

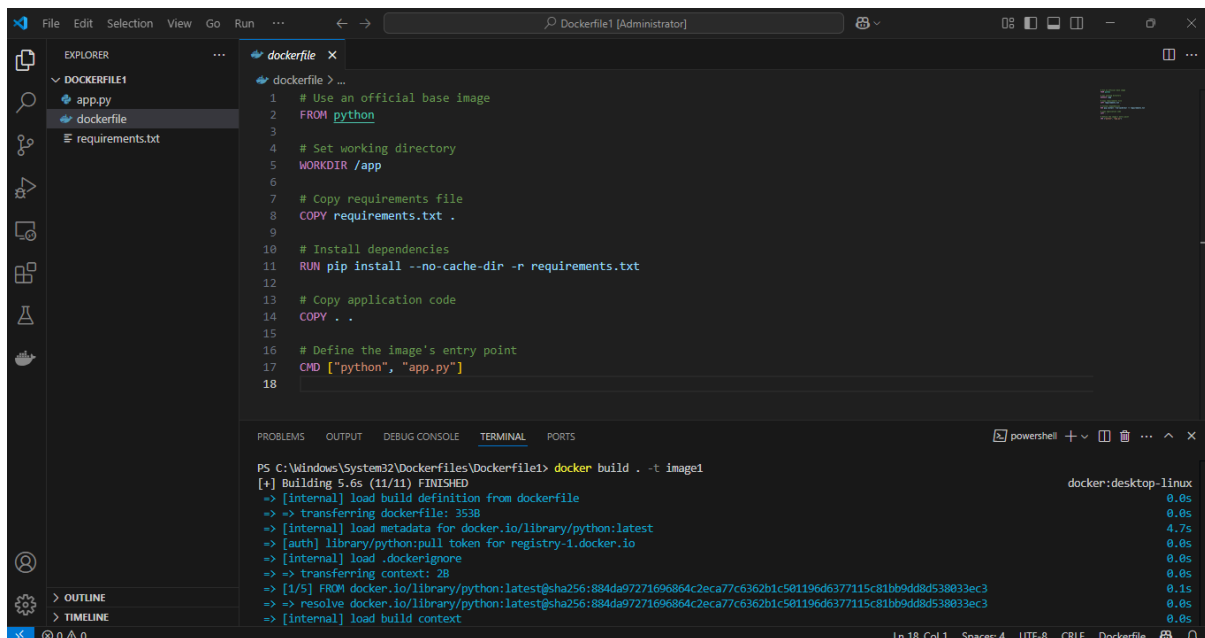
Containerization with Docker-Lab1

In this lab, I created an image of a dockerfile and pushed it to my repository in dockerhub.

A python image was pulled from the docker hub. The dockerfile was given several instructions. For instance, specifying the directory where the application will reside when the container is created from the image and also where all commands indicated on the dockerfile will be executed was specified using the WORKDIR command. Commands such as COPY, RUN and CMD was executed in this directory. All the files in the current directory of the local desktop were copied into the directory specified in the container as the work directory.

The requirement.txt file which was a dependency of the application was copied first to provide the specifications needed for the installation of the python hence reducing slow build times which may be as a result of unnecessary reinstallations of dependencies and reduction of potential errors. These dependencies were used by the application when it ran. The CMD command was final step in the container's startup process. This enables the python script to be executed by the python installed.

The Docker build command was used to create an image from the dockerfile and the docker run command created a container based on the image.

