



UNIVERSITY OF PETROLEUM AND ENERGY
STUDIES
DEHRADUN

DISTRIBUTED SYSTEMS

Docker - Containerization

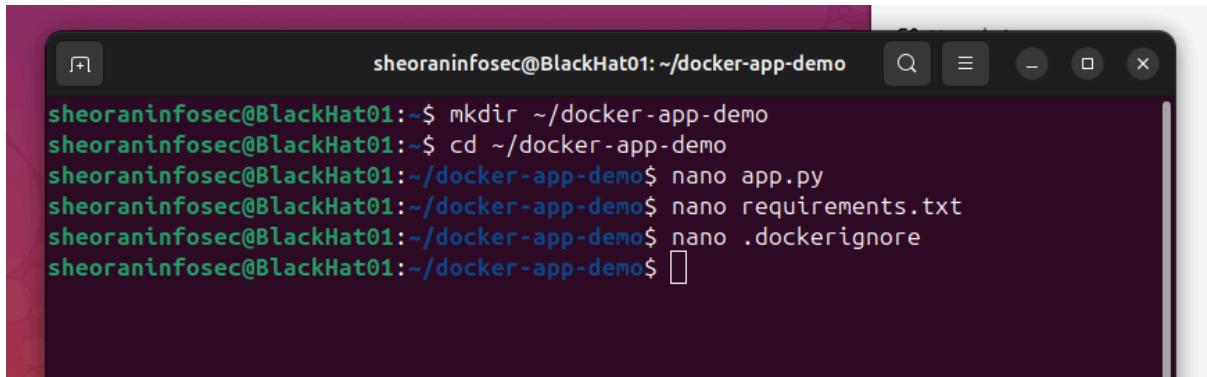
MTECH-COMPUTER SCIENCE
ENGINEERING
CYBER SECURITY AND FORENSICS

Name: Jigesh Sheoran
SAP ID: 590025428

Objective — implement containerization by deploying applications in Docker containers. Students