SOFTWARE ENGINEERING LAB

EXERCISE – 7

TOPIC – 3

CREATE AND PUSH DOCKER FILE IMAGE

By following these Commands, you will learn how to:

- Creating a JavaScript calculator program.
- Building and running a Docker container.
- Pushing the container image to Docker Hub for sharing.
- Pulling and reusing the image on another system.

- **Note: At every step take screenshots and save in a document**

Step 1: Create the JavaScript File

Create a file named calculator.js in your project directory.

Add the following code to define simple calculator functions:

```
// calculator.js
function add(a, b) {
    return a + b;
}
function subtract(a, b) {
    return a - b;
}
function multiply(a, b) {
    return a * b;
}
```

```
function divide(a, b) {
    if (b === 0) {
        return "Cannot divide by zero!";
    }
    return a / b;
}
// Print the calculated values
console.log("Addition (2 + 3):", add(2, 3));
console.log("Subtraction (5 - 2):", subtract(5, 2));
console.log("Multiplication (4 * 3):", multiply(4, 3));
console.log("Division (10 / 2):", divide(10, 2));
```

Purpose: This script performs basic arithmetic operations and logs the results to the console.

Output: When run, this program prints the results of the calculations.

Step 2: Create a Dockerfile

The Dockerfile contains instructions for building the Docker image.

1. Create a file named Dockerfile (no file extension).
2. Add the following content:

```
FROM node:16-alpine
WORKDIR /app
COPY calculator.js /app
CMD ["node", "calculator.js"]
```

Explanation of the Dockerfile

- FROM node:16-alpine

This uses the Alpine version of Node.js 16, which is a minimal, lightweight image.

- WORKDIR /app

Sets the working directory inside the container to /app, where your app files will go.

- COPY calculator.js /app

Copies the calculator.js file into the container.

- CMD ["node", "calculator.js"]

Tells Docker to run the calculator.js file using Node.js when the container starts.

Step 3: Build the Docker Image

1. Open a terminal in the directory containing your Dockerfile and calculator.js.
2. Run the following command:

```
docker build -t simple-calculator .
```

- docker build: This command builds an image from the Dockerfile.
- -t simple-calculator: Tags the image with the name simple-calculator.
- .: Refers to the current directory where the Dockerfile is located.

Step 4: Run the Docker Container

Run the container using the image you just created:

```
docker run simple-calculator
```

- docker run: Starts a new container from the image.
- simple-calculator: The name of the image to use.

Expected Output: The console will display the results of the calculations.

Step 6: Push the Image to Docker Hub

Ensure you have a Docker Hub account. Log in using the following command:

```
docker login
```

Enter your **Docker Hub username** and **password** if prompted

Tag the Image for Docker Hub:

Docker images need to be tagged with your Docker Hub username before they can be uploaded:

```
docker tag simple-calculator your-dockerhub-username/simple-calculator
```

Replace your-dockerhub-username with your actual Docker Hub username.

Push the Image to Docker Hub:

Push the tagged image to Docker Hub:

```
docker push your-dockerhub-username/simple-calculator
```

Once complete, your image will be available in your Docker Hub repository.

Step 7: Pull the Image from Docker Hub

To test the reusability, first, remove the existing image and container:

```
# List all containers (even stopped ones)

docker ps -a

# Remove the container by ID

docker rm <container-id>

# Remove the local image

docker rmi your-dockerhub-username/simple-calculator
```

On another machine or after deleting the local image and container, pull the image from Docker Hub:

```
docker pull your-dockerhub-username/simple-calculator
```

Step 8: Run the Pulled Image

Run the container from the pulled image:

```
docker run your-dockerhub-username/simple-calculator
```

You'll see the same calculation results printed to the console.

```
#List all the containers

docker ps -a

#Delete the container by its ID

docker rm <container-id>

# List all images

docker images

# Remove unused images by ID

docker rmi <image-id>

#Log out of Docker Hub.

docker logout
```