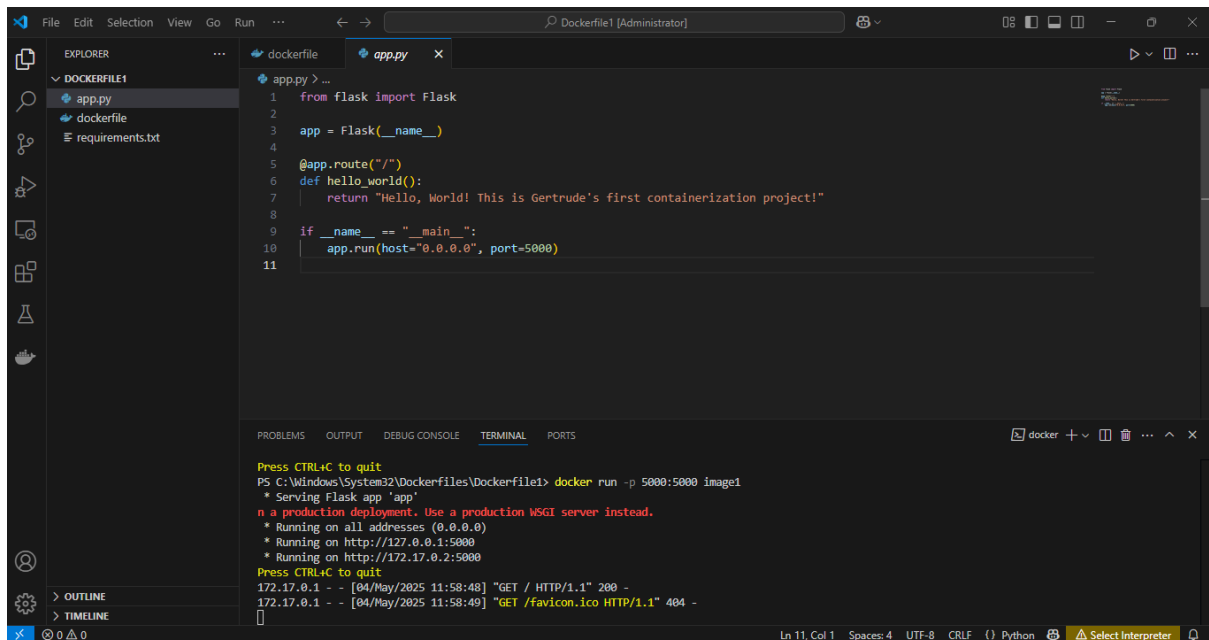


The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains the following instructions:

```
1 # Use an official base image
2 FROM python
3
4 # Set working directory
5 WORKDIR /app
6
7 # Copy requirements file
8 COPY requirements.txt .
9
10 # Install dependencies
11 RUN pip install --no-cache-dir -r requirements.txt
12
13 # Copy application code
14 COPY . .
15
16 # Define the image's entry point
17 CMD ["python", "app.py"]
18
```

The terminal at the bottom shows the output of the `docker build` command, indicating that the image was successfully built and pushed to the registry.

```
PS C:\Windows\System32\Dockerfiles\Dockerfile1> docker build . -t image1
[*] Building 5.6s (11/11) FINISHED
-> [internal] load build definition from dockerfile
-> transferring dockerfile: 353B
-> [internal] load metadata for docker.io/library/python:latest
-> [auth] library/python:pull token for registry-1.docker.io
-> [internal] load .dockerignore
-> transferring context: 2B
-> [1/5] FROM docker.io/library/python:latest@sha256:884da97271696864c2eca77c6362b1c501196d6377115c81bb9dd8d538033ec3
-> resolve docker.io/library/python:latest@sha256:884da97271696864c2eca77c6362b1c501196d6377115c81bb9dd8d538033ec3
-> [internal] load build context
```

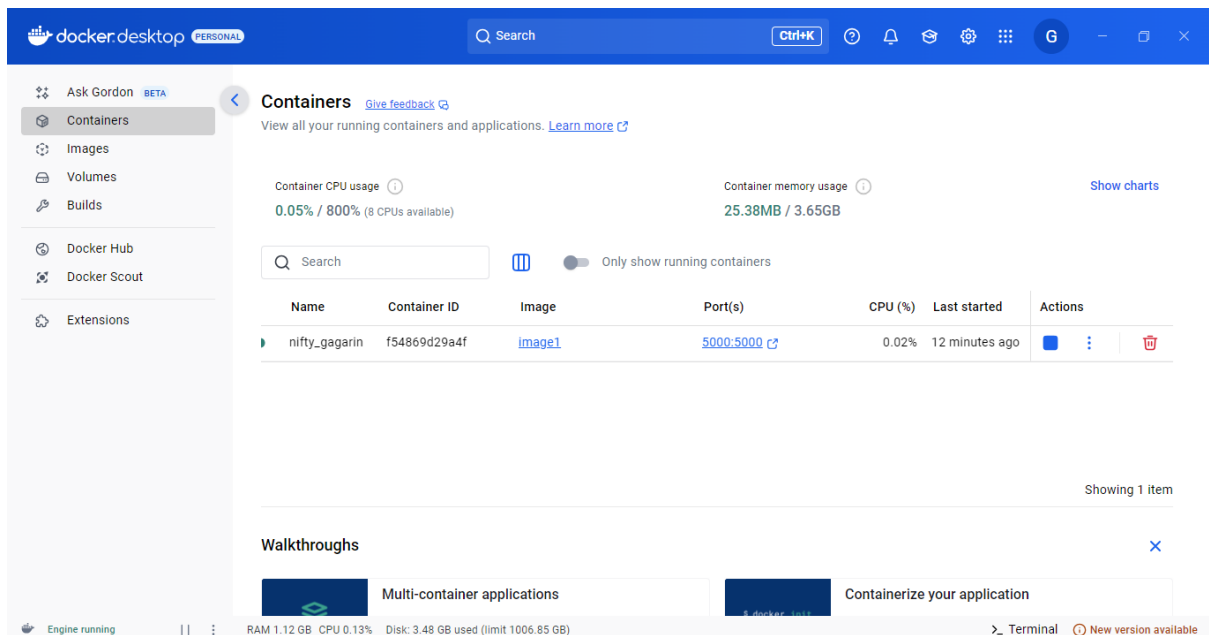


The screenshot shows the Visual Studio Code editor with a Dockerfile and a Flask application (app.py) open. The Dockerfile is in the left pane, and the app.py file is in the right pane. The terminal at the bottom shows the command `docker run -p 5000:5000 image1` and the output of the Flask application, which is a simple "Hello, World!" message. The terminal also shows the command `docker run -p 5000:5000 image1` and the output of the Flask application, which is a simple "Hello, World!" message.

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def hello_world():
7     return "Hello, World! This is Gertrude's first containerization project!"
8
9 if __name__ == "__main__":
10     app.run(host="0.0.0.0", port=5000)
11
```

Press CTRL+C to quit
PS C:\Windows\System32\Dockerfiles\Dockerfile1> docker run -p 5000:5000 image1
* Serving Flask app 'app'
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [04/May/2025 11:58:48] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [04/May/2025 11:58:49] "GET /favicon.ico HTTP/1.1" 404 -

To allow inbound traffic into the container, port mapping was applied to the container during its creation. While the dockerfile provided a template for creating an image from which the container was built, the port mapping determined the traffic flow into the container for testing and development of the application in the container environment. The host machine acted as a proxy which forwarded traffic to the container allowing the testing and validation of the application inside a container.



Because the dockerfile was configured to copy app.py file from the local machine into a directory in the container, after the container was built, the application was tested and validated in the container.



The docker image was pushed to my docker hub as shown below.

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> cd dockerfiles
PS C:\WINDOWS\system32\dockerfiles> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
image1 latest 9ba651358262 8 hours ago 1.49GB
PS C:\WINDOWS\system32\dockerfiles> docker tag image1:latest gertrudechichi/dockerlabs:latest
PS C:\WINDOWS\system32\dockerfiles> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
gertrudechichi/dockerlabs latest 9ba651358262 8 hours ago 1.49GB
image1 latest 9ba651358262 8 hours ago 1.49GB
PS C:\WINDOWS\system32\dockerfiles> docker push gertrudechichi/dockerlabs:latest
The push refers to repository [docker.io/gertrudechichi/dockerlabs]
aid29a344585: Pushed
e1307aae4433: Pushed
5680a53553bf: Pushed
776493ee5e4c: Pushed
cf05a52c0235: Pushed
63964a8510f5: Pushed
39ca2d92a129: Pushed
c187b51b626e: Pushed
ca513cad200b: Pushed
fbb336c69786: Pushed
ab89b3116421: Pushed
61d6683bb1b4: Pushed
latest: digest: sha256:9ba651358262847e099a309b551f0f552ddb689c8a1ec5c1620e2d778eb54fc9 size: 856
PS C:\WINDOWS\system32\dockerfiles>
```

Explore

My Hub

Search Docker Hub

Cr1k

G

gertrudechichi

Docker Personal

Repositories

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories / dockerlabs / General

Using 0 of 1 private repositories. [Get more](#)

gertrudechichi/dockerlabs

Last pushed 1 minute ago • Repository size: 368.9 MB

Add a description

Add a category

Docker commands

To push a new tag to this repository:

```
docker push gertrudechichi/dockerlabs:tagname
```

Public view

General

Tags

Image Management

Collaborators

Webhooks

Settings

Tags

DOCKER SCOUT INACTIVE

Activate

This repository contains 0 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	1 minute

See all

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your