Docker Task3

Date: 16/05/24

1. **Write a note on Dockerfile with usage of its attributes.**

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

A Dockerfile is a script that contains instructions for building a customized docker image. Each instruction in a Dockerfile creates a new layer in the image, and the final image is composed of all the layers stacked on top of each other.

It includes instructions for installing dependencies, copying files, setting environment variables, and configuring the container.

Dockerfile uses a simple, easy-to-read syntax that can be created and edited with any text editor. Once a Dockerfile has been created, it can be used to build an image using the **docker build** command. The resulting image can then be run as a container using the docker run command.

The structure of a Dockerfile is based on a set of simple instructions, such as “FROM”, “RUN”, “COPY”, “ENV”, etc. Each instruction adds a new layer to the image and each layer includes the instructions specified in the previous layer. The final image is a result of all the instructions specified in the Dockerfile.

Dockerfile Attributes with usage:

FROM = To pull base image from dockerhub.

RUN = To execute command.

WORKDIR = To specify working directory.

MAINTAINER = To specify author or owner of file.

EXPOSE = To open specific port.

ENV = To set environment variables.

ADD = Copies files from host to contains downloads files form specific URL & extracts tar.gz or zip files.

ENTRYPOINT = similar to CMD, but having higher priority. Also allows additional arguments to pags.

ARG = Defines variables that passed to container.

LABLE = To add metadata.

USER = To set user.

HEALTHCHECK = To specify path for health check.

SHELL = To specify shell to be used to run command.

STOPSIGNAL = specifies the signal to stop container gracefully.

VOLUME = to create volume.

ON BUILD = Specifies instruction to be used when we we this as base image for other image.

Copy = To copy files from host to image / container.

CMD = execute command during container creation.

2. **What is difference between CMD and ENTRYPOINT?**

CMD	**ENTRYPOINT**
CMD is used to specify the default command to run when a container starts.	**ENTRYPOINT** is used to specify the executable that will run when the container starts.
You can specify the command and its arguments in the **CMD** instruction.	It is often used to define the main application process inside the container.
If a Dockerfile has multiple **CMD** instructions, only the last one will take effect.	The command specified in **ENTRYPOINT** is not overridden by a command specified at runtime. However, arguments specified at runtime will be appended to the **ENTRYPOINT** command.
If a command is specified when running a container (**docker run <image> <command>**), it will override the **CMD** instruction defined in the Dockerfile.	We can still override **ENTRYPOINT** by using the **--entrypoint** flag when running the container.

3. **Write a Dockerfile to run Nodejs application build an image from it and create a container using that image (also include persistent volume and network in Dockerfile).**

