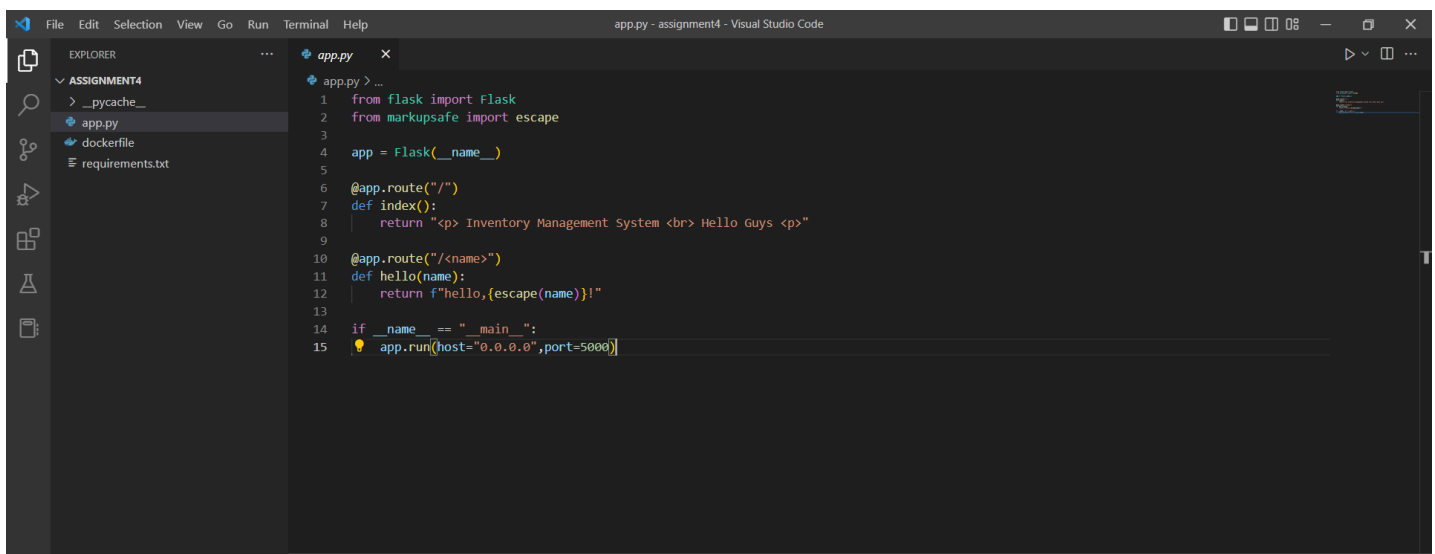


# Containerize Your Flask Application

Student Name	Rishi Ragul.R
Student Roll Number	610519104082

## Question:

### 1. Add docker file in flask app

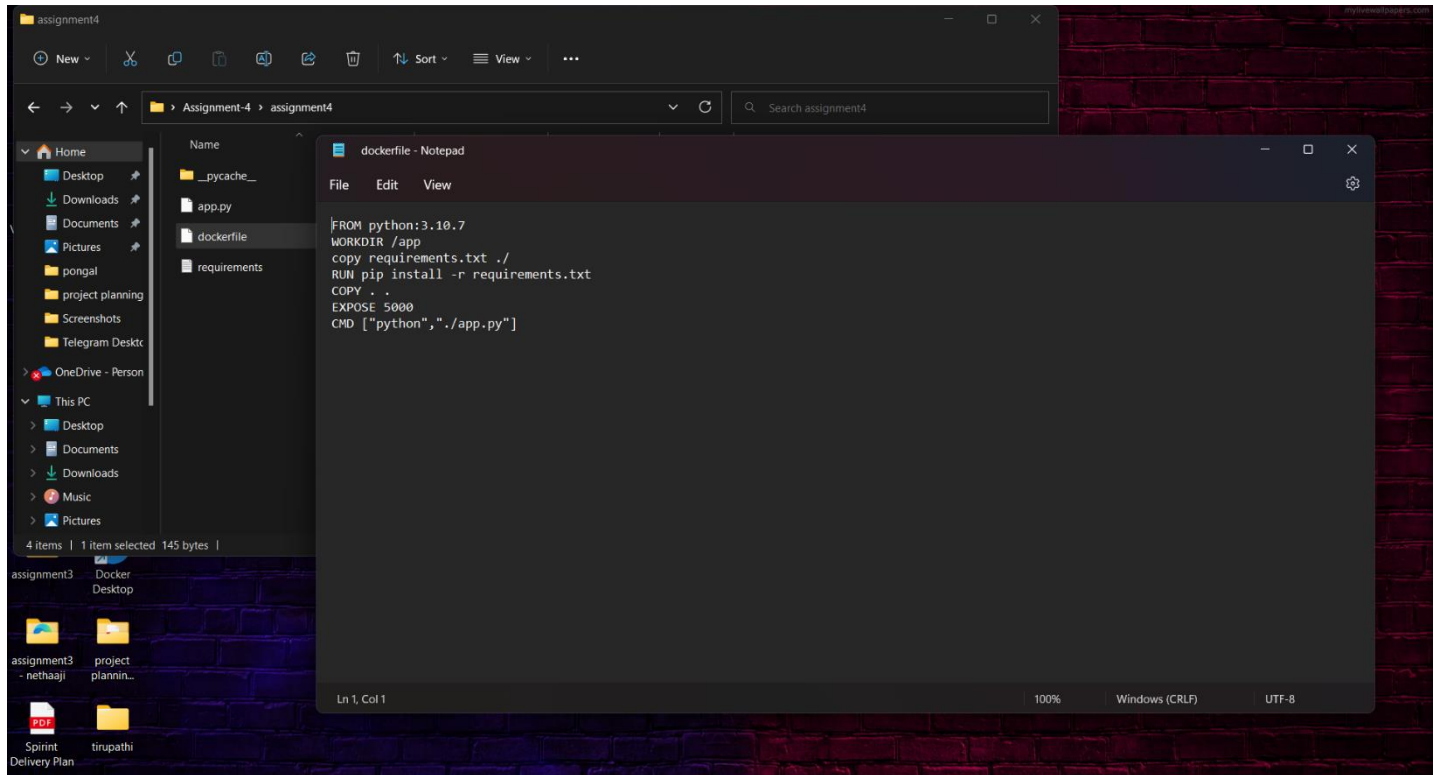


The screenshot shows the Visual Studio Code interface with a Flask application in `app.py` and a `Dockerfile` in the Explorer. The `app.py` file contains the following code:

```
1 from flask import Flask
2 from markupsafe import escape
3
4 app = Flask(__name__)
5
6 @app.route("/")
7 def index():
8     return "<p> Inventory Management System <br> Hello Guys <p>"
9
10 @app.route("/<name>")
11 def hello(name):
12     return f"hello,{escape(name)}!"
13
14 if __name__ == "__main__":
15     app.run(host="0.0.0.0",port=5000)
```

The Explorer on the left shows the project structure for `ASSIGNMENT4`, including `__pycache__`, `app.py`, `Dockerfile`, and `requirements.txt`.

## 2.Docker File.



## 3. Build docker image using docker build command.

```
Command Prompt

C:\Users\RISHI\Desktop\Assignment-4\assignment4>docker build -t docker_with_flask_inventory .
[+] Building 115.0s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/python:3.10.7 1.2s
=> [1/5] FROM docker.io/library/python:3.10.7@sha256:53e577204d362233ee92aeb5119449271f5eb24f99c61464efe9167ddbc8640f 109.0s
=> => resolve docker.io/library/python:3.10.7@sha256:53e577204d362233ee92aeb5119449271f5eb24f99c61464efe9167ddbc8640f 0.0s
=> => sha256:d884e56c3a7ca695cd8a4c439270ff19b01b39e22b47cae5ba410a858a13b107 8.53kB / 8.53kB 0.0s
=> => sha256:f60cd8928ed378229f2460b94b504cca239f9906efc57aebdf9340bd29bd5ddf 55.05MB / 55.05MB 73.0s
=> => sha256:47db815ca4547dc224b7522193cb1851cf529d2cbd726f954b9bbf97099b98 5.16MB / 5.16MB 3.0s
=> => sha256:53e577204d362233ee92aeb5119449271f5eb24f99c61464efe9167ddbc8640f 2.36kB / 2.36kB 0.0s
=> => sha256:d400cf9859b6814b180d9c8b522917b13a1ced9fc198f2a047220307fede1a5c 2.22kB / 2.22kB 0.0s
=> => sha256:bf48494000001a037b72870d2a6a2536f9da8bc5d1ceddd72d79f4a51fe7a69e 10.88MB / 10.88MB 10.3s
