b6aec962456499f0163a5b58fcfcb10fe802743d73f344d7
 => CACHED [2/2] ADD target/*.jar app.jar
 => exporting to image
 => => exporting layers
 => => writing image sha256:686dc9d3d1ca7d2315f543298888a97174479947c867d3ef142789d72ec2ac55
 => => naming to docker.io/library/todoapp_image

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview
Running Docker container: todoapp_container...
3335119eeedba139aa3b02b84a9816fb12ee306a2e15a2f14c803e9667a8f986
TodoApp is running successfully at http://localhost:8081/swagger-ui/index.html
```

The script will:

- Stop and remove any previously running container with the same name.
- Build the Docker image from the `Dockerfile` in the current directory.
- Run the Docker container and map port `8081` on the container to port `8081` on the host machine.
- Automatically restart the container if it stops.

Step-5: Test the Script

After running the script, open your web browser and navigate to: http://localhost:8081/swagger-ui/index.html

You should see the **TodoApp** running. If any errors occur, check the Docker logs by running:

```
docker logs todoapp_container
```

This will help troubleshoot any issues with the deployment.

References

- Docker Official Documentation: https://docs.docker.com/
- Bash Scripting Guide: https://www.gnu.org/software/bash/manual/

- Automating Docker Workflows: https://docs.docker.com/engine/admin/