- **nano Dockerfile**

FROM node:10-alpine

RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app

WORKDIR /home/node/app

COPY package*.json ./

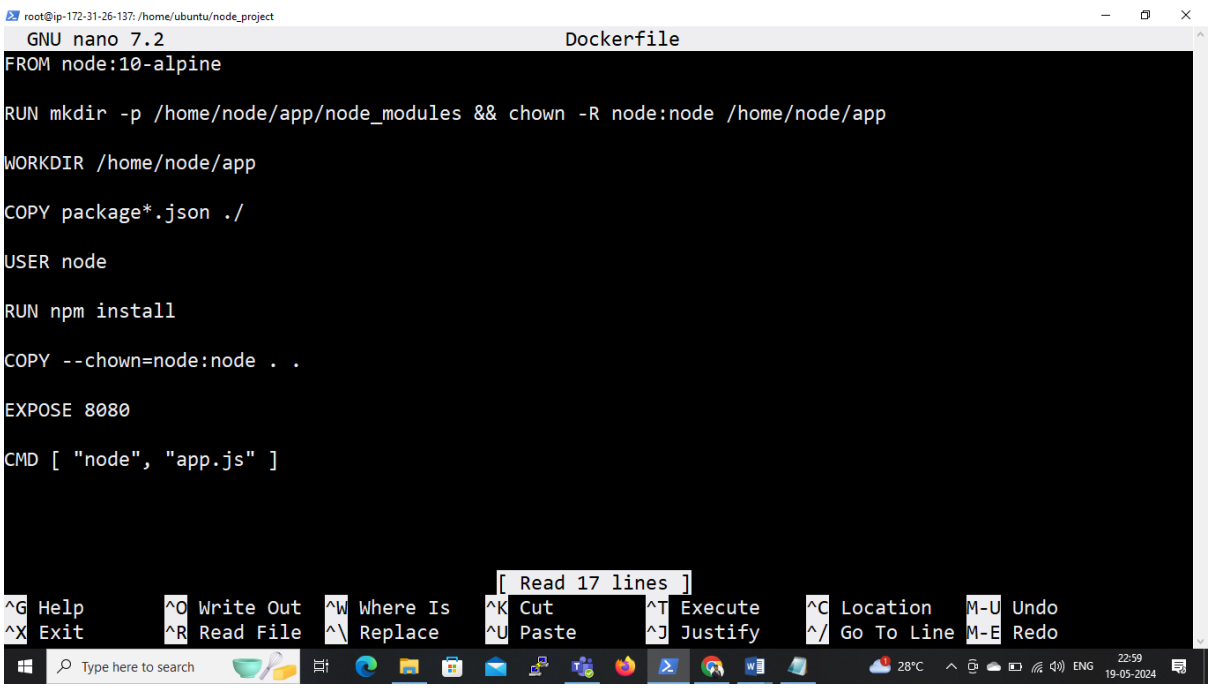
USER node

RUN npm install

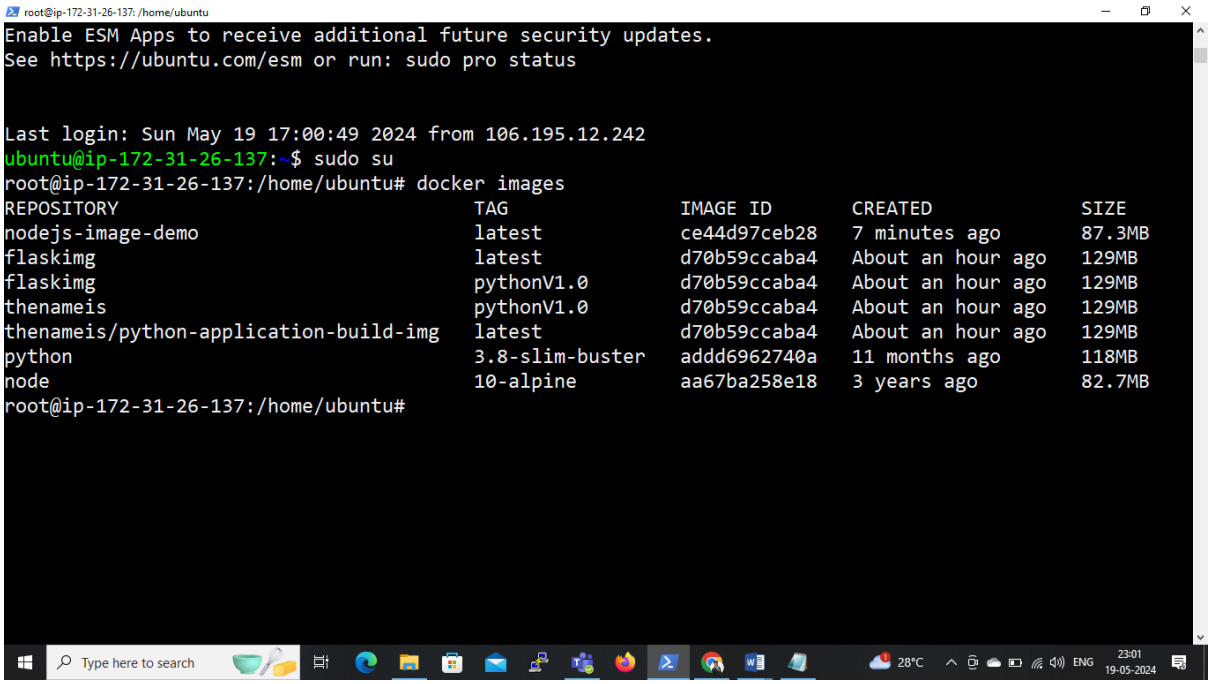
COPY --chown=node:node . .

