**Lab Exercise 3- Docker Image Build and Push**

**Objective**

Learn how to create a Docker image for a simple HTML web application, build the image, and push it to Docker Hub.

**Prerequisites**

* Docker installed on your machine. Refer to the official Docker website for installation instructions.
* A Docker Hub account. Sign up at Docker Hub if you don’t have one.

**Exercise Steps**

**1. Create a Project Directory**

Create a directory for your Docker project. Open your terminal or command prompt and run:

mkdir simple-html-app

cd simple-html-app

**2. Create the HTML File**

Create a simple HTML file named index.html in your project directory:

nano index.html

Add the following HTML code to index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Simple HTML App</title>

</head>

<body>

<h1>Hello, Docker!</h1>

<p>This is a simple HTML web application running in a Docker container.</p>

</body>

</html>

**3. Create a Dockerfile**

Create a new file named Dockerfile in your project directory. This file will define how to build your Docker image.

nano Dockerfile

Add the following content to the Dockerfile. This example uses the official Nginx image to serve the HTML file.

# Use the official Nginx image as a base

FROM nginx:alpine

# Copy the HTML file to the Nginx HTML directory

COPY index.html /usr/share/nginx/html/index.html

# Expose the port that Nginx listens on

EXPOSE 80

**4. Build the Docker Image**

Now that you have your Dockerfile and HTML file, you can build your Docker image. Run the following command in your terminal (make sure you're in the project directory):

docker build -t <your-dockerhub-username>/simple-html-app:latest .

* Replace <your-dockerhub-username> with your actual Docker Hub username.

**5. Verify the Image Build**

After the build completes, you can verify that your image has been created by listing all Docker images:

docker images

**6. Run the Docker Container Locally**

You can test your image locally by running a container. Map port 80 of the container to port 8080 on your host:

docker run -d -p 8080:80 <your-dockerhub-username>/simple-html-app:latest

**7. Access the Web Application**

Open your web browser and go to http://localhost:8080. You should see your "Hello, Docker!" message displayed on the page.

**8. Log in to Docker Hub**

Before you can push your image, you need to log in to your Docker Hub account:

docker login

Enter your Docker Hub username and password when prompted.

**9. Push the Image to Docker Hub**

Now that you are logged in, you can push your Docker image to Docker Hub:

docker push <your-dockerhub-username>/simple-html-app:latest

**10. Verify the Push**

Go to your Docker Hub account in your web browser and navigate to the “Repositories” section. You should see your simple-html-app repository with the pushed image.

**11. Clean Up**

If you want to remove the local image after you are done testing, you can run:

docker rmi <your-dockerhub-username>/simple-html-app:latest

If you want to remove any containers, you can find their IDs using:

docker ps -a

Then use:

docker rm <container\_id>

**12. Summary of Basic Commands**

Here’s a recap of the key Docker commands you practiced:

* **Build an image**: docker build -t <username>/<image-name>:<tag> .
* **List images**: docker images
* **Run a container**: docker run -d -p <host-port>:<container-port> <image-name>
* **Log in to Docker Hub**: docker login
* **Push an image**: docker push <username>/<image-name>:<tag>
* **Remove an image**: docker rmi <image-name>
* **Remove a container**: docker rm <container\_id>

**Conclusion**

By completing this lab exercise, you should now have a foundational understanding of how to build and push a Docker image for a simple HTML web application. This knowledge will be useful as you continue to work with Docker and containerization in your projects. Feel free to experiment further by modifying the HTML content, creating new images, or exploring other features of Docker!