Experiment No. 2.1

Student Name: Ambika Sharma	**UID: 22MCC20022**
Branch: MCA CC & Devops	**Section/Group – 22MCD - 1/A**
Semester: III	**Date of Performance: 29/09/23**
Subject Name: Containerization with Docker	**Subject Code: - 22CAH-742**

1. Aims/Overview for the practical:

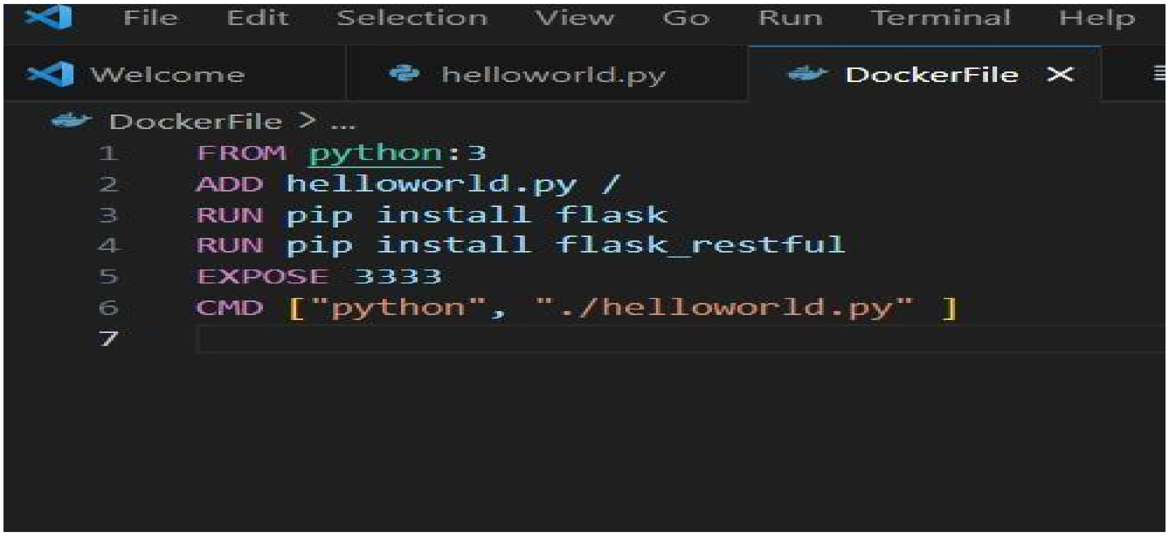
Create one Dockerfile. Using this Dockerfile create and build and image.

2. Steps for experiment/practical:

Step 1: Open the installed **Docker Desktop** in your local machine. Further, you need to sign into it using the credentials given while creating the account on Docker Hub.

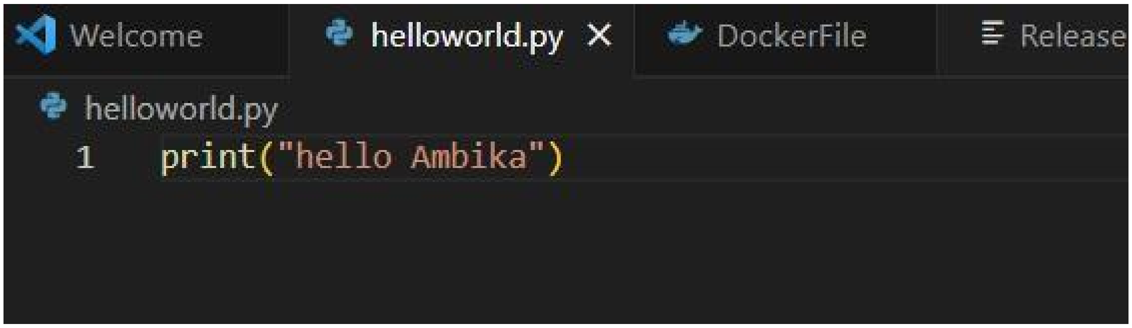
Step 2: Now, you need to open the **Visual Studio Code** while make sure that the Docker Desktop must be ON in background.