will create Docker images and run containers to observe how virtualization and resource management are achieved in cloud computing environments

Step 1: Create Project Folder

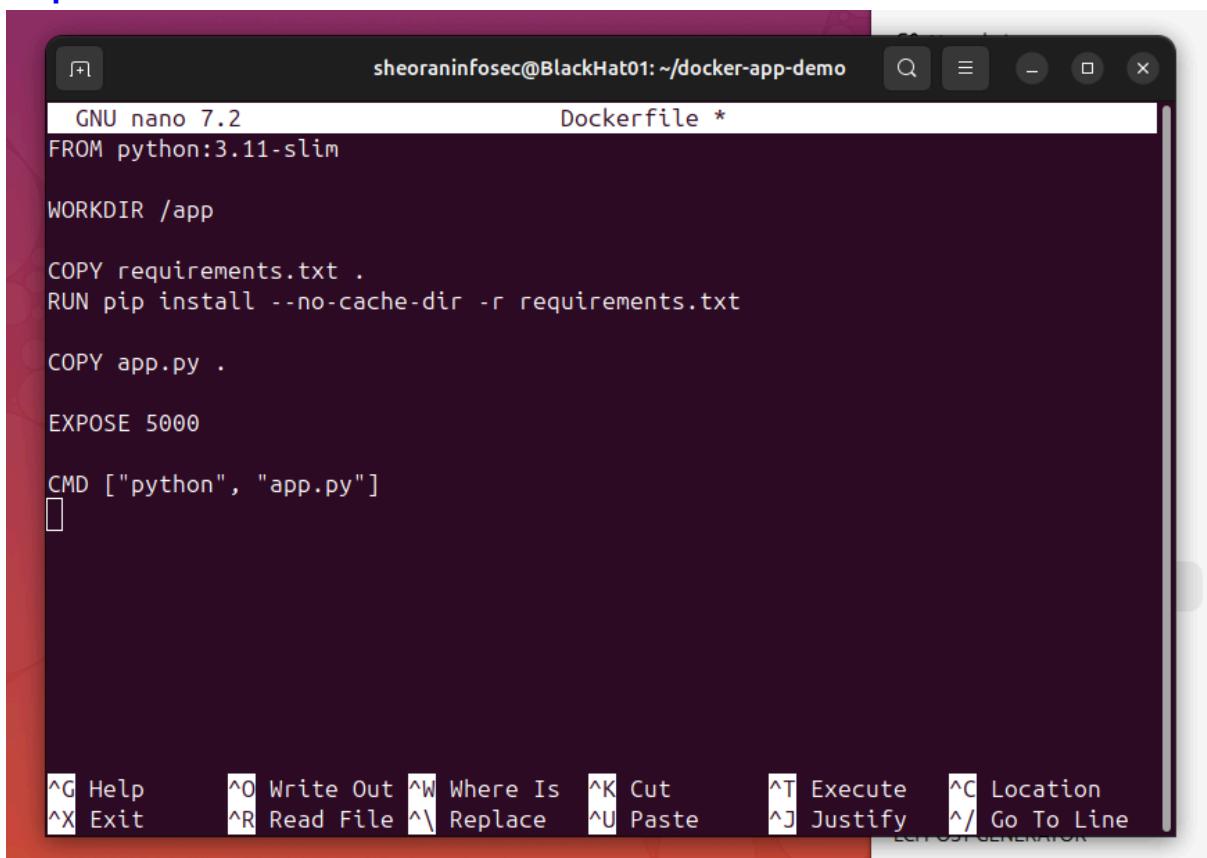
Step 2: write nano [app.py](#) , requirement.txt and i also have written .dockerignore as a good coding practice.



