Lab 3 – P1: Docker

Docker is a platform that enables you to automate the deployment, scaling, and management of applications in lightweight containers.

Installing Docker

For Windows and macOS:

1. Download Docker Desktop: Go to the Docker website and download Docker Desktop for Windows or macOS.
2. Install Docker Desktop: Follow the installer instructions.
3. Start Docker Desktop: After installation, launch Docker Desktop.

For Linux:

1.Update the apt package index:

```bash
sudo apt update
```

2.Install Docker:

```bash
sudo apt install docker.io
```

3.Start Docker:

```bash
sudo systemctl start docker
```

4.Enable Docker to start at boot:

```bash
sudo systemctl enable docker
```

5.Add user to Docker Group:

```bash
sudo usermod -aG docker $USER
```

6.Restart the OS.

Verify Installation

Run the following command to check if Docker installed correctly:

```bash
docker --version
```

Basic Docker Commands

After installing Docker, try these commands to get familiar with Docker's functionality:

1. Check Docker Version

```bash
docker --version
```

This shows both the Docker client and server versions.

2. Run Your First Container

```bash
docker run hello-world
```

This command pulls the hello-world image from Docker Hub (if it's not already on your machine) and runs it. You'll see a confirmation message that Docker is installed and working correctly.

3. List of the Running Containers

```bash
docker ps
```

This lists all currently running containers. To see all containers (running and stopped), use:

```bash
docker ps -a
```

Docker Hub

4. Pull an Image from Docker Hub

```bash
docker pull ubuntu
```

This pulls the ubuntu image from Docker Hub, allowing you to use Ubuntu within a container.

5. Run a Command in a Container

```bash
docker run ubuntu echo "Hello from Docker!"
```

This starts a new Ubuntu container and executes the echo command inside it.

6. Start an Interactive Shell in a Container

```bash
docker run -it ubuntu /bin/bash
```

The -it option opens an interactive shell, allowing you to execute commands within the container directly.

7. Stop a Running Container

```bash
docker stop <container_id>
```

You can find the container ID by running docker ps.

8. Remove a Container

```bash
docker rm <container_id>
```

Make sure the container is stopped before removing it.

9. Remove an Image

```bash
docker rmi <image_name>
```

Scenario Overview

You will create a simple web server using Python and Flask, Dockerize the application, and upload it to Docker Hub.

Steps

1. Create a Simple Web Application

1.Set Up the Project Directory:

Create a new directory named docker-lab and navigate into it.

```bash
mkdir docker-lab
```

```bash
cd docker-lab
```

2.Create the Flask Application:

Create a file named app.py inside docker-lab.

```
# app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def home():
    return "Hello, Docker!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

3.Create a Requirements File:

Create a requirements.txt file to specify Flask as a dependency.

```
Flask==2.0.1
Werkzeug==2.0.2
```

2. Write a Docker file

The Docker file is a blueprint for Docker to build the image.

1.Create a Docker file:

- Inside the docker-lab directory, create a file named Dockerfile (no extension) with this content:

```
# Use an official Python runtime as a base image
FROM python:3.9

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install dependencies
RUN pip install -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable
ENV NAME Docker

# Run app.py when the container launches
CMD ["python", "app.py"]
```

3. Build and Run the Docker Image

1.Build the Docker Image:

In your terminal, navigate to the docker-lab directory and build your image with a tag name flask-app.

```bash
docker build -t flask-app .
```

2.Run the Docker Container:

Use the following command to run a container from your image:

```bash
docker run -p 5000:5000 flask-app
```

3.Verify the Application:

Open a web browser and go to http://localhost:5000. You should see the message "Hello, Docker!"

4. Push the Image to Docker Hub

1.Create a Docker Hub Account (if you don't have one):

Go to Docker Hub and create an account.

2.Log In to Docker Hub:

Run the following command in your terminal:

```bash
docker login
```

Enter your Docker Hub username and password.

3.Tag the Docker Image:

To upload your image to Docker Hub, tag it with your Docker Hub username.

```bash
docker tag flask-app <your-dockerhub-username>/flask-app
```

4.Push the Image to Docker Hub:

Run the following command to push your image:

```bash
docker push <your-dockerhub-username>/flask-app
```

5.Verify on Docker Hub:

Go to your Docker Hub repository, and you should see the newly pushed image.

5. Pull and Run the Docker Image from Docker Hub

On any machine with Docker installed, use the following command to pull and run your image directly from Docker Hub:

```bash
docker pull <your-dockerhub-username>/flask-app
```

```bash
docker run -p 5000:5000 <your-dockerhub-username>/flask-app
```

Summary

In this lab, you created a Dockerized Flask web application, built a Docker image, ran it locally, and pushed it to Docker Hub for others to access.

Lab Submission

Students must submit:

- Screenshots of key commands & outputs.

Prepared by:

Marwa M. El-Azab
Cybersecurity Research Assistant (MSc Candidate), Teaching Assistant at AASTMT

Omar M. El-Azab
Cybersecurity Instructor, Technical Advisor, Adjunct Teaching Assistant at AASTMT