24-2-2025

DOCKER:

DOCKER→

Docker is a platform for building shipping and running applications in containers.

Imagine you're baking a cake. You have a recipe (your app), ingredients (libraries and dependencies), and an oven (your computer).

1. The lunchbox → **Container** (holds everything the cake needs)
2. The recipe → **Docker image** (blueprint of the lunchbox)
3. The instructions → **Dockerfile** (step-by-step instructions for making the **lunchbox**)
4. The kitchen → **Docker Engine** (tool that runs the containers)
5. The grocery store → **Docker hub** (find pre-made images)

CONTAINER→

Containers are lightweight, portable packages that include everything the application needs to run (code, libs, sys tools and settings)

WHY DOCKER?

- **Consistency** → Ensures that your application runs the same way across different environments, from development to production.
- **Portability** → Docker containers can run on any machine that has Docker installed, regardless of the underlying operating system.
- **Efficiency** → Containers share the host operating system kernel, making them more lightweight and efficient than virtual machines.
- **Speed** → Docker containers start up quickly, making it faster to deploy and scale applications.

Docker CLI (command Line Interface) →

- It is a primary tool for interacting with Docker.
- Allows us to send instructions to the Docker Daemon.
- Docker run, docker build

Docker Daemon →

- It is a background process that manages Docker containers.
- It's the core of Docker, responsible for building, running, and managing your containers.
- The Daemon listens for requests from the Docker CLI. When it receives a request, it performs the action

Docker Registry →

- A Docker Registry is a storage service for Docker images. It's like a library where you can store and retrieve images.
- When you build a Docker image, you can push it to a registry. Others can then pull that image from the registry to run their own containers.
- Docker Hub

Basic Docker Commands:

1. **docker pull** → pull the image
2. **docker build -t my_application** → create a docker image
3. **docker push** → push code into the registry
4. **docker run my_application** → run the application
5. **docker ps** --> how many containers are running

DOWNLOAD DOCKER:

1. echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
2. sudo apt update
3. sudo apt install -y docker-ce docker-ce-cli containerd.io
4. docker --version
5. sudo usermod -aG docker $USER
6. docker –version

HOW DOCKER WORKS?

- Set up the basic project structure for a **Python** application that will be containerized with Docker.
- Create the necessary directories and files, including source code locations, dependency lists, Docker build instructions, and files to be excluded from the Docker image.

```
1825  mkdir python-docker-project
1826  cd python-docker-project
1827  mkdir src tests
1828  touch src/__init__.py
1829  touch src/main.py
1830  touch requirements.txt
1831  touch Dockerfile
1832  touch .dockerignore
1833  touch docker-compose.yml
1834  vi requirements.txt
1835  vi .dockerignore
1836  code .
```

- **requirements.txt** → has the required libs to download

requirements.txt

```
Flask>=2.2.0
Werkzeug>=2.2.0
gunicorn
```

- **Flask** is a web framework for building web applications and it provides only the essential tools for web development.
- **Werkzeug** is a library providing utilities for WSGI (Web Server Gateway Interface) applications which is often used by flask.
- **Gunicorn** is a production-ready WSGI HTTP server commonly used to serve Flask applications.

.docker.ignore → This file tells Docker which files and directories to *exclude* when building a Docker image.

.dockerignore

```
__pycache__
*.pyc
*.pyo
*.pyd
.Python
env/
venv/
.env
*.log
.git
.gitignore
Dockerfile
.dockerignore
tests/
README.md
```