EXPOSE 8080

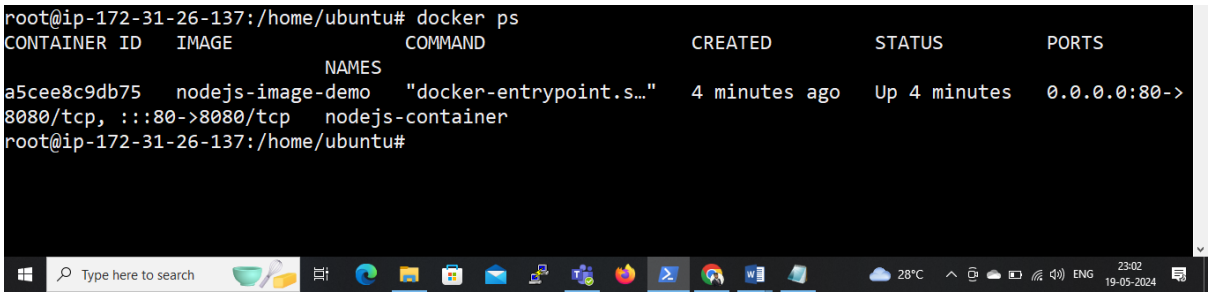
CMD ["node", "app.js"]



- **docker images**


root@ip-172-31-26-137: /home/ubuntu
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Sun May 19 17:00:49 2024 from 106.195.12.242
ubuntu@ip-172-31-26-137:~$ sudo su
root@ip-172-31-26-137:/home/ubuntu# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
nodejs-image-demo latest ce44d97ceb28 7 minutes ago 87.3MB
flaskimg latest d70b59ccaba4 About an hour ago 129MB
flaskimg pythonV1.0 d70b59ccaba4 About an hour ago 129MB
thenameis pythonV1.0 d70b59ccaba4 About an hour ago 129MB
thenameis/python-application-build-img latest d70b59ccaba4 About an hour ago 129MB
python 3.8-slim-buster addd6962740a 11 months ago 118MB
node 10-alpine aa67ba258e18 3 years ago 82.7MB
root@ip-172-31-26-137:/home/ubuntu#
Type here to search
28°C
ENG
23:01
19-05-2024

- **docker ps**


root@ip-172-31-26-137:/home/ubuntu# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
a5cee8c9db75 nodejs-image-demo "docker-entrypoint.s…" 4 minutes ago Up 4 minutes 0.0.0.0:80->
8080/tcp, :::80->8080/tcp nodejs-container
root@ip-172-31-26-137:/home/ubuntu#
Type here to search
28°C
ENG
23:02
19-05-2024

4. **Write a Dockerfile to create a python application build image from it and push that image to private repository of Docker hub.**

   i. **Mkdir my-flask**
   ii. **Cd my-flask**
   iii. **Create a file requirements.txt and write ,**

        flask

   iv. **Create a file app.py and write code,**

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello_geek():

    return '<h1>Hello from Flask & Docker</h2>'

if __name__ == "__main__":

    app.run(debug=True)
```

see below screenshot;

   v. **Create file Dockerfile and write code,**

