**Practical No: 4**

**Aim:** Building and pushing a docker image.

**Implementation:**

Step 1 – Install Docker Desktop and sign up in Docker Hub

Step 2- Log in with credentials in Docker Desktop and start Docker engine

Step 3 – Open the folder with the application code.

The code (application) we need to create an image of is a basic flask application

**Flaskdoc.py**

from flask import Flask

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

@app.route("/")

def display():

  return f"<h3>Hello Akansha:)</h3>"

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

  app.run()

Step 4 – create a Dockerfile with all configuration details and an additional requirements text file

**Dockerfile**

FROM python:3.11.6

WORKDIR /app

COPY . /app

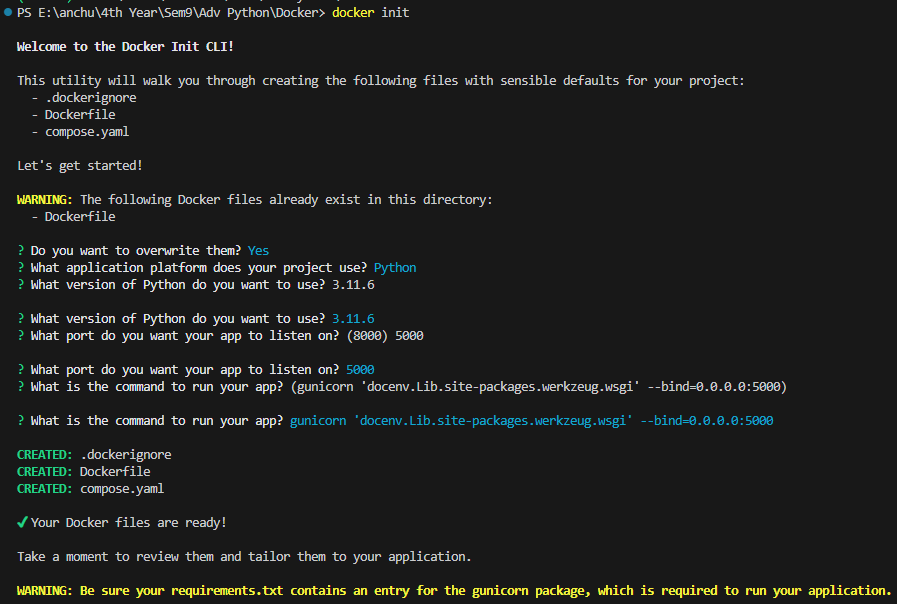
RUN pip install -r requirements.txt

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

**Requirements.txt**

flask==3.0.0

Step 5 – use the ***docker init*** command to initialize a project with the files necessary to run the project in a container.

****

Step 6 – use the ***docker compose up –build*** command:

The following only builds the images, does not start the containers:

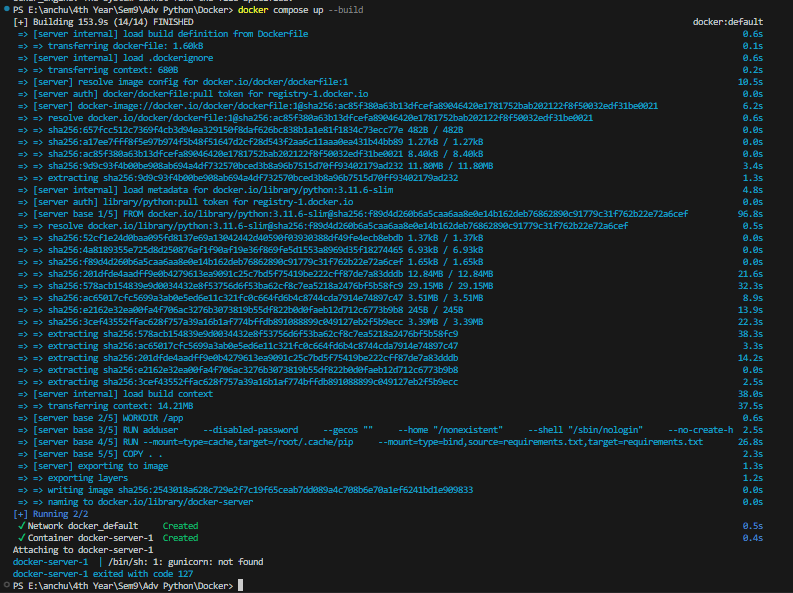
* docker-compose build

The following builds the images if the images do not exist and starts the containers:

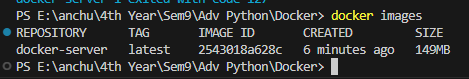
* docker-compose up

If you add the --build option, it is forced to build the images even when not needed:

* docker-compose up --build



The docker image is built and visualized with ***docker images*** command



It image has been composed in the docker desktop

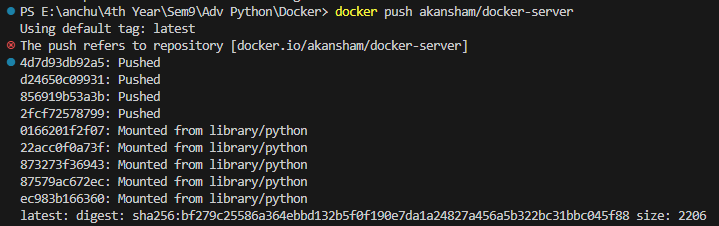


Step 7: Login to docker using ***docker login***

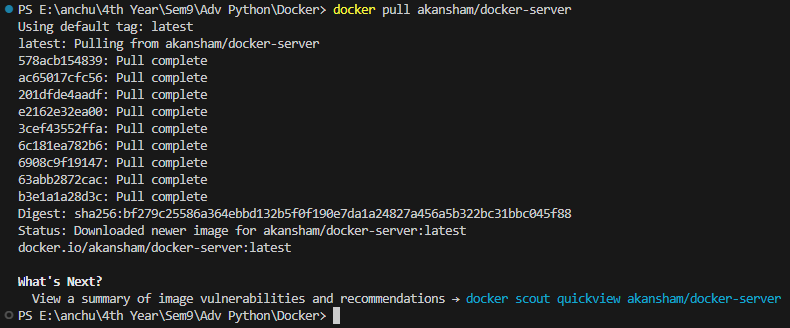
******

Step 8: use the command ***docker tag image\_name DOCKER\_HUB\_NAME/image\_name*** to tag the image which is a custom, human-readable manifest identifier that is typically a specific version or variant of an image.



Step 9: use the command ***docker push DOCKER\_HUB\_NAME/image\_name*** to share your images to the Docker Hub registry or to a self-hosted one. 

Step 10: use the command ***docker pull DOCKER\_HUB\_NAME/image\_name*** to pull your images from the Docker Hub registry to the local repository.

****

**B) Develop docker image for python script and store it on docker registry**

1. Use the ***docker build -t image\_name*** command – used to build a Docker image from a Dockerfile

FROM python:3.9.10-slim-buster

WORKDIR /usr/src/app

ENV VIRTUAL\_ENV=/opt/venv

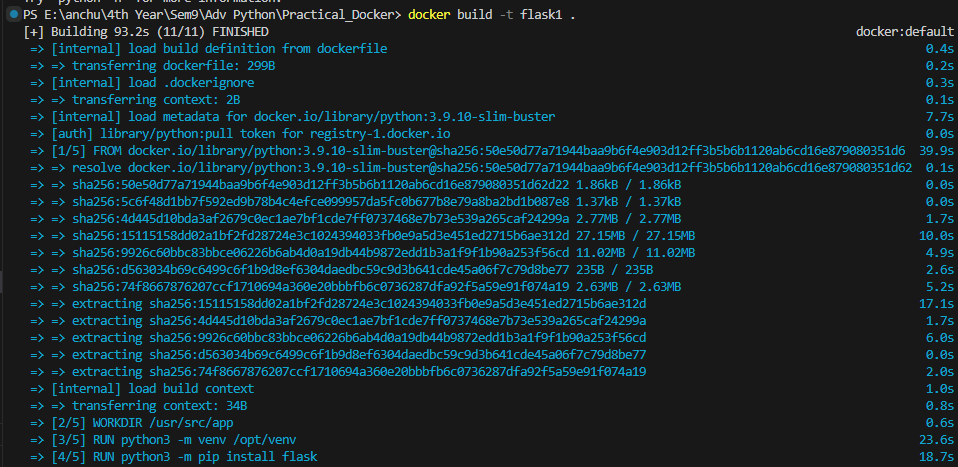
RUN python3 -m venv $VIRTUAL\_ENV

ENV PATH="$VIRTUAL\_ENV/bin:$PATH"

