Docker Lab

**Installing Docker and Implementing Docker Commands**

Here's a guide on installing Docker and using some common Docker commands. I'll also outline the process of creating a Docker image and pushing it to Docker Hub.

**1. Install Docker Desktop**

* Windows:
  1. Download the Docker Desktop installer from the official Docker website: [https://www.docker.com/products/docker-desktop](https://www.google.com/url?sa=E&source=gmail&q=https://www.docker.com/products/docker-desktop)
  2. Run the installer and follow the on-screen instructions. Ensure that the "Use WSL 2 instead of Hyper-V" option is selected if your system supports it.
  3. Restart your computer after installation.

**2. Verify Installation**

Open a terminal or command prompt and run:

Bash

docker --version

This should display the installed Docker version.

**3. Basic Docker Commands**

* docker pull <image\_name>: Downloads an image from Docker Hub.
  + Example: docker pull ubuntu
* docker images: Lists all downloaded images.
* docker run <image\_name>: Creates and starts a container from an image.
  + Example: docker run -it ubuntu /bin/bash (-it for interactive mode)
* docker ps: Lists running containers.
* docker ps -a: Lists all containers (running and stopped).
* docker stop <container\_id>: Stops a running container.
* docker rm <container\_id>: Removes a stopped container.
* docker rmi <image\_id>: Removes an image.

**4. Create a Docker Image**

To create a Docker image of your application, you'll need a Dockerfile. This file contains instructions for building the image.

Here's an example Dockerfile for a simple Python application:

Dockerfile

# Use a base image

FROM python:3.9-slim-buster

# Set the working directory

WORKDIR /app

# Copy requirements file

COPY requirements.txt .

# Install dependencies

RUN pip install -r requirements.txt

# Copy application code

COPY . .

# Set the command to run the application

CMD ["python", "app.py"]

To build the image, navigate to the directory containing the Dockerfile in your terminal and run:

Bash

docker build -t <your\_dockerhub\_username>/<image\_name> .

(Replace <your\_dockerhub\_username> and <image\_name> with your Docker Hub username and desired image name.)

**5. Deploy to Docker Hub**

1. Create a Docker Hub account: If you don't have one, sign up at [https://hub.docker.com/](https://www.google.com/url?sa=E&source=gmail&q=https://hub.docker.com/)
2. Log in to Docker Hub:

<!-- end list -->

Bash

docker login

Enter your Docker Hub username and password.

1. Push the image:

<!-- end list -->

Bash

docker push <your\_dockerhub\_username>/<image\_name>

Your image is now available on Docker Hub!

Note: This is a basic guide. For more advanced configurations and best practices, refer to the official Docker documentation.

Example

Let's create a simple Python "Hello, world!" application and Dockerize it, then push it to Docker Hub.

1. Create the Python Application (app.py):

Create a file named app.py with the following content:

Python

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def hello():

return "Hello, Dockerized World!"

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True, host='0.0.0.0', port=5000)

This uses Flask, a simple web framework, to create a web server that returns "Hello, Dockerized World!".

2. Create the requirements.txt:

Because we're using Flask, we need to list it as a dependency. Create a file named requirements.txt with this line:

Flask

3. Create the Dockerfile:

Create a file named Dockerfile (no extension) in the same directory as app.py and requirements.txt with the following content:

Dockerfile

# Use a base image with Python 3.9

FROM python:3.9-slim-buster

# Set the working directory inside the container

WORKDIR /app

# Copy the requirements file

COPY requirements.txt .

# Install the dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Copy the application code

COPY . .

# Expose port 5000

EXPOSE 5000

# Set the command to run the application

CMD ["python", "app.py"]

4. Build the Docker Image:

Open your terminal or command prompt, navigate to the directory containing the Dockerfile, app.py, and requirements.txt, and run the following command (replace your\_dockerhub\_username with your actual Docker Hub username):

Bash

docker build -t your\_dockerhub\_username/hello-docker .

This command does the following:

* docker build: Starts the build process.
* -t your\_dockerhub\_username/hello-docker: Tags the image with your Docker Hub username and the name hello-docker. This is crucial for pushing to Docker Hub.
* .: Specifies the build context (the current directory).

5. Run the Docker Image (Locally):

Before pushing to Docker Hub, let's test the image locally:

Bash

docker run -p 5000:5000 your\_dockerhub\_username/hello-docker

* docker run: Starts a container.
* -p 5000:5000: Maps port 5000 on your host machine to port 5000 in the container. This allows you to access the web application.

Open your web browser and go to http://localhost:5000. You should see "Hello, Dockerized World!".

Press Ctrl+C in your terminal to stop the container.

6. Log in to Docker Hub:

If you haven't already, log in to Docker Hub from your terminal:

Bash

docker login

Enter your Docker Hub username and password.

7. Push the Docker Image to Docker Hub:

Now, push the image to Docker Hub:

Bash

docker push your\_dockerhub\_username/hello-docker

This uploads your image to your Docker Hub repository.

8. Verify on Docker Hub:

Go to the Docker Hub website and log in. You should now see your hello-docker repository under your username.

Complete Example File Structure:

hello-docker-project/

├── app.py

├── requirements.txt

└── Dockerfile

This step-by-step example should make the process much clearer. You now have a working Docker image of a simple web application hosted on Docker Hub. You can now use this image to deploy your application on any system with Docker installed by simply running docker pull your\_dockerhub\_username/hello-docker and docker run -p 5000:5000 your\_dockerhub\_username/hello-docker.