```
# syntax=docker/dockerfile:1
```

FROM python:3.8-slim-buster

WORKDIR /python-docker

COPY requirements.txt requirements.txt

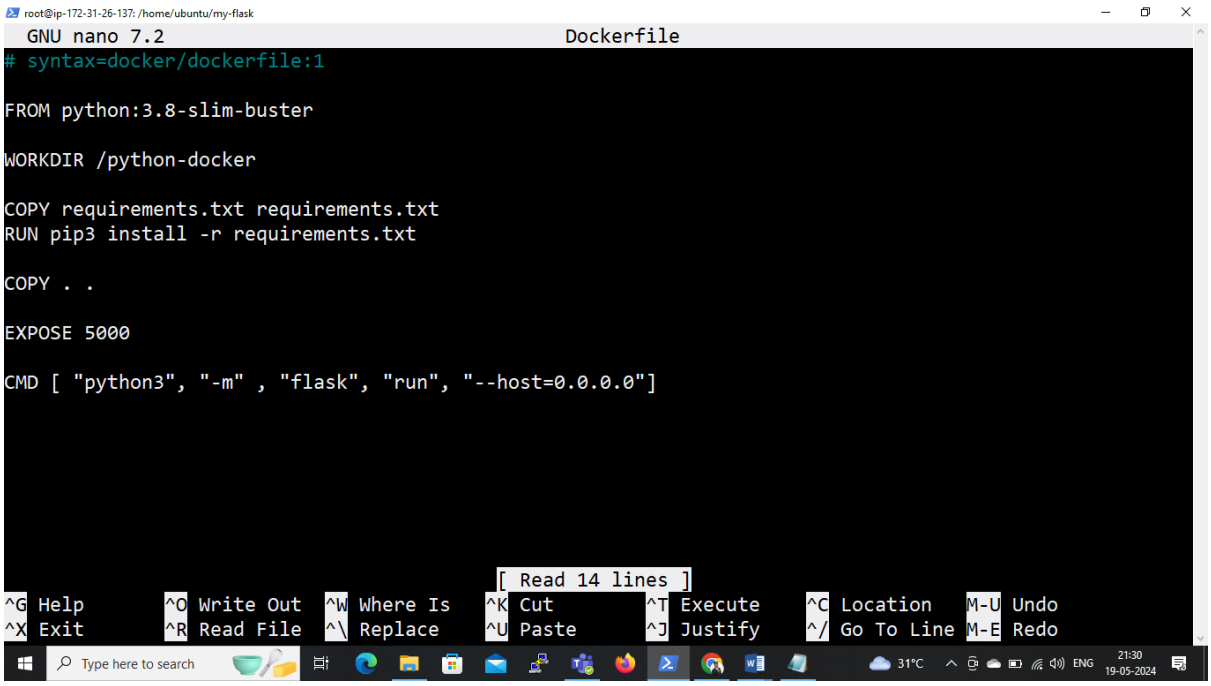
RUN pip3 install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python3", "-m" , "flask", "run", "--host=0.0.0.0"]

see below screenshot;

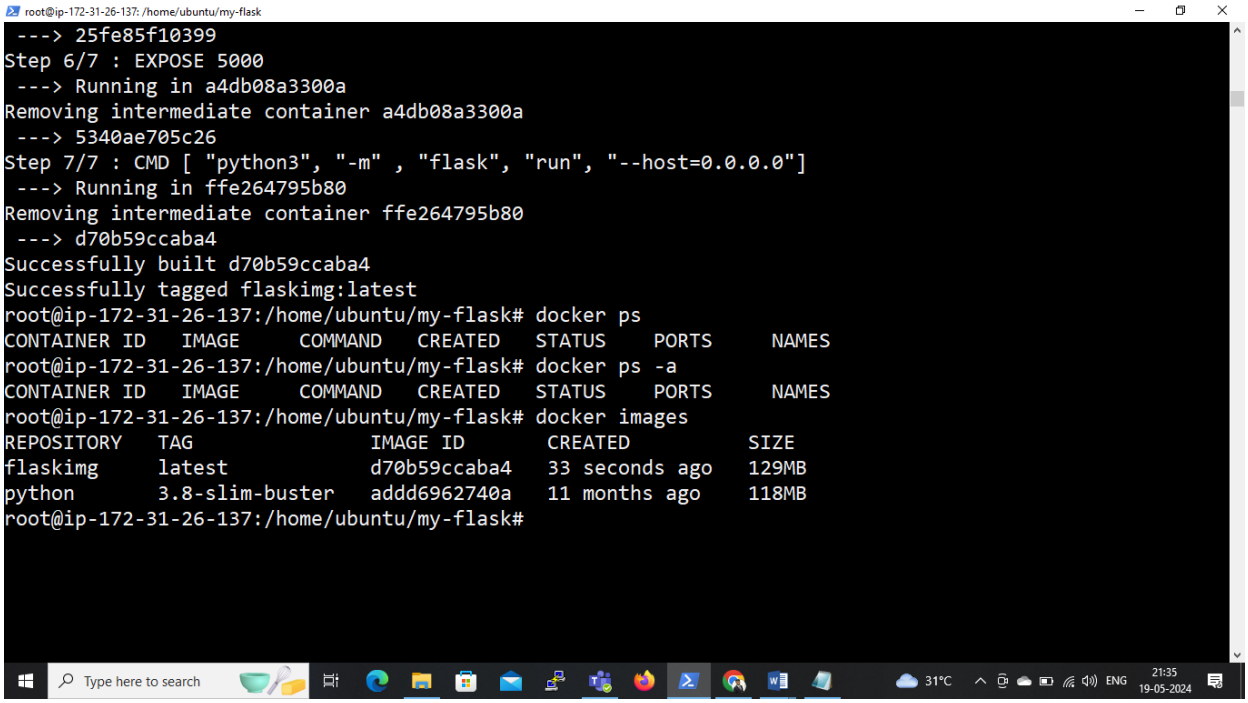


vi. **To build this image using following command;**

- **docker build –t flaskimg .**

vii. **To check python image**

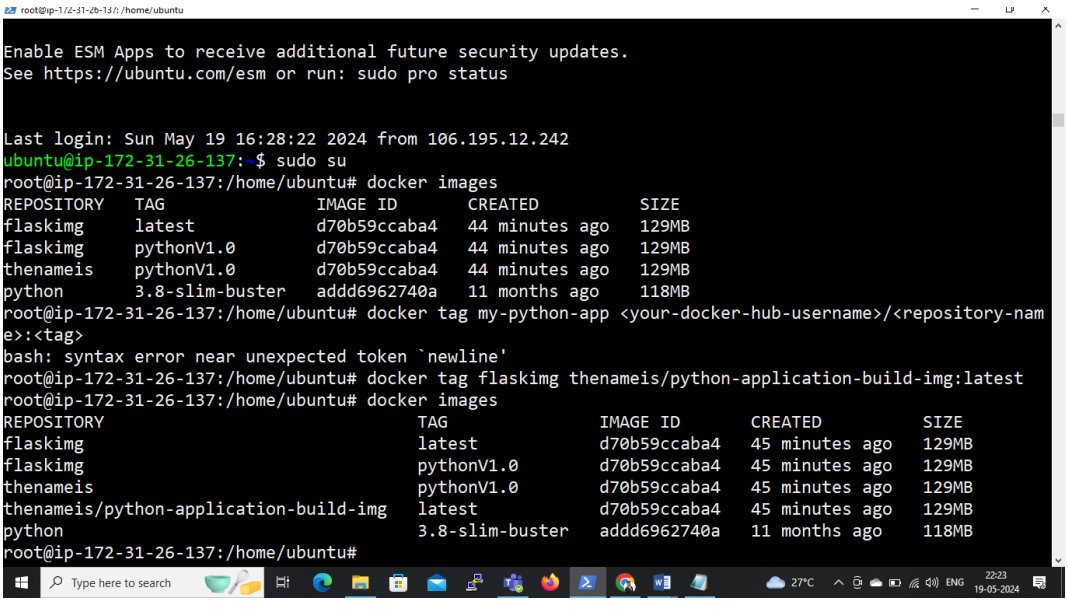
- **docker images**



viii. After the build completes, tag the image with our Docker Hub username and repository name:

docker tag my-python-app <your-docker-hub-username>/<repository-name>:<tag>

- **eg. docker tag flaskimg thenameis/python-application-build-img:latest**



ix. Log in to Docker Hub using the following command and enter your credentials:

- **docker login**

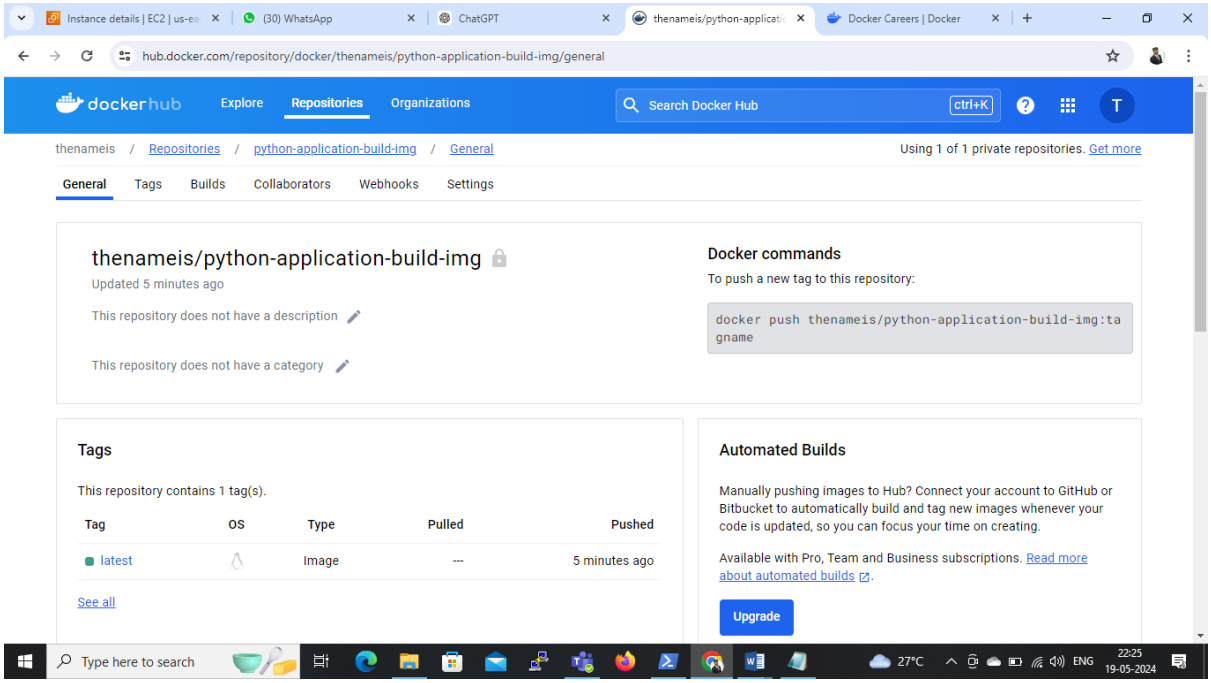
  Username: thenameis

  Password: access token (my account – security – generate new access token)

x. **Finally, push the tagged image to your private repository on Docker Hub:**

docker push thenameis/python-application-build-img:tagname

- **eg. docker push thenameis/python-application-build-img:latest**