A terminal window titled "sheoraninfosec@BlackHat01: ~/docker-app-demo". The user has run several commands to create a project directory and open nano editors for app.py, requirements.txt, and .dockerignore files.

```
sheoraninfosec@BlackHat01:~$ mkdir ~/docker-app-demo
sheoraninfosec@BlackHat01:~$ cd ~/docker-app-demo
sheoraninfosec@BlackHat01:~/docker-app-demo$ nano app.py
sheoraninfosec@BlackHat01:~/docker-app-demo$ nano requirements.txt
sheoraninfosec@BlackHat01:~/docker-app-demo$ nano .dockerignore
sheoraninfosec@BlackHat01:~/docker-app-demo$
```

Step 3: Create the DockerFile



A terminal window titled "sheoraninfosec@BlackHat01: ~/docker-app-demo". The user is editing a Dockerfile using the nano editor.

```
GNU nano 7.2                               Dockerfile *
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 5000

CMD ["python", "app.py"]
```

The nano editor status bar at the bottom shows various keyboard shortcuts:

- ^G Help
- ^O Write Out
- ^W Where Is
- ^K Cut
- ^T Execute
- ^C Location
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Paste
- ^J Justify
- ^L Go To Line

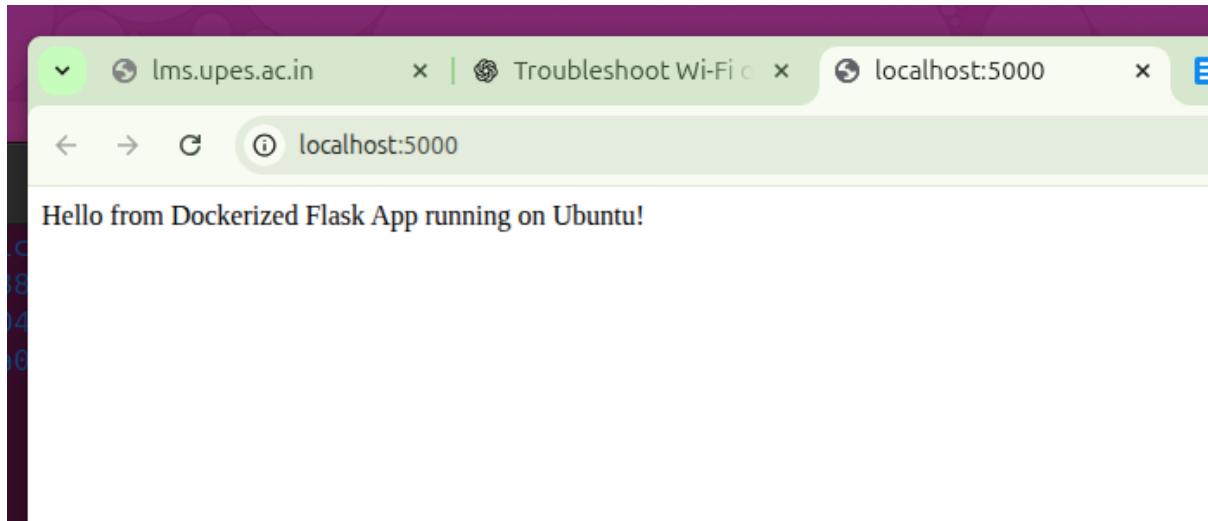
Step 4: Build the Docker image

```
sheoraninfosec@BlackHat01:~/docker-app-demo$ nano .dockerrcignore
sheoraninfosec@BlackHat01:~/docker-app-demo$ nano Dockerfile
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker build -t flask-demo:1.0 .
[+] Building 60.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 205B
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [internal] load .dockerrcignore
=> => transferring context: 58B
=> [1/5] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fca
=> => sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da 251B / 251B
=> => sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60c55 14.36MB / 14.36MB
=> => sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974 1.29MB / 1.29MB
=> => sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 29.78MB / 29.78MB
=> => extracting sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db
=> => extracting sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974
=> => extracting sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60c55
=> => extracting sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da
=> [internal] load build context
=> => transferring context: 299B
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY app.py .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:2d7083ddacfec58773c2a40c5f390d45c56820884fec04b96bf6a82daf7f525b
=> => exporting config sha256:56ffa8a6e56f7c39f1b967d6d7e6783823f1a31183f1a32a0abb2e8fe71f4e5e
=> => exporting attestation manifest sha256:f7a8649fd98dfde943cc59f847e4f7ce001155e998db42185e3bb8341d
=> => exporting manifest list sha256:f63d1eac9cbd9548a77dbabe7f562932f8d0c989df173a4cfce41c8879cc575ac
=> => naming to docker.io/library/flask-demo:1.0
=> => unpacking to docker.io/library/flask-demo:1.0
sheoraninfosec@BlackHat01:~/docker-app-demo$ 
```

Step 5: Run the Container

```
=> => unpacking to docker.io/library/flask-demo:1.0
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker run -d --name flask1 -p 5000:5000 flask-demo:1.0
088648fda48abe1928de708bf9bd62069a5321f6729c82ca7d61b1ff250f8cef
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
088648fda48a flask-demo:1.0 "python app.py" 10 seconds ago Up 10 seconds 0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp flask1
sheoraninfosec@BlackHat01:~/docker-app-demo$ 
```

Step 6: Check if its running or not



Step 7: Monitor Resource Usage

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
088648fda48a	flask1	0.01%	22.6MiB / 15.46GiB	0.14%	7.8kB / 2.31kB	0B / 147kB	2

Step 8: Show detailed container info.

