

Name: Vivek Saroj

Reg Email: viveksaroj098@gmail.com

Course Name: DevOps

Assignment Name: Module 36 Theoretical Overview of Docker and Containerization Assignment.

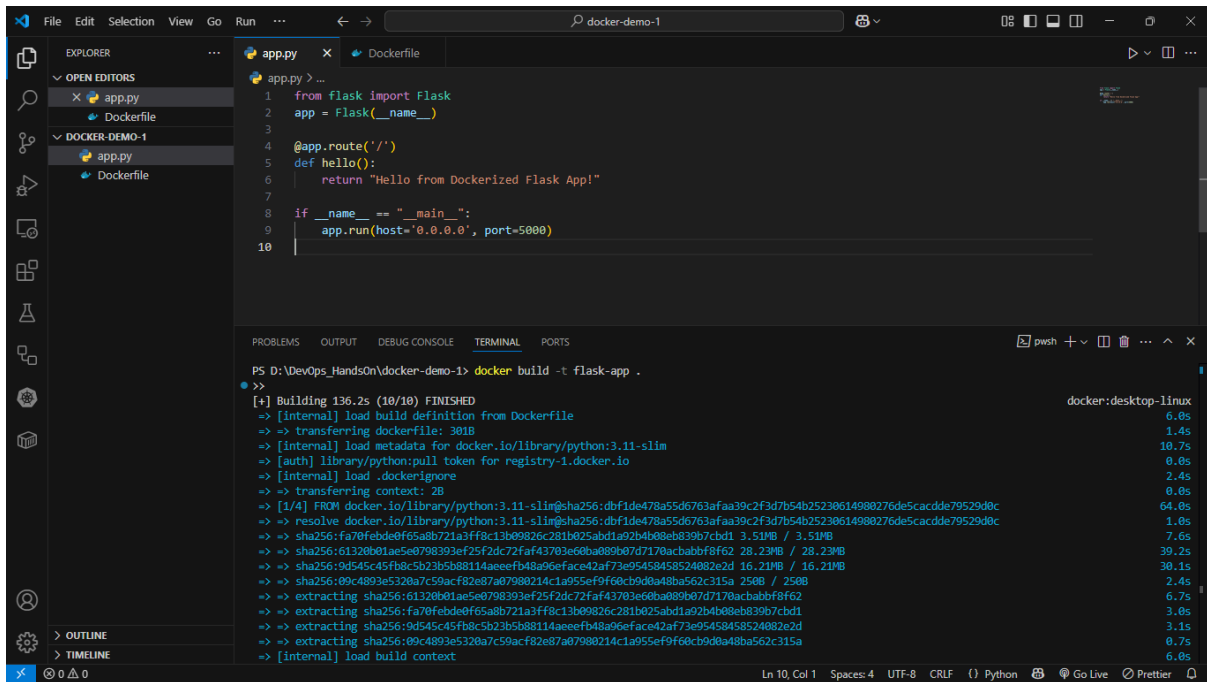
Task:

Q) Install Docker and create Dockerfiles to containerize applications?

Ans:

For Windows:

1. Go to: <https://www.docker.com/products/docker-desktop/>
2. Download and install **Docker Desktop**.
3. During installation:
 - Enable WSL 2 backend (recommended)
 - Restart your system if prompted
4. After installation, run this to verify:
 - docker version
 - docker info