```
root@ip-172-31-26-137: /home/ubuntu
root@ip-172-31-26-137:/home/ubuntu# docker tag my-python-app <your-docker-hub-username>/<repository-nam
e>:<tag>
bash: syntax error near unexpected token `newline'
root@ip-172-31-26-137:/home/ubuntu# docker tag flaskimg thenameis/python-application-build-img:latest
root@ip-172-31-26-137:/home/ubuntu# docker images
REPOSITORY                                TAG              IMAGE ID       CREATED          SIZE
flaskimg                                  latest           d70b59ccaba4   45 minutes ago   129MB
flaskimg                                  pythonV1.0       d70b59ccaba4   45 minutes ago   129MB
thenameis                                 pythonV1.0       d70b59ccaba4   45 minutes ago   129MB
thenameis/python-application-build-img    latest           d70b59ccaba4   45 minutes ago   129MB
python                                    3.8-slim-buster  addd6962740a   11 months ago    118MB
root@ip-172-31-26-137:/home/ubuntu#
root@ip-172-31-26-137:/home/ubuntu# docker push thenameis/python-application-build-img:latest
The push refers to repository [docker.io/thenameis/python-application-build-img]
909c309c13e9: Pushed
810f4e880c0b: Pushed
67d69523ef41: Pushed
72e0e305c79a: Pushed
e6c5004ee77f: Pushed
997b8e79e84f: Pushed
3054512b6f71: Pushed
ae2d55769c5e: Pushed
e2ef8a51359d: Pushed
latest: digest: sha256:afcfe1f86475ee0a7a1fc3265af0cc03f03c6b807b8538d4e0ef1236f85be0d5 size: 2202
root@ip-172-31-26-137:/home/ubuntu#
```

Also check on docker hub: