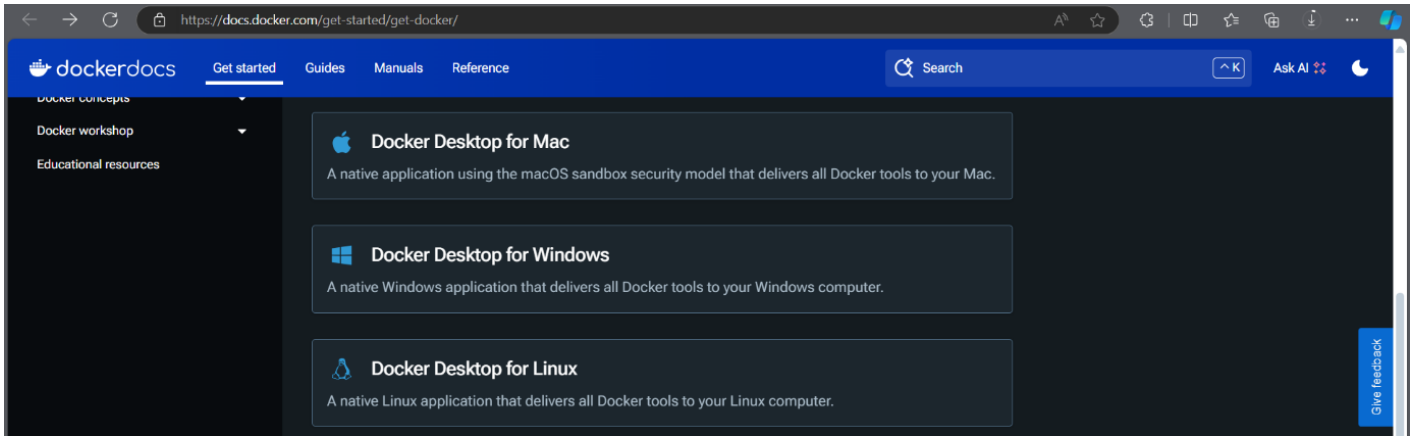
Exp 12 **Containerize and Run a sample Application**
07/10/24

Aim:

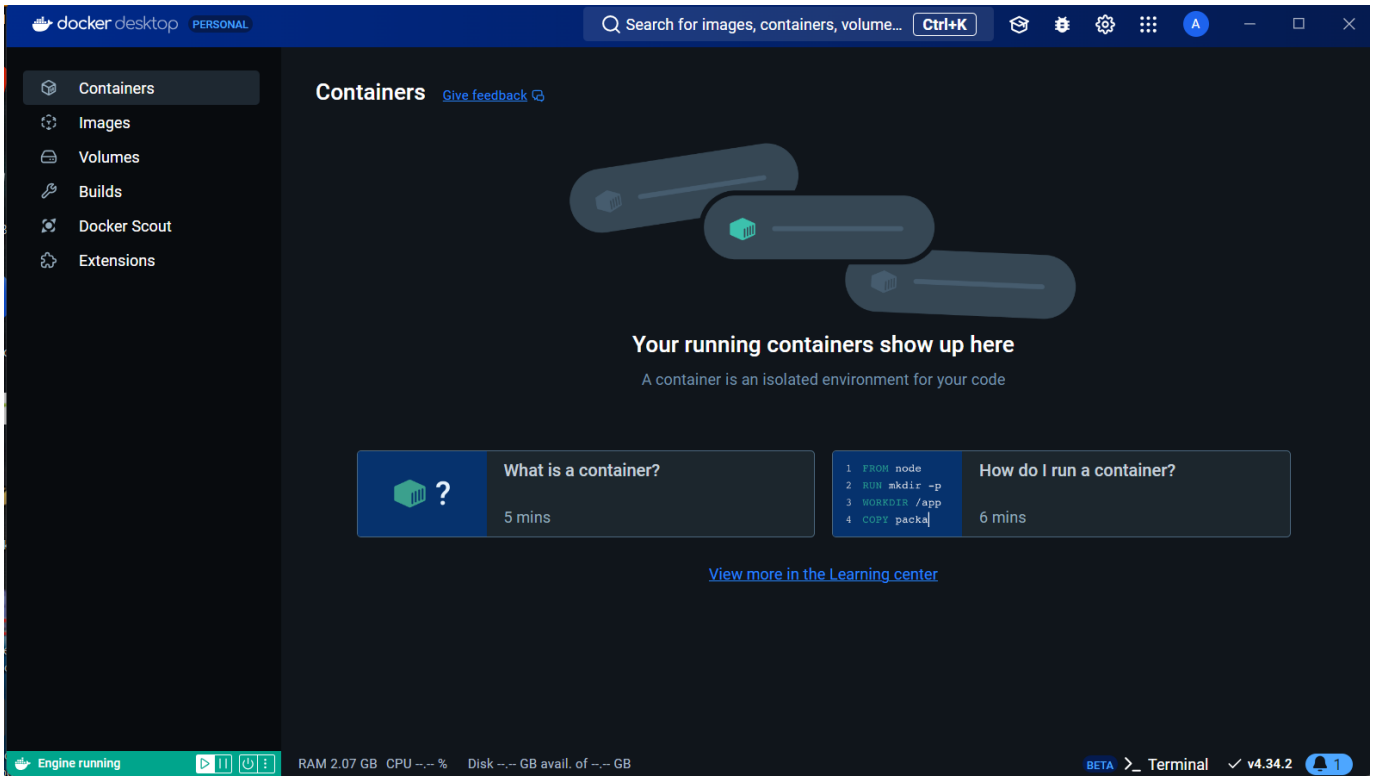
To containerize and run a Python application using Docker.

