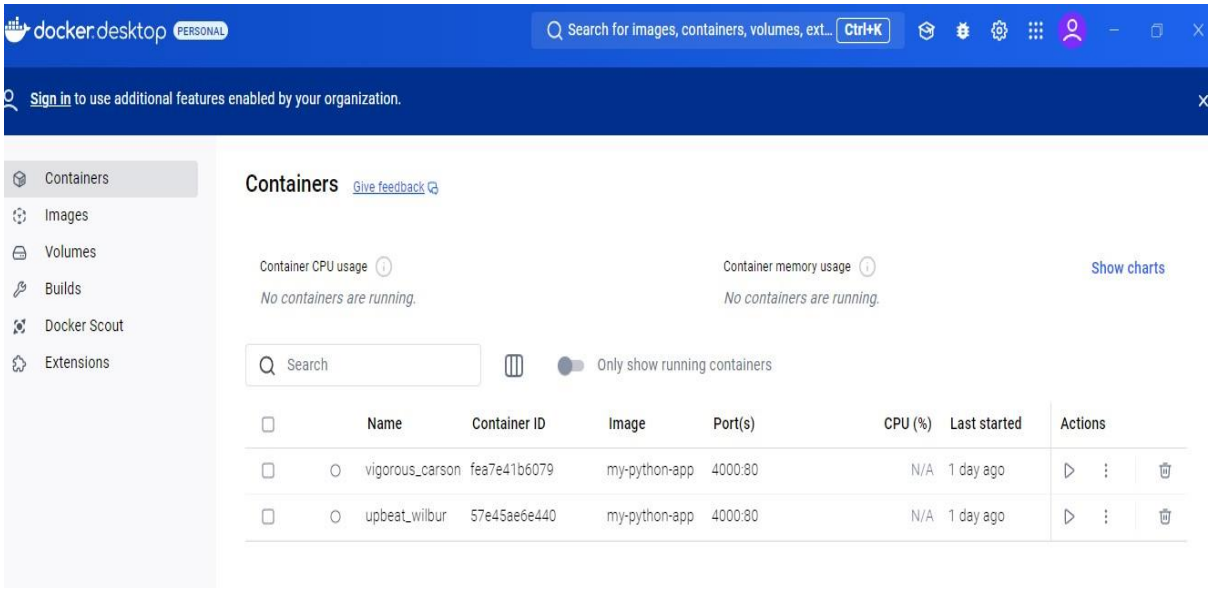
CONTAINERIZING A PYTHON APPLICATION WITH DOCKER: DEPLOYING PYTHON IN DOCKER WITH FLASK

Step 1: Write Your Python Application Code

1. **Create a Python script** called app.py with the following content:

```
# app.py  print("Python Application using
Docker!")
```



Step 2: Create a Dockerfile

1. Create a file named Dockerfile in the same directory as app.py. This file contains instructions for Docker to set up the environment for the Python application:

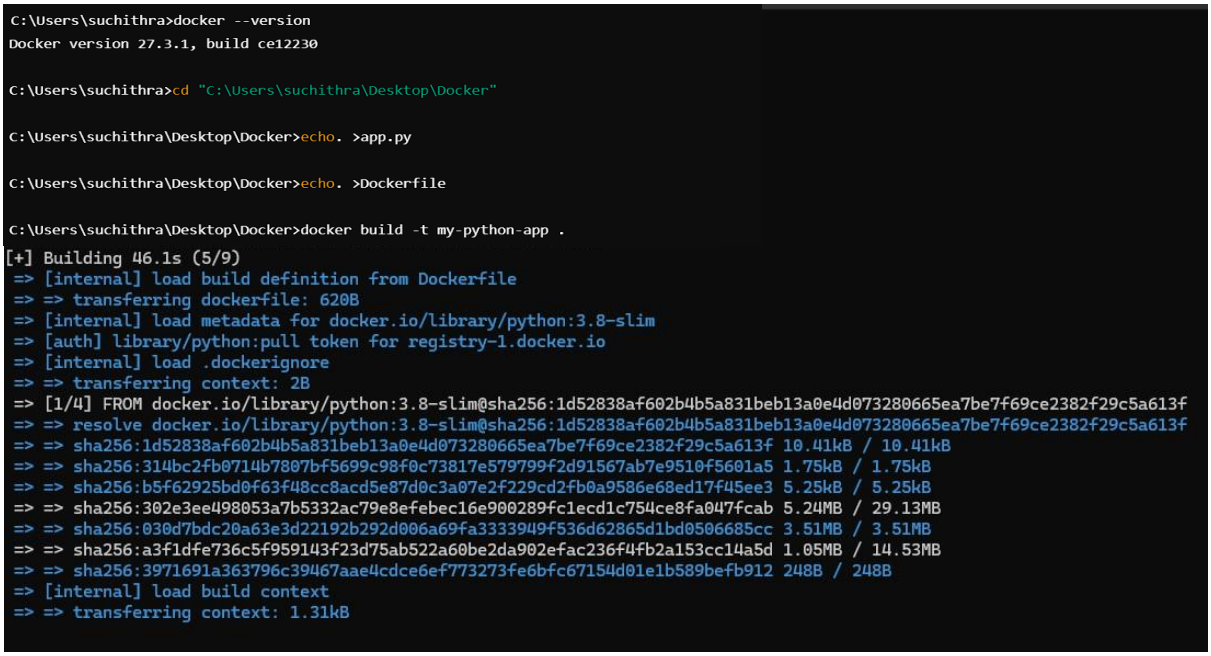
```
# Use an official Python runtime as a parent image
FROM python:3.8-slim
# Set the working directory in the container
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install any needed packages specified in requirements.txt
# (Skip this step if no external packages are needed)
RUN pip install --trusted-host pypi.python.org -r requirements.txt || true
# Make port 80 available to the outside world
EXPOSE 80
# Define environment variable
ENV NAME World
# Run app.py when the container launches
CMD ["python", "app.py"]
```

```
C:\Users\suchithra>cd "C:\Users\suchithra\Desktop\Docker"
C:\Users\suchithra\Desktop\Docker>echo. >app.py
C:\Users\suchithra\Desktop\Docker>echo. >Dockerfile
```

Step 3: Build the Docker Image

1. Build the image with the following command, replacing my-python-app with your preferred image name: docker build -t my-python-app .

   This command tells Docker to build an image using the Dockerfile in the current directory (.) and tag it as my-python-app.



ZZZ

Step 4: Run the Docker Container

1. Run the container with the following command:

   bash Copy code docker run -p
   4000:80 my-python-app

   This command:

   - Maps port 4000 on your machine to port 80 in the container.
   - Starts the container and runs app.py, which will print "Python Application using Docker!".

```
C:\Users\suchithra\Desktop\Docker>docker run -p 4000:80 my-python-app
hello Docker
```

Step 5: Access Your Python Application

Since app.py simply prints text to the console and doesn't start a web server, there won't be anything to interact with at http://localhost:4000. To create a web-based application, you'll need to use a Python web framework like Flask.

To display the message in a web browser, follow these steps:

1. **Install Flask** by creating a requirements.txt file with this line:

   Copy code Flask

2. **Update app.py** to create a simple web server:

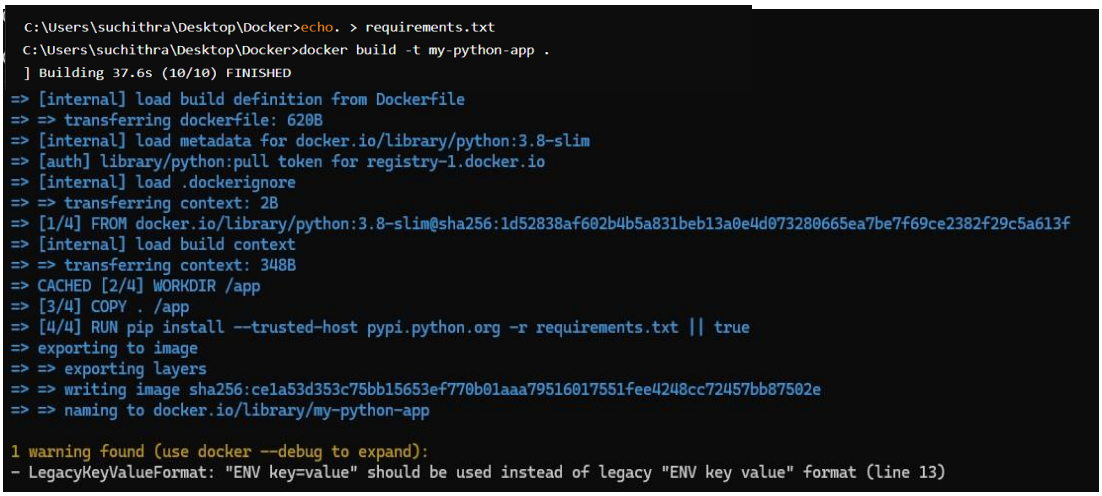
   python Copy code #
   app.py from flask
   import Flask
   app = Flask(__name__)

   @app.route("/") def
   hello():
   return "Python Application using Docker!"

   if __name__ == "__main__":
   app.run(host="0.0.0.0", port=80)

3. **Rebuild the Docker Image** (after modifying app.py and requirements.txt):

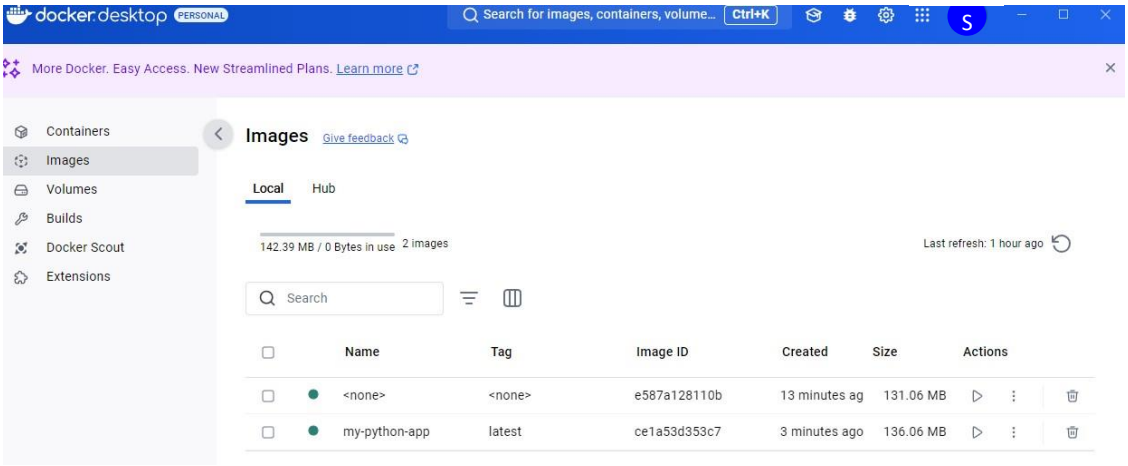
   docker build -t my-python-app .



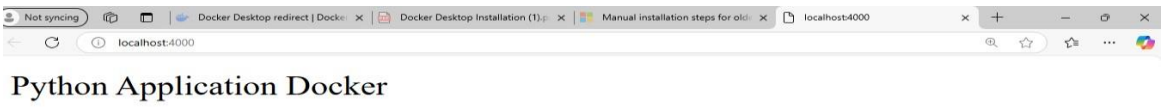
4 Run the Container Again: docker

run -p 4000:80 my-python-app

```
C:\Users\sriya\Desktop\Docker>docker run -p 4000:80 my-python-app
 * 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://127.0.0.1:80
 * Running on http://172.17.0.2:80
Press CTRL+C to quit
```



5.Access the Application by visiting http://localhost:4000 in a browser, where you should now see "Python Application using Docker!".



This setup will give you a simple Flask web application running in Docker, accessible via your browser.