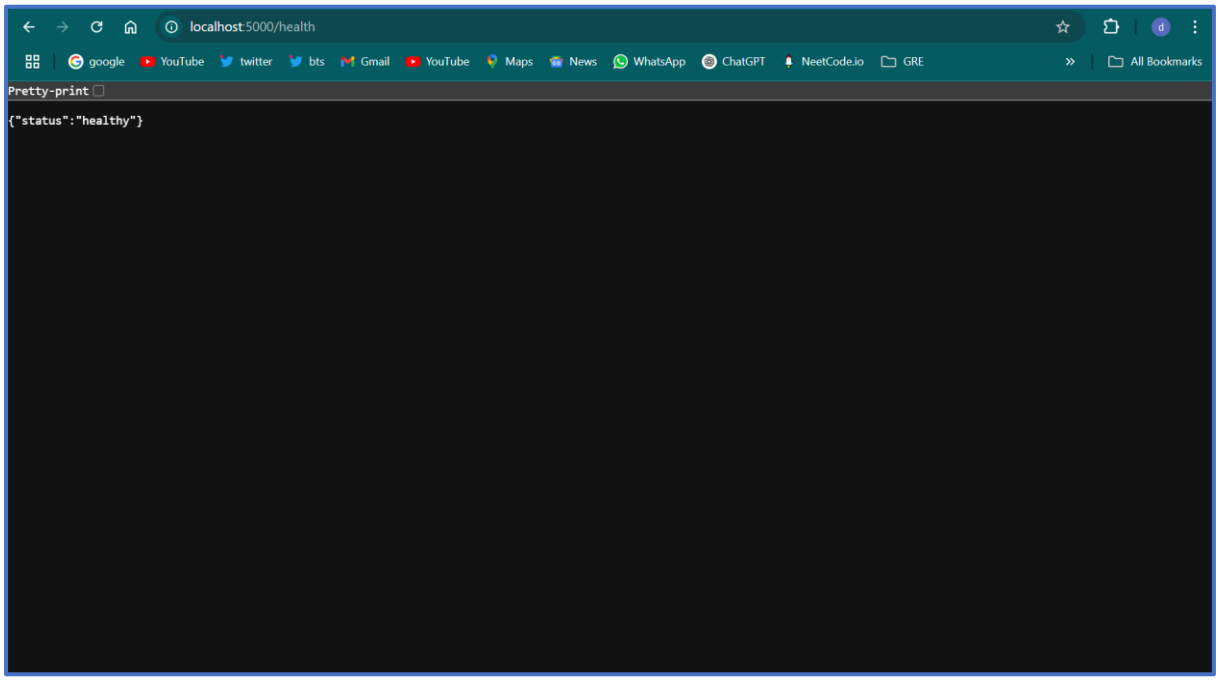
- Create a virtual environment → **python3 -m venv env**
- Activate the virtual environment → **source env/bin/activate**
- Install the python packages → **pip install -r requirements.txt**

Main.py → Defines a Flask web application.

- **/health** → Returns a JSON response indicating the health status of the application.
- **/** → Returns a JSON response with a greeting message.
- **/pyramid/<int:n>** → Takes an integer n as input and returns a text response containing a star pyramid pattern

```python
from flask import Flask,jsonify, Response
app = Flask(__name__)
Qodo Gen: Options | Test this function
@app.route("/health")
def health_check():
    return jsonify({"status": "healthy"})
Qodo Gen: Options | Test this function
@app.route("/")
def hello_world():
    return jsonify({"message":"Hello from Khushi docker"})

Qodo Gen: Options | Test this function
@app.route("/pyramid/<int:n>")
def star_pyramid(n):
    pattern = "\n".join([" " * (n - i - 1) + "*" * (2 * i + 1) for i in range(n)])
    return Response(pattern, mimetype="text/plain")

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=5000)
```

- **python3 src/main.py** → this will execute the main.py



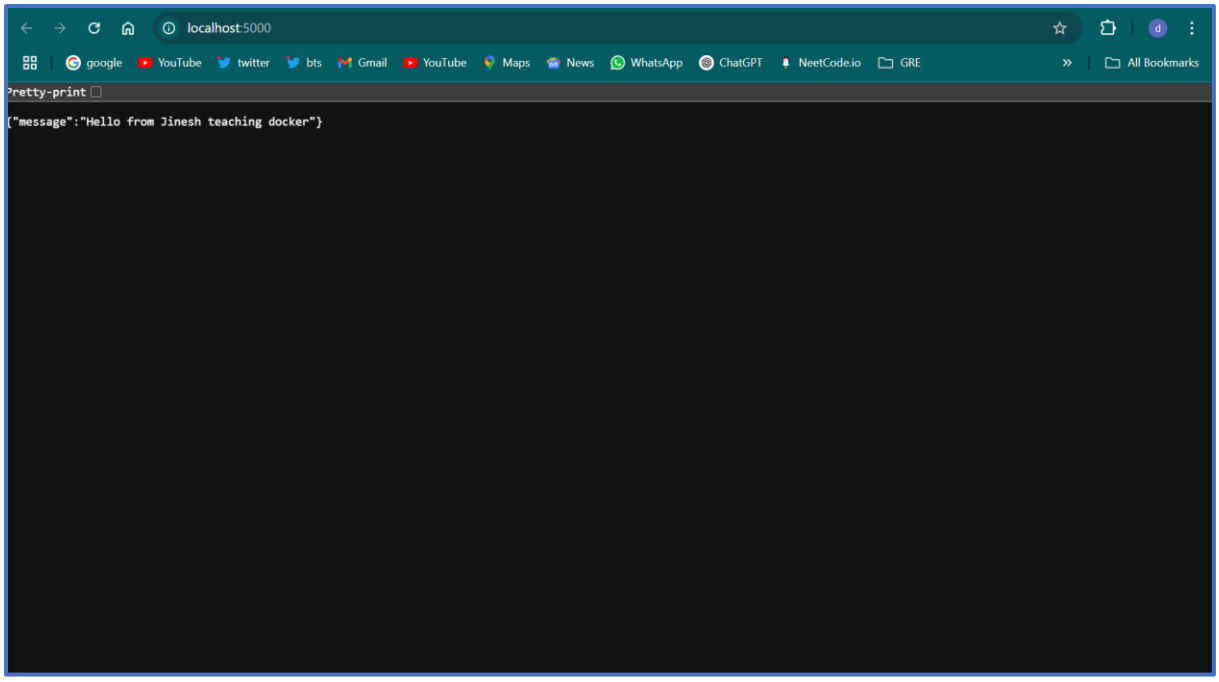
Dockerfile →

- **FROM** → Specifies the base image to start from (python)
- **RUN** → Executes commands inside the image
- **COPY** → Copies files from your local machine into the image
- **CMD** → Specifies the command to run when the container starts
- **WORKDIR** → Sets the working directory inside the image
- **EXPOSE** → Declares which ports the container will listen on
- **ENV** → Flask environment

Dockerfile > ...

```dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt && \
    pip install --no-cache-dir -r requirements.txt

COPY . .

ENV FLASK_APP=src/main.py
ENV FLASK_ENV=development
ENV PYTHONPATH=/app

EXPOSE 5000

CMD ["gunicorn","--bind","0.0.0.0:5000","src.main:app"]
```