Step 3: Now the task is to **create one Dockerfile** and **one python file** in Visual Studio Code. Make sure that both these files need to be created in separate one folder only. While creating the Dockerfile, use the Predefined Instructions/Commands. Must remember that these commands or instructions should be in **CAPITAL FORM.**
The instructions that you can use are:
FROM: For base image. This command should be on top of the Dockerfile.
COPY: This will copy files from local system. We need to provide source and destination.
ADD: It is similar to COPY but, it provided features to download files from internet.
EXPOSE: To expose ports such as port 8080 for tomcat, port 80 for nginx etc.
WORKDIR: To set the working directory for a container.
CMD: Execute commands but during container creation.
ENTRYPOINT: Similar to CMD, but has higher priority over CMD, first commands will be executed by ENTRYPOINT only.
RUN: To execute commands, it will create a layer in image.



```
FROM python:3
ADD helloworld.py /
RUN pip install flask
RUN pip install flask_restful
EXPOSE 3333
CMD ["python", "./helloworld.py" ]
```

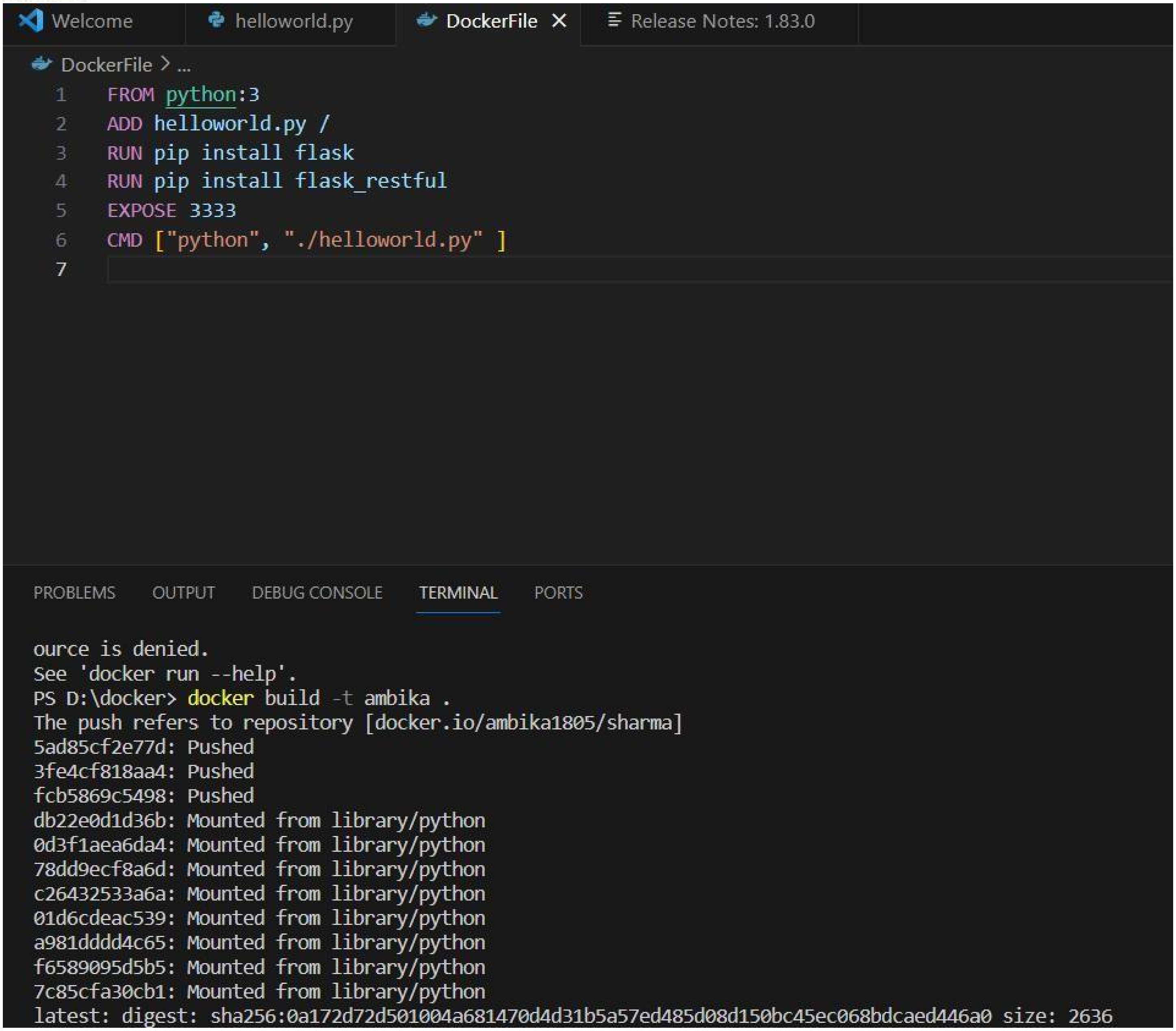
Step 4: In Python file, you have to write all the content that you want to show while running the image that you are creating. And use this python file in Dockerfile while writing the command for CMD.



```
print("hello Ambika")
```

Step 5: After creating both the files (Dockerfile and Python file), you have to open the terminal so that you can write the command of building the image. Make sure the path of building the image is correct otherwise it will throw an error. Path is the location of Dockerfile where all the commands get exeuted to create an image using the commands of Dockerfile.