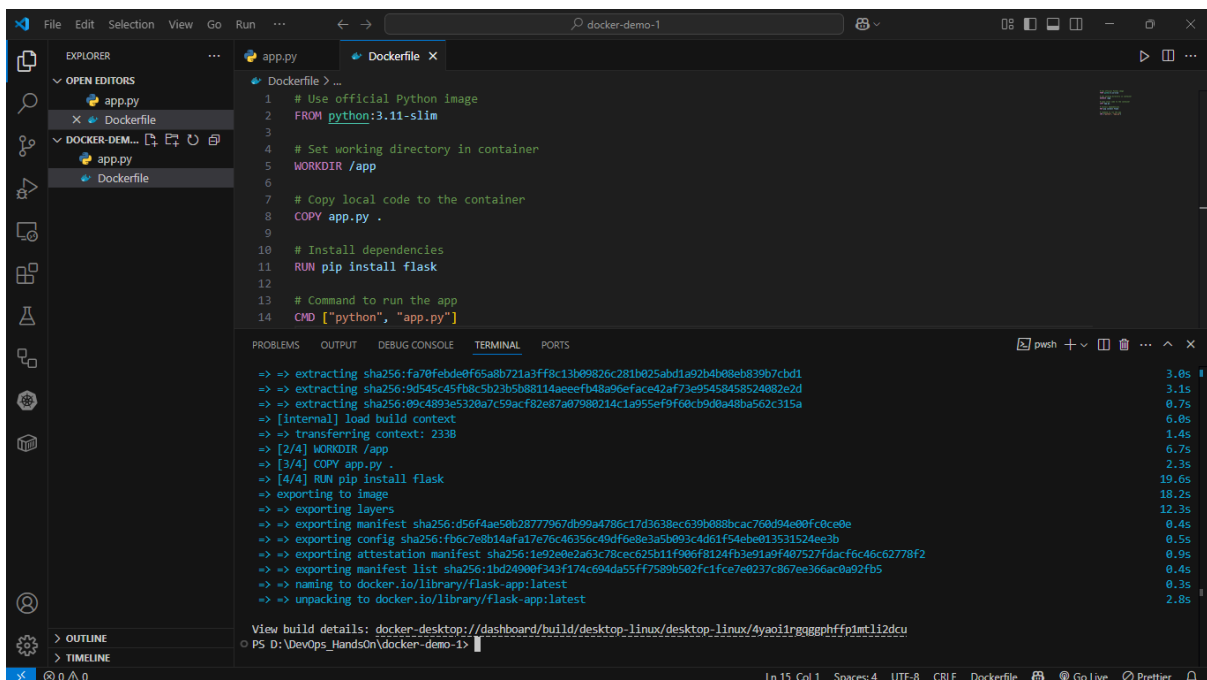


The screenshot shows the VS Code interface with a Dockerfile open in the editor. The Dockerfile contains the following code:

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello():
6     return "Hello from Dockerized Flask App!"
7
8 if __name__ == "__main__":
9     app.run(host="0.0.0.0", port=5000)
10
```

The terminal output shows the command `docker build -t flask-app .` being executed, resulting in a successful build of the `flask-app` image.

Command: `docker build -t flask-app` ## here flask-app is image name.



The screenshot shows the VS Code interface with a Dockerfile open in the editor. The Dockerfile contains the following code:

```
1 # Use official Python image
2 FROM python:3.11-slim
3
4 # Set working directory in container
5 WORKDIR /app
6
7 # Copy local code to the container
8 COPY app.py .
9
10 # Install dependencies
11 RUN pip install flask
12
13 # Command to run the app
14 CMD ["python", "app.py"]
```

The terminal output shows the command `docker build -t flask-app .` being executed, resulting in a successful build of the `flask-app` image.

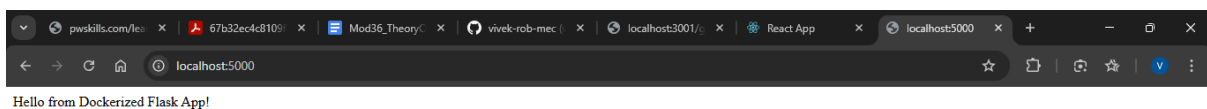
Command: `docker run -d -p 5000:5000 flask-app`

The screenshot shows the VS Code interface with a file explorer on the left showing 'app.py' and 'Dockerfile'. The main editor displays the contents of 'app.py', which is a simple Flask application. Below the editor, the 'TERMINAL' panel shows the output of a Docker build command. The build process includes exporting layers, manifest, config, and attestation, and then naming the image 'docker.io/library/flask-app:latest'. The terminal also shows the command to run the container: 'docker run -d -p 5000:5000 flask-app'. The output of the container is visible in the browser window below.

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello():
6     return "Hello from Dockerized Flask App!"
7
8 if __name__ == "__main__":
9     app.run(host="0.0.0.0", port=5000)
10
```

```
>>> exporting layers 12.3s
>>> exporting manifest sha256:d56f4ae58b28777967db99a4786c17d3638ec639b088bcac760d94e00fc0ce0e 0.4s
>>> exporting config sha256:fb6c7e8b14afa17676c46356c49df6e8e3a5b093c4d61f54ebe013531524ee3b 0.5s
>>> exporting attestation manifest sha256:1e92e0e2a63c78cec625b11f906f8124fb3e91a9f487527fdacfc6c62778f2 0.9s
>>> exporting manifest list sha256:1bd24900f343f174c694da55ff7589b502fc1fce7e0237c867ee366ac0a92fb5 0.4s
>>> naming to docker.io/library/flask-app:latest 0.3s
>>> unpacking to docker.io/library/flask-app:latest 2.8s

View build details: docker-desktop:///dashboard/build/desktop-linux/desktop-linux/4yaoi1rgggghffp1mtli2dcu
PS D:\DevOps_HandsOn\docker-demo-1> docker run -d -p 5000:5000 flask-app
5ad6641533a3ce545f085326901c3787de214fa0197d553196b08376b47e8f2a
PS D:\DevOps_HandsOn\docker-demo-1>
```



List container running:

The screenshot shows a terminal window with the output of the 'docker ps' command. The output lists the running container, its image, command, created time, status, ports, and name.

```
View build details: docker-desktop:///dashboard/build/desktop-linux/desktop-linux/4yaoi1rgggghffp1mtli2dcu
PS D:\DevOps_HandsOn\docker-demo-1> docker run -d -p 5000:5000 flask-app
>>>
5ad6641533a3ce545f085326901c3787de214fa0197d553196b08376b47e8f2a
PS D:\DevOps_HandsOn\docker-demo-1> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
5ad6641533a3   flask-app "python app.py"         2 minutes ago Up 2 minutes   0.0.0.0:5000->5000/tcp            tender_bell
PS D:\DevOps_HandsOn\docker-demo-1>
```

Stop, remove container and delete image:

File Edit Selection View Go Run ... docker-demo-1

EXPLORER

OPEN EDITORS

- app.py
- Dockerfile

DOCKER-DEMO-1

- app.py
- Dockerfile

app.py

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello():
6     return "Hello from Dockerized Flask App!"
7
8 if __name__ == "__main__":
9     app.run(host='0.0.0.0', port=5000)
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

pwsh + v ... ^ X

```
-> -> naming to docker.io/library/flask-app:latest 0.3s
-> -> unpacking to docker.io/library/flask-app:latest 2.8s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/4yaoi1rgggphffp1mtli2dcu
PS D:\DevOps_HandsOn\docker-demo-1> docker run -d -p 5000:5000 flask-app
5ad6641533a3ce545f885326901c3787de214fa0197d553196b08376b47e8f2a
PS D:\DevOps_HandsOn\docker-demo-1> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
5ad6641533a3   flask-app "python app.py"         2 minutes ago Up 2 minutes   0.0.0.0:5000->5000/tcp             tender_bell
PS D:\DevOps_HandsOn\docker-demo-1> docker stop 5ad6641533a3
5ad6641533a3
PS D:\DevOps_HandsOn\docker-demo-1> docker rm 5ad6641533a3
5ad6641533a3
PS D:\DevOps_HandsOn\docker-demo-1> docker rmi flask-app
Error response from daemon: No such image: flask-app:latest
PS D:\DevOps_HandsOn\docker-demo-1> docker rmi flask-app
Untagged: flask-app:latest
Deleted: sha256:1bd24900f343f174c694da55ff7589b502fc1fce7e0237c867ee366ac0a92fb5
PS D:\DevOps_HandsOn\docker-demo-1>
```

Ln 10, Col 1 Spaces: 4 UTF-8 CRLF Python Go Live Prettier