Procedure:

1. Install the Docker in desktop :
   https://docs.docker.com/get-started/get-docker/



Complete the setup and run the application.



Check the version

```
C:\Users\anbar>docker --version
Docker version 27.2.0, build 3ab4256
```

2. Save the file *app.py*

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
   return "Welcome to Cloud Lab!"

if __name__ == "__main__":
   app.run(host="0.0.0.0")
```

3. Save the Dockerfile .

```
C:\Users\anbar>type nul > Dockerfile
```

```
C:\Users\anbar>type nul > Dockerfile

C:\Users\anbar>notepad Dockerfile

C:\Users\anbar>
```

```
FROM python:3.11.9-alpine
RUN pip install --upgrade pip
RUN pip install flask
CMD ["python","app.py"]
COPY app.py /app.py
```

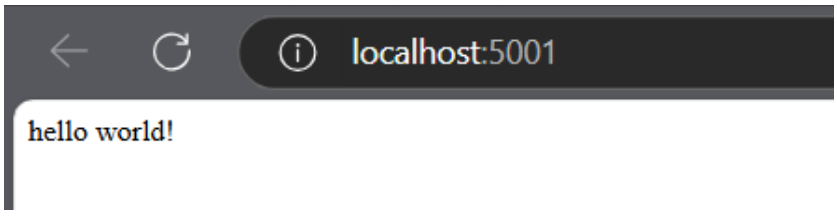
4. Build docker using the command: *docker image build -t python-hello-world .*

```
C:\Users\anbar>docker image build -t python-hello-world .
[+] Building 45.7s (9/9) FINISHED                                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                              0.1s
 => => transferring dockerfile: 164B                                                                                              0.0s
 => [internal] load metadata for docker.io/library/python:3.11.9-alpine                                                           3.9s
 => [internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 2B                                                                                                   0.0s
 => [1/4] FROM docker.io/library/python:3.11.9-alpine@sha256:f9ce6fe33d9a5499e35c976df16d24ae80f6ef0a28be5433140236c2ca482686     21.8s
 => => resolve docker.io/library/python:3.11.9-alpine@sha256:f9ce6fe33d9a5499e35c976df16d24ae80f6ef0a28be5433140236c2ca482686      0.1s
 => => sha256:420e49ae696c51bf9aa15e83080c0642570ec1b0bb3856c2b8a870b5bac450e5 3.13MB / 3.13MB                                   14.0s
 => => sha256:aa7d806471fdf84576e269237ffa832b9840d1a8c9e8adf5de53ca7aa92e54b9 229B / 229B                                        2.5s
 => => sha256:8068c973c3ecc523cec597574d23f6fe384aee2c1b7c6f0cd54b2d84ba857e70 12.67MB / 12.67MB                                  20.2s
 => => sha256:b9afd1f50702b9ff2b187430471498298c96d4919de0ed435aa4ecaf72ba6534 455.09kB / 455.09kB                                 4.8s
 => => sha256:43c4264eed91be63b206e17d93e75256a6097070ce643c5e8f0379998b44f170 3.62MB / 3.62MB                                   15.2s
 => => extracting sha256:43c4264eed91be63b206e17d93e75256a6097070ce643c5e8f0379998b44f170                                         0.3s
 => => extracting sha256:b9afd1f50702b9ff2b187430471498298c96d4919de0ed435aa4ecaf72ba6534                                         0.4s
 => => extracting sha256:8068c973c3ecc523cec597574d23f6fe384aee2c1b7c6f0cd54b2d84ba857e70                                         0.7s
 => => extracting sha256:aa7d806471fdf84576e269237ffa832b9840d1a8c9e8adf5de53ca7aa92e54b9                                         0.1s
 => => extracting sha256:420e49ae696c51bf9aa15e83080c0642570ec1b0bb3856c2b8a870b5bac450e5                                         0.4s
 => [internal] load build context                                                                                                 0.0s
 => => transferring context: 28B                                                                                                  0.0s
 => [2/4] RUN pip install --upgrade pip                                                                                           8.2s
 => [3/4] RUN pip install flask                                                                                                   8.7s
 => [4/4] COPY app.py /app.py                                                                                                     0.2s
 => exporting to image                                                                                                            2.5s
 => => exporting layers                                                                                                           1.6s
 => => exporting manifest sha256:a3fedcf4996825ae83c6e31845f5f9b5121a26b3fbc22bc98d4c7b641e93a3aa                                 0.0s
 => => exporting config sha256:be645705643d1e3927ccfd3022ae96f4d17854d247f60a627c48e6aef2e4a696                                   0.1s
 => => exporting attestation manifest sha256:bcb0af6038845cf0433501d05e0b96c8715e771005bb0b7d66d9aca54ebf5d53                     0.1s
 => => exporting manifest list sha256:0ded6d0b37734f5d15905ac1c9e0708389f08f0d0095d419b75fafe3cbb4dbdf                             0.1s
 => => naming to docker.io/library/python-hello-world:latest                                                                      0.0s
 => => unpacking to docker.io/library/python-hello-world:latest                                                                   0.5s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Users\anbar>
```

5. Run the docker using the command : *docker run -p 5001:5000 -d python-hello-world*

```
C:\Users\anbar>docker run -p 5001:5000 -d python-hello-world
3fce99f6d72bcad92264f555b033f109724dde6ce403cc3b56d7b71d2bed6325
```

Visit : http://localhost:5001/



APPLICATION:

1. Save your code *factorial.py*

```
C:\Cloud\Docker>notepad factorial.py
```

```
from flask import Flask, request, jsonify

app = Flask(__name__)

def factorial(n):

    if n == 0:

        return 1

    return n * factorial(n - 1)
```

```
@app.route('/factorial', methods=['GET'])
def get_factorial():
    num = request.args.get('num', default=10, type=int)  # Default to 10 if no query parameter
    fact_value = factorial(num)
    return jsonify(factorial=fact_value)
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

2. Save the requirements and Dockerfile.

```
C:\Cloud\Docker_app>type nul > Dockerfile

C:\Cloud\Docker_app>notepad Dockerfile
```

Dockerfile

File Edit View

```
# Use the official Python image from the Docker Hub
FROM python:3.11.9-alpine

# Upgrade pip
RUN pip install --upgrade pip

# Install Flask
RUN pip install flask

# Copy the factorial.py into the container
COPY factorial.py /factorial.py

# Command to run the Python app
CMD ["python", "factorial.py"]
```