The command to build an image is: docker build –t imagename .Example:
docker build –t shalini .



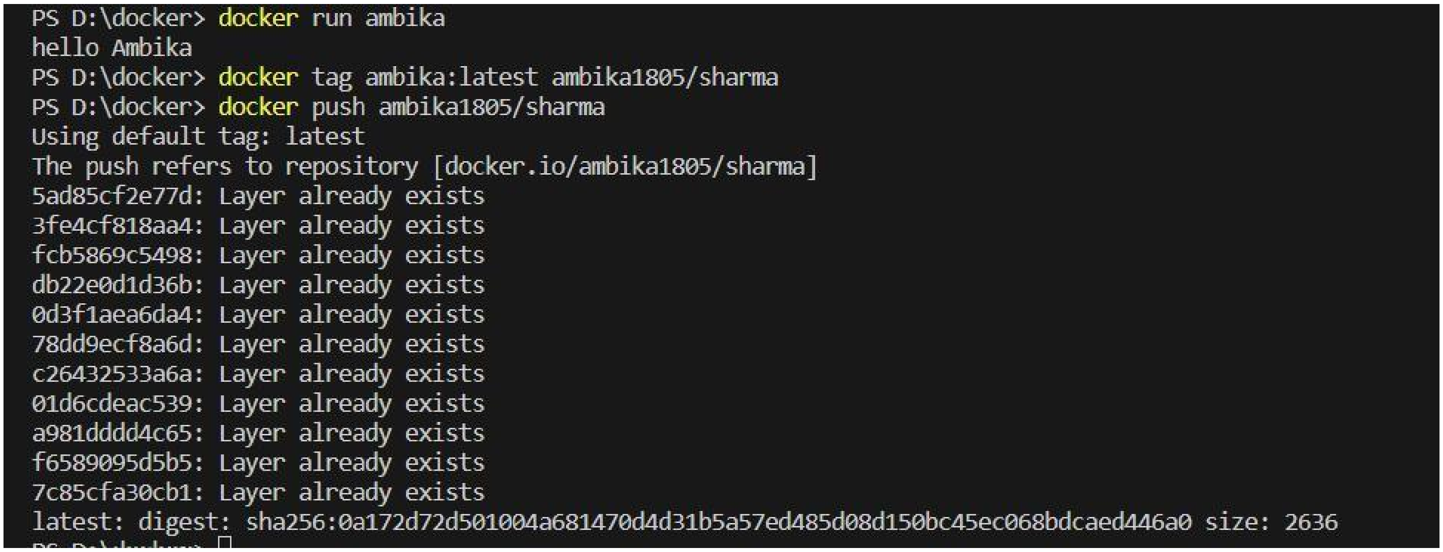
```
FROM python:3
ADD helloworld.py /
RUN pip install flask
RUN pip install flask_restful
EXPOSE 3333
CMD ["python", "./helloworld.py" ]
```

```
ource is denied.
See 'docker run --help'.
PS D:\docker> docker build -t ambika .
The push refers to repository [docker.io/ambika1805/sharma]
5ad85cf2e77d: Pushed
3fe4cf818aa4: Pushed
fcb5869c5498: Pushed
db22e0d1d36b: Mounted from library/python
0d3f1aea6da4: Mounted from library/python
78dd9ecf8a6d: Mounted from library/python
c26432533a6a: Mounted from library/python
01d6cdeac539: Mounted from library/python
a981dddd4c65: Mounted from library/python
f6589095d5b5: Mounted from library/python
7c85cfa30cb1: Mounted from library/python
latest: digest: sha256:0a172d72d501004a681470d4d31b5a57ed485d08d150bc45ec068bdcaed446a0 size: 2636
```

Step 6: If the above command is run successfully, then the image has been created of the given configuration. You can check the created image in Docker Desktop.
Step 7: The command to run an image using CLI is : **docker run imagename**
Example: docker run Shalini

Step 8: Now, the other task is to push this image into Docker Hub. For that you have to give the tag to the image otherwise a default tag is assigned to the image.
Command to give tag to image : docker tag imagename:latest username/tagExample: docker tag shalini:latest shaliniyadav/yadav