=> => sha256:a572f27a256d36a93ab0777949771b120c5d7dce75ea2a2d3d9444793b26b2ef1 54.58MB / 54.58MB 34.3s
=> => sha256:8f7d0525895528fdb73153451e112bbd8e1854549bd1e0e6f4ac0b4a2ee98172 196.85MB / 196.85MB 100.3s
=> => sha256:7110f04115ae2d7232ed9e59b97db7cf7337c91f95edc25428baa3e522064187 6.29MB / 6.29MB 41.3s
=> => sha256:c4b413ca4894499e2ee7df958d411ccfcc899dcccffe61de71cbb433e4e76143 20.05MB / 20.05MB 59.3s
=> => sha256:22311b72a3cb993d70dc2f9440feb89ca571ae931b080d975c81f4270ca909b8 231B / 231B 60.6s
=> => sha256:8dcfbe38b6fa1182030e006f135471f3726cef064f6fc90edbf80f450efad79 3.04MB / 3.04MB 64.3s
=> => extracting sha256:f60cd8928ed378229f2460b94b504cca239f9906efc57aebdf9340bd29bd5ddf 2.3s
=> => extracting sha256:47db815ca4547dc224b7522193cb1851cf529d2cbd726f954b9bbf97099b98 0.3s
=> => extracting sha256:bf48494000001a037b72870d2a6a2536f9da8bc5d1ceddd72d79f4a51fe7a69e 0.4s
=> => extracting sha256:a572f27a256d36a93ab0777949771b120c5d7dce75ea2a2d3d9444793b26b2ef1 3.0s
=> => extracting sha256:8f7d0525895528fdb73153451e112bbd8e1854549bd1e0e6f4ac0b4a2ee98172 6.0s
=> => extracting sha256:7110f04115ae2d7232ed9e59b97db7cf7337c91f95edc25428baa3e522064187 0.4s
=> => extracting sha256:c4b413ca4894499e2ee7df958d411ccfcc899dcccffe61de71cbb433e4e76143 0.8s
=> => extracting sha256:22311b72a3cb993d70dc2f9440feb89ca571ae931b080d975c81f4270ca909b8 0.8s
=> => extracting sha256:8dcfbe38b6fa1182030e006f135471f3726cef064f6fc90edbf80f450efad79 0.2s
=> [internal] load build context
=> => transferring context: 190B 0.0s
=> [2/5] WORKDIR /app 0.3s
=> [3/5] COPY requirements.txt ./ 0.0s
=> [4/5] RUN pip install -r requirements.txt 4.1s
=> [5/5] COPY . . 0.0s
=> exporting to image 0.2s
=> => exporting layers 0.1s
=> => writing image sha256:5d844c12d2eff9ba170fc15034f0c9c00efbed528ea42cb6a351f21377d4c6bd 0.0s
=> => naming to docker.io/library/docker_with_flask_inventory 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\RISHI\Desktop\Assignment-4\assignment4>kubectl get nodes
NAME STATUS ROLES AGE VERSION
docker-desktop Ready control-plane 6m19s v1.25.2

C:\Users\RISHI\Desktop\Assignment-4\assignment4>
```