3. Build your docker application.

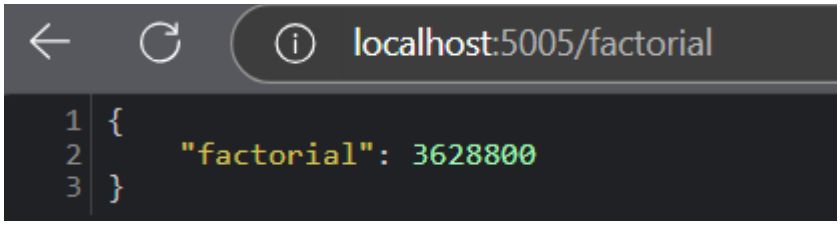
```
C:\Cloud\Docker_app>docker image build -t python-factorial .
[+] Building 32.1s (10/10) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                    0.2s
 => => transferring dockerfile: 355B                                                                                    0.1s
 => [internal] load metadata for docker.io/library/python:3.11.9-alpine                                                 6.1s
 => [auth] library/python:pull token for registry-1.docker.io                                                           0.0s
 => [internal] load .dockerignore                                                                                       0.1s
 => => transferring context: 2B                                                                                         0.0s
 => [1/4] FROM docker.io/library/python:3.11.9-alpine@sha256:f9ce6fe33d9a5499e35c976df16d24ae80f6ef0a28be54331402  0.4s
 => => resolve docker.io/library/python:3.11.9-alpine@sha256:f9ce6fe33d9a5499e35c976df16d24ae80f6ef0a28be54331402  0.1s
 => [internal] load build context                                                                                       0.3s
 => => transferring context: 505B                                                                                       0.1s
 => [2/4] RUN pip install --upgrade pip                                                                                15.4s
 => [3/4] RUN pip install flask                                                                                         5.9s
 => [4/4] COPY factorial.py /factorial.py                                                                               0.1s
 => exporting to image                                                                                                  3.5s
 => => exporting layers                                                                                                 1.8s
 => => exporting manifest sha256:96cd9da13cd831da820d8c405337ed1e5c37220971ce10aad507a59f0412f110                       0.0s
 => => exporting config sha256:f2a7136a3dba3723ebcc2a225b0c1322241e611768c301dec3e8c2c5e56b82bd                         0.0s
 => => exporting attestation manifest sha256:59c44d9b1e6088c31e304df1996137e5d14fa0d01dae392dd4844798c4888d32           0.1s
 => => exporting manifest list sha256:3dc191bed83ad3ca761e8469189af47970af958aecc7404c29f236e9f106baae                  0.0s
 => => naming to docker.io/library/python-factorial:latest                                                              0.0s
 => => unpacking to docker.io/library/python-factorial:latest                                                           1.4s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

4. Run your docker application.

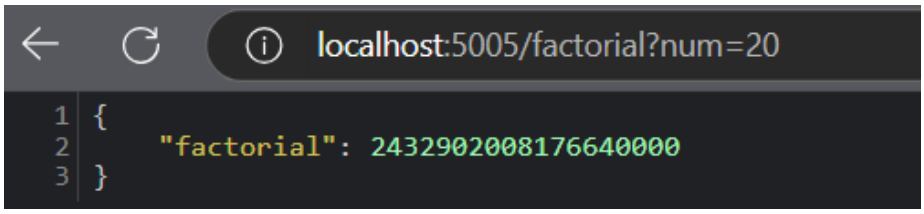
```
C:\Cloud\Docker_app>docker run -p 5005:5000 -d python-factorial
68ff9c0e02c7a9fe5efb47de8d581c809db590ff514040cee90bd3c39e1993e7
```

docker run -p 5005:5000 -d python-factorial

Default : http://localhost:5005/factorial



Passing arguments : http://localhost:5005/factorial?num=20



RESULT :

Thus , containerizing a Python application using Docker has been executed successfully and output has been verified.