Run a container from an existing image and manage it using Docker commands

Table of Contents

- Description
- Problem Statement
- Prerequisites
 - Software Requirement
 - Hardware Requirement
- Implementation Steps
 - Step-1: Run a Container from an Existing Image
 - Step-2: Managing Running Containers
 - Step-3: Stopping, Restarting, and Removing Containers
 - Step-4: Accessing Logs and Container Status
- References

Description

In this section, we'll explain how to run a **Java-based TodoApp** container from a pre-existing Docker image and manage it using common Docker commands. This includes how to start, stop, remove, and inspect running containers, as well as monitor logs.

Problem Statement

Once you've built a Docker image for the Java-based TodoApp (or any other image), you need to manage the container lifecycle: starting, stopping, inspecting, and removing containers efficiently. Docker provides several commands to manage these tasks.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-02) is required before proceeding with Lab Guide-03.

Software Requirement

- **Docker Desktop**: Installed and running on a Windows system.
- TodoApp Docker Image: Make sure Docker image is present for todoapp.

Hardware Requirement

- **CPU**: 64-bit processor with virtualization support.
- RAM: 4 GB minimum (8 GB recommended).
- Disk Space: 500 MB or more for Docker images and containers.

Implementation Steps

Step-1: Run a Container from an Existing Image

You can run a Docker container from the existing **Java-based TodoApp** image (**todoapp**) using the **docker run** command.

1. Run the Container:

```
docker run -d -p 8081:8081 --name my_todoapp todoapp
```

- **-d**: Runs the container in **detached mode** (in the background).
- -p 8081:8081: Maps port 8081 of the host to port 8081 of the container.
- --name my_todoapp: Names the container my_todoapp for easier management.
- **todoapp**: The name and tag of the image you built.



2. Verify the Container is Running:

You can list all running containers with the following command:

```
docker ps
```

You should see output similar to:



3. Access the Application:

Open a browser and navigate to **http://localhost:8081/swagger-ui/index.html** to see your TodoApp running inside the container.



Step-2: Managing Running Containers

Once your container is up and running, Docker provides several commands to manage the containers.

1. List All Running Containers:

```
docker ps
```

2. List All Containers (Including Stopped Containers):

docker ps -a

This will list all containers, including those that have stopped. ListAllContainers

Step-3: Stopping, Restarting, and Removing Containers

1. Stop a Running Container:

Use the **docker stop** command to stop a running container:

docker stop my_todoapp



This will gracefully stop the **my_todoapp** container.

ListAllContainers

2. Restart a Stopped Container:

Use the **docker start** command to restart the container:

docker start my_todoapp

DockerStart

This starts the container with the previous configuration (i.e., running the same image).

3. Remove a Stopped Container:

To remove a stopped container, use the **docker rm** command:

docker rm my_todoapp

RemoveContainer

Note: A running container cannot be removed. You need to stop it first using **docker stop**.

4. Force Remove a Running Container:

If you need to remove a running container, use the **-f** flag:

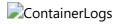
docker rm -f my_todoapp

Step-4: Accessing Logs and Container Status

1. View Container Logs:

To see the logs from a running container, use the **docker logs** command:

```
docker logs my_todoapp
```



This shows the standard output logs of the running application.

2. Follow Logs in Real-Time:

If you want to stream the logs in real-time, add the **-f** flag:

```
docker logs -f my_todoapp
```

RealtimeLogs

This will continue displaying logs as they are written.

3. Inspect a Container's Details:

To get detailed information about the container, use the **docker inspect** command:

```
docker inspect my_todoapp
```



This will return JSON-formatted data that contains all the configuration and status details of the container.

4. View the Running Container's Status:

Use **docker stats** to monitor resource usage like CPU, memory, network, etc.

```
docker stats my_todoapp
```



This command continuously updates with live statistics of the container's resource consumption.

References

For more detailed usage and documentation of Docker commands:

- Docker Official Documentation: https://docs.docker.com/
- Docker Command Reference: https://docs.docker.com/engine/reference/commandline/docker/