## Running in docker desktop

The screenshot shows the Visual Studio Code editor with a Python file named `app.py` open. The code is a simple Flask application that serves a static page with the text "Inventory Management System" and "Hello Guys". It also has a `hello` endpoint that returns a personalized message. The terminal at the bottom shows the command to run the application using `python.exe`, and the output indicates that the application is running successfully on `http://127.0.0.1:5000`.

```
app.py > ...
1 from flask import Flask
2 from markupsafe import escape
3
4 app = Flask(__name__)
5
6 @app.route("/")
7 def index():
8     return "<p> Inventory Management System <br> Hello Guys <p>"
9
10 @app.route("/<name>")
11 def hello(name):
12     return f"hello, {escape(name)}!"
13
14 if __name__ == "__main__":
15     app.run(host="0.0.0.0", port=5000)
```

Terminal Output:

```
PS C:\Users\RISHI\Desktop\Assignment-4\assignment4> & C:\Users\RISHI\AppData\Local\Programs\Python\Python310\python.exe c:\Users\RISHI\Desktop\Assignment-4\assignment4\app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.101.5:5000
Press CTRL+C to quit
```

## 4. Run and test your container.

The screenshot shows the Docker Desktop interface. The left sidebar contains navigation options: Containers, Images, Volumes, and Dev Environments. The main area displays a list of images on disk, including local images and remote repositories. The table lists various images, their tags, IDs, and sizes. The `hubproxy.docker.internal:500...` image is highlighted. The bottom status bar shows system information: RAM 3.53GB, CPU 3.60%, and Connected to Hub.

Image Name	Tag	ID	Updated	Size
hubproxy.docker.internal:500...	latest	09d7e1dbc2c4	about 14 hours ago	363.32 MB
jp.icr.io/do-fs/docker_with_flas...	latest	5d844c12d2ef	about 14 hours ago	932.79 MB
jp.icr.io/do-fs/docker_with_flas...	latest	5d844c12d2ef	about 14 hours ago	932.79 MB
k8s.gcr.io/coredns	v1.9.3	5185b96f0bec	6 months ago	48.8 MB
k8s.gcr.io/etcd	3.5.4-0	a8a176a5d5d6	5 months ago	299.52 MB
k8s.gcr.io/kube-apiserver	v1.25.2	97801f839490	about 2 months ago	127.73 MB
k8s.gcr.io/kube-controller-ma...	v1.25.2	dbfceb93c69b	about 2 months ago	117.1 MB
k8s.gcr.io/kube-proxy	v1.25.2	1c7d8c51823b	about 2 months ago	61.69 MB
k8s.gcr.io/kube-scheduler	v1.25.2	ca0ea1ee3cfd	about 2 months ago	50.58 MB
k8s.gcr.io/pause	3.8	4873874c08ef	5 months ago	711.18 KB
kubernetes/dashboard	v2.6.1	783e2b6d87ed	3 months ago	245.72 MB
kubernetes/metrics-scraper	v1.0.8	115053965e86	5 months ago	43.82 MB
rishiragur/flask-docker	latest	5d844c12d2ef	about 14 hours ago	932.79 MB