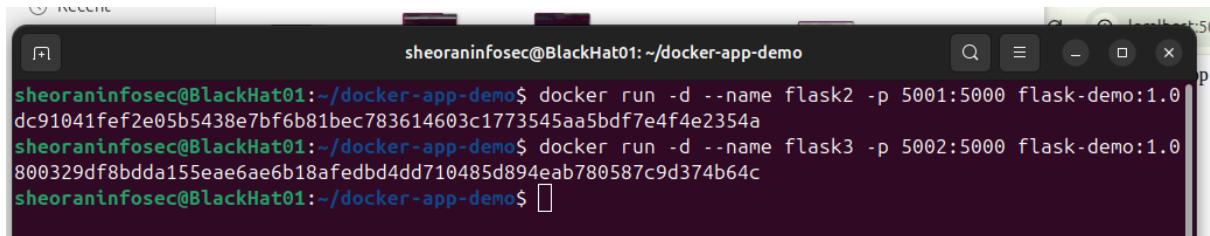
```
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker inspect flask1
[{"Id": "088648fda48abe1928de708bf9bd62069a5321f6729c82ca7d61b1ff250f8cef", "Created": "2025-11-26T15:39:03.196868877Z", "Path": "python", "Args": ["app.py"], "State": {"Status": "running", "Running": true, "Paused": false, "Restarting": false, "OOMKilled": false, "Dead": false, "Pid": 9897, "ExitCode": 0, "Error": "", "StartedAt": "2025-11-26T15:39:03.255629997Z", "FinishedAt": "0001-01-01T00:00:00Z"}, "Image": "sha256:f63d1eac9cbd9548a77dbabe7f562932f8d0c989df173a4cf41c8879cc575ac", "ResolvConfPath": "/var/lib/docker/containers/088648fda48abe1928de708bf9bd62069a5321f6729c82ca7d61b1ff250f8cef/resolv.conf", "HostnamePath": "/var/lib/docker/containers/088648fda48abe1928de708bf9bd62069a5321f6729c82ca7d61b1ff250f8cef/hostname", "HostsPath": "/var/lib/docker/containers/088648fda48abe1928de708bf9bd62069a5321f6729c82ca7d61b1ff250f8cef/hosts"}]
```

Step 9: Scaling from 1 container to multiple containers.

In this case i have created two more container at

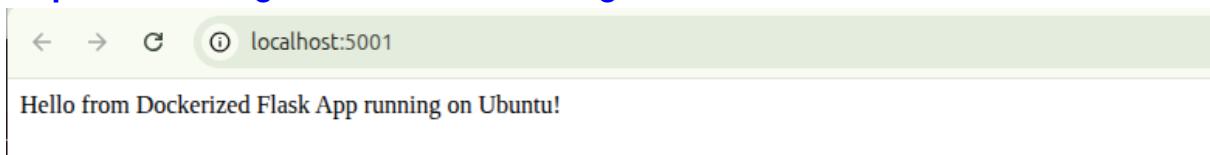
localhost:5001

localhost:5002



```
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker run -d --name flask2 -p 5001:5000 flask-demo:1.0
dc91041fef2e05b5438e7bf6b81bec783614603c1773545aa5bdf7e4f4e2354a
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker run -d --name flask3 -p 5002:5000 flask-demo:1.0
800329df8bdda155eae6ae6b18afedbd4dd710485d894eab780587c9d374b64c
sheoraninfosec@BlackHat01:~/docker-app-demo$
```

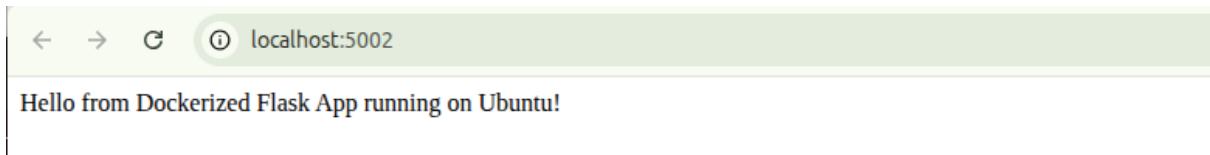
Step 10: checking the cointainer running at localhost:5001



localhost:5001

Hello from Dockerized Flask App running on Ubuntu!