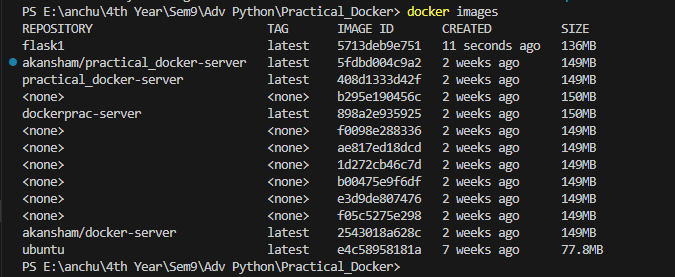
RUN python3 -m pip install flask

COPY basicflask.py .

ENTRYPOINT FLASK\_APP=basicflask flask run --host=0.0.0.0

****

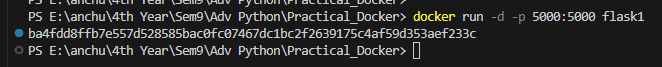
1. Use *docker images* command - used to list all the Docker images that are currently stored on the local system.

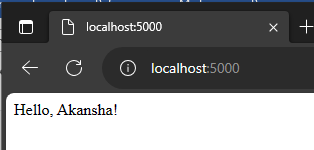


1. Use ***docker run -d -p 5000:5000 image\_name*** command - used to run a Docker container in detached mode (-d) and map a port from the host to a port in the container (-p).

-d runs the container in the background, freeing up the terminal for other commands.

-p specifies the port mapping, with the format <host\_port>:<container\_port>---it maps port 5000 from the host to port 5000 in the container.

