Lab Assignment 5: Docker and Containerization - Documentation

Due Date Submit before 23:59 on Friday, November 3, 2023

Student Name: Emmanuel Alafonye

Step 1: Install Docker Desktop: I installed Docker Desktop on my Mac computer by following the instructions in the Docker documentation downloaded from <https://docs.docker.com/engine/install/>

A screenshot of a phone

Description automatically generated

Step 2: Create a Docker Hub Account: I created a Docker Hub account at https://hub.docker.com and signed in.

Step 3: Find a Python Image: I searched for a Python image on Docker Hub, and I found the official Python image.

Step 4: Pull Python Image: I used the command, ***docker pull python*** to download the Python image from Docker Hub.

A screenshot of a computer

Description automatically generated

Step 5: List Docker Images: To confirm the image download, I ran ***docker image ls***, and I observed the Python image in the list.

Step 6: Rename Python Image: I renamed the Python image to "ensf607" using the command ***docker tag python ensf607***.

A screenshot of a computer

Description automatically generated

Step 7: Run a Container: I ran a Docker container using the renamed image: ***docker run -itd ensf607***. The `-itd` options were used, and I understood that they control interactive and detached execution of the container.

Description of -itd: “i” means interactive, which allows the user to interact with the docker containers input standard – stdin. This means that the user can interact and send input commands without which there is no interaction with the docker container.

The “t” is a pseudo terminal of the container that provides a more interactive and responsive experience for the command line interactions with docker containers.

The “d” means detached, is used to run the docker container in the background processes and a response is immediately given back to the user that provides the prompt. Hence it is very useful for continuous running of the docker container without blocking the terminal.

Step 8: List Active Containers: To see the active containers, I executed ***docker ps***. Below is the random id: 69ee9abe2bda3367c4de61b7e0ee4af2fc72b28e24b74ffede77fc1017c198d8

Step 10: Rename Container: I renamed the Python container using the command ***docker rename ensf607 ensf607\_alafonye***. I confirmed the change with docker ps.

Step 11: Enter the Container

A screenshot of a computer

Description automatically generated

I entered the container using the command ***docker exec -it ensf607\_alafonye sh***. I understood that "sh" stands for the shell environment, allowing me to run Linux commands inside the container.

Command Line within Container

Inside the container, I executed various Linux commands, such as ***ls -ltr***, ***cd ./home***, ***mkdir ./python\_scripts***, and more. I also used ***exit*** to exit the container and return to my computer's OS.

Uploading Python Script: I uploaded the Python script, ***testprint.py***, to the container using the command: ***docker cp Desktop/ENSF607/Assignments/Assignment5/testprint.py ensf607\_alafonye:/home/python\_scripts/testprint.py.***

To verify the script's presence, I re-entered the container with ***docker exec -it ensf607\_alafonye sh,*** navigated to the ***/home/python\_scripts*** directory, and used ***ls -ltr***.

Running the Python Script: I executed the Python script with the command ***python ./testprint.py***, and it produced the expected output: "This is a container test."

Conclusion: I have successfully completed this lab assignment, which involved creating a Docker container, interacting with it, and running a Python script within the container. This exercise has given me a hands-on understanding of containerization and Docker.

A screenshot of a computer program

Description automatically generated