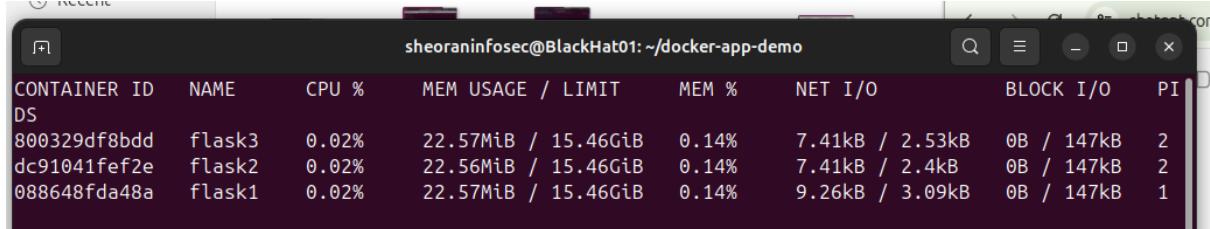
Step 11: checking the cointainer running at localhost:5002



localhost:5002

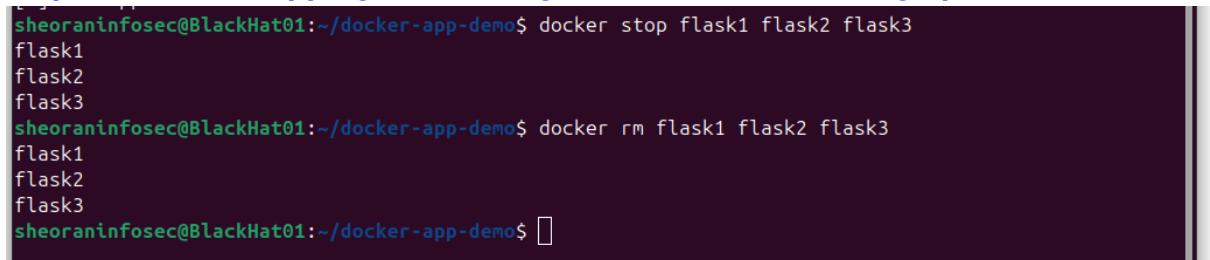
Hello from Dockerized Flask App running on Ubuntu!

Step 12: checking resource usage and stats



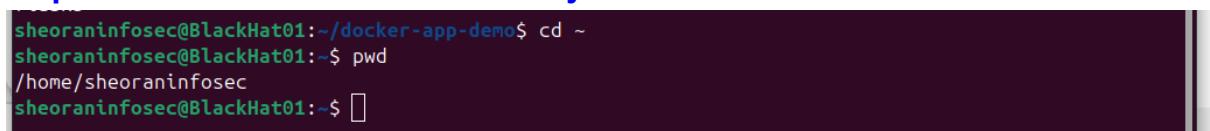
CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PI
800329df8bdd	flask3	0.02%	22.57MiB / 15.4GiB	0.14%	7.41kB / 2.53kB	0B / 147kB	2
dc91041fef2e	flask2	0.02%	22.56MiB / 15.4GiB	0.14%	7.41kB / 2.4kB	0B / 147kB	2
088648fda48a	flask1	0.02%	22.57MiB / 15.4GiB	0.14%	9.26kB / 3.09kB	0B / 147kB	1

Step 13: At last stopping the running container and cleaning up.



```
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker stop flask1 flask2 flask3
flask1
flask2
flask3
sheoraninfosec@BlackHat01:~/docker-app-demo$ docker rm flask1 flask2 flask3
flask1
flask2
flask3
sheoraninfosec@BlackHat01:~/docker-app-demo$
```

Step 14: come back to home directory.



```
sheoraninfosec@BlackHat01:~/docker-app-demo$ cd ~
sheoraninfosec@BlackHat01:~$ pwd
/home/sheoraninfosec
sheoraninfosec@BlackHat01:~$
```

----- END OF DOCUMENT -----