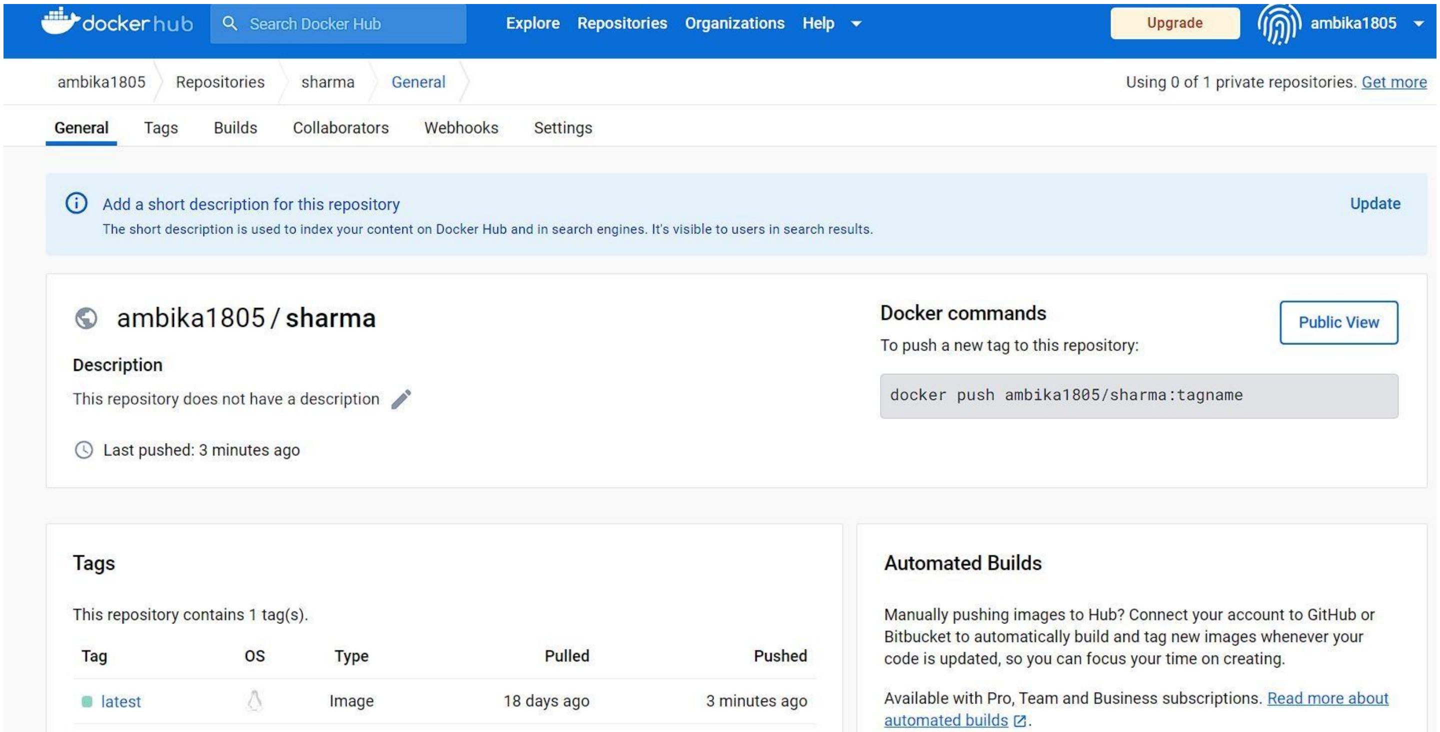
Step 9: After assigning the tag to image, you can push this image to Docker Hub successfully.
Command to push the image: docker push username/tagname
Example: docker push shaliniyadav/Yadav



```
PS D:\docker> docker run ambika
hello Ambika
PS D:\docker> docker tag ambika:latest ambika1805/sharma
PS D:\docker> docker push ambika1805/sharma
Using default tag: latest
The push refers to repository [docker.io/ambika1805/sharma]
5ad85cf2e77d: Layer already exists
3fe4cf818aa4: Layer already exists
fcb5869c5498: Layer already exists
db22e0d1d36b: Layer already exists
0d3f1aea6da4: Layer already exists
78dd9ecf8a6d: Layer already exists
c26432533a6a: Layer already exists
01d6cdeac539: Layer already exists
a981dddd4c65: Layer already exists
f6589095d5b5: Layer already exists
7c85cfa30cb1: Layer already exists
latest: digest: sha256:0a172d72d501004a681470d4d31b5a57ed485d08d150bc45ec068bdcaed446a0 size: 2636
PS D:\docker>
```

Step 10: If successful, you will notice that the image is pushed to the docker hub with the given tagnameonly.

3. Result/Output/Writing Summary



Using 0 of 1 private repositories. Get more

Add a short description for this repository
The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results.

ambika1805 / sharma

Description
This repository does not have a description

Last pushed: 3 minutes ago

Docker commands
To push a new tag to this repository:

```
docker push ambika1805/sharma:tagname
```

Tags
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	18 days ago	3 minutes ago

Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

4. Learning outcomes (What I have learnt): Times new roman 12 size

a) I have learnt to create Dockerfile with proper instructions.

b) I have learnt to build the image from Dockerfile, further learnt to push it into docker hub.