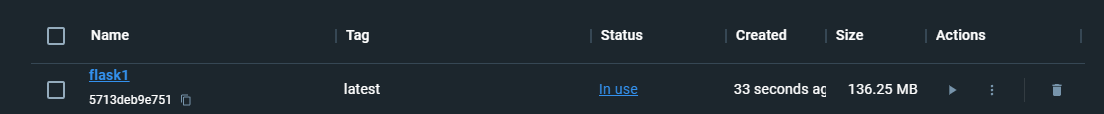
****

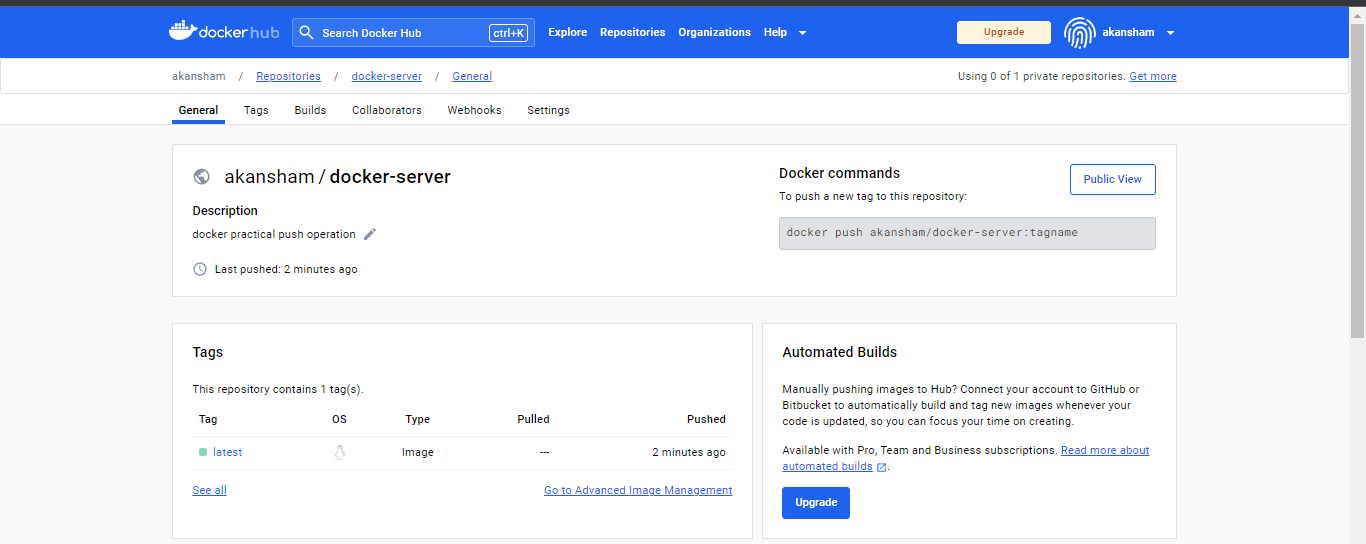


Flask file output after running the browser with <http://localhost:5000>

**Output:**

The docker image built and pushed to the Docker Desktop repository:





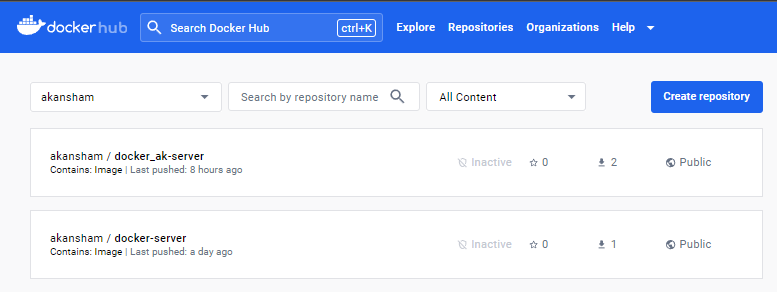
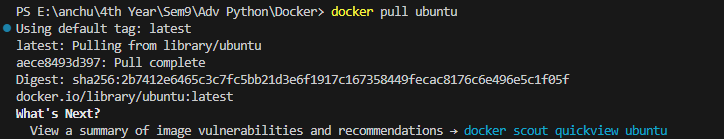
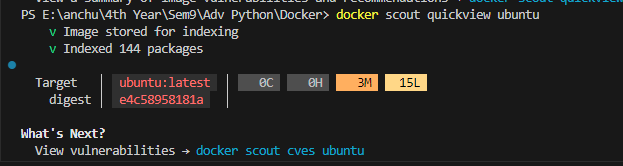


Image seen on the docker hub after the push command

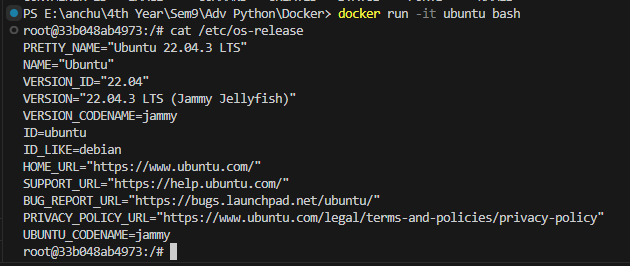
1. **To pull ubuntu image from docker registry**
2. Use ***docker pull ubuntu*** command – it pulls the latest version of ubuntu image

****

1. Use ***docker scout quickview ubuntu*** command – displays a quick overview of an image

****

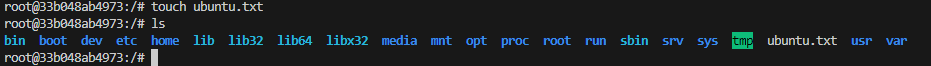
1. Use ***docker run -it ubuntu bash*** – to start an interactive Docker container based on the Ubuntu operating system image



1. Use ***echo “text to print”*** command - used to print text or display a message on the standard output (usually the terminal or command line).

****

1. Use ***touch filename.txt*** command to create an empty file in the current directory

****

1. Use ***exit*** command to exit from the ubuntu bash script

****