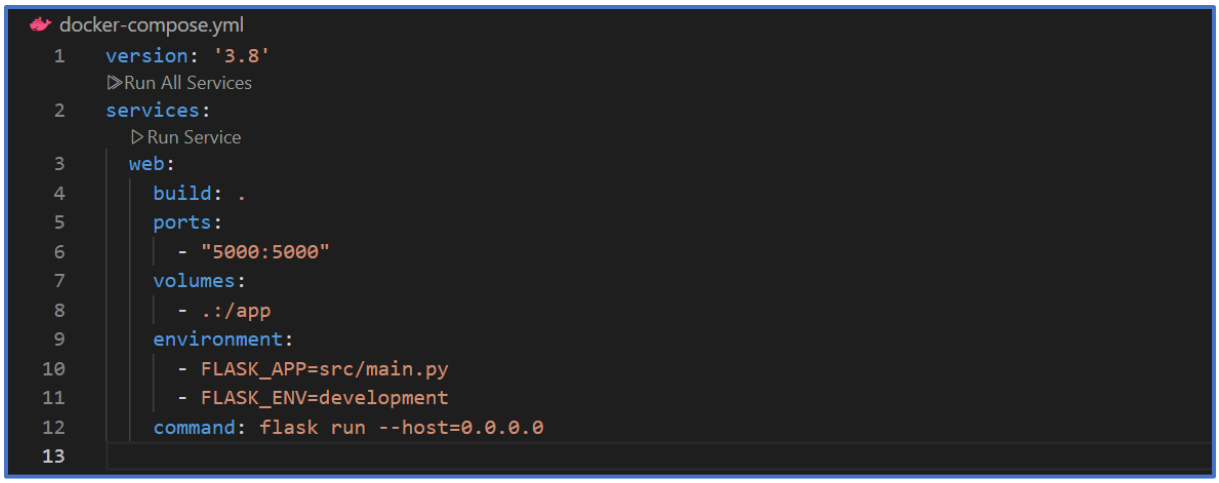
- This Dockerfile uses a Python base image, installs project dependencies, copies the application code, sets environment variables for Flask, exposes port 5000, and starts the application using Gunicorn.
- It containerizes the **Python Flask app** for easy deployment.
- **docker build -t python-docker-app .** → build the docker image and tags it with the name "python-docker-app"
- **docker run -p 5000:5000 python-docker-app** → runs the docker container



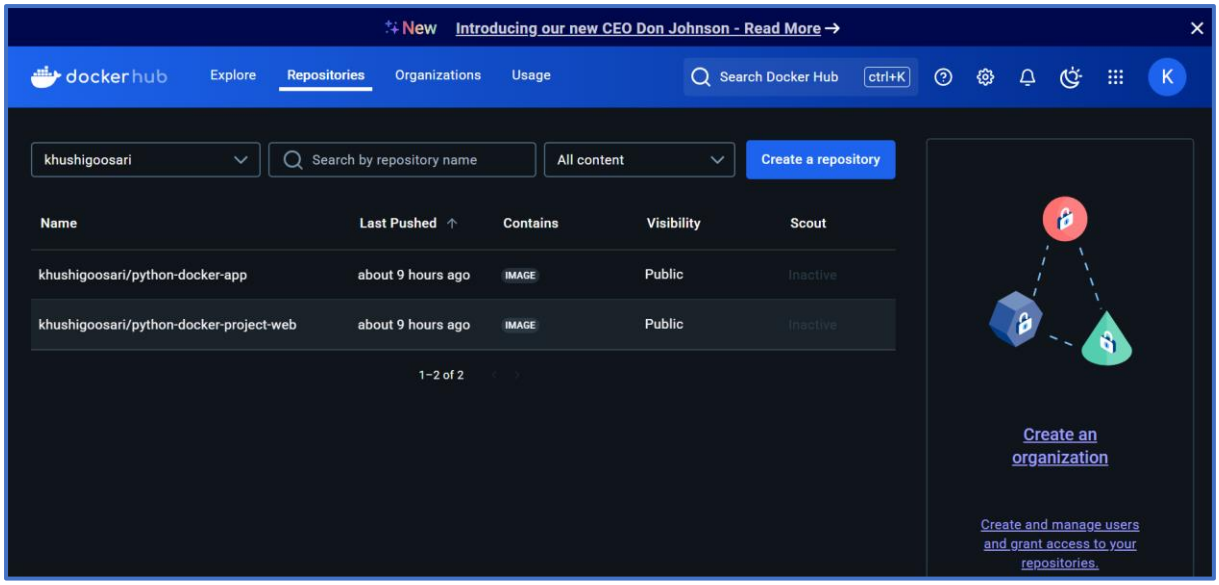
INSTALL DOCKER COMPOSE:

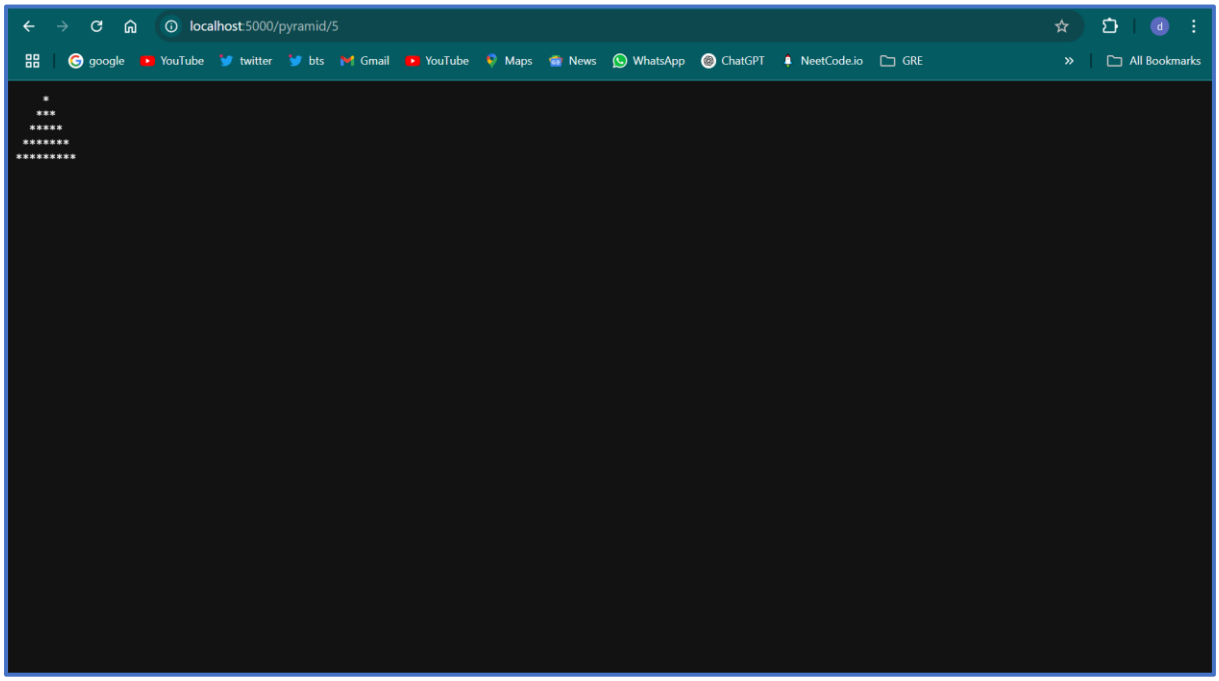
1. sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
2. sudo chmod 711 /usr/local/bin/docker-compose
3. docker-compose --version
4. docker-compose up –build

docker-compose.yml → Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file



- Defines a web service that builds a Docker image from the current directory, maps port 5000, mounts the current directory as a volume, sets Flask environment variables, and runs the Flask development server.
- It simplifies running the Flask app in a Docker container for development.
- Create account in **Docker Hub**
- **docker login**
- **docker images** --> images created
- **docker tag python-docker-app:latest khushigoosari/python-docker-app:v1**
- **docker push khushigoosari/python-docker-app:v1**
- **docker pull khushigoosari/python-docker-app:v1**
- **run the docker file--> docker run -p 5000:5000 khushigoosari/python-docker-app:v1**


New Introducing our new CEO Don Johnson - Read More →
docker hub
Explore
Repositories
Organizations
Usage
Search Docker Hub
ctrl+K
K
khushigoosari
Search by repository name
All content
Create a repository
Name
Last Pushed
Contains
Visibility
Scout
khushigoosari/python-docker-app
about 9 hours ago
IMAGE
Public
Inactive
khushigoosari/python-docker-project-web
about 9 hours ago
IMAGE
Public
Inactive
1–2 of 2
Create an organization
Create and manage users and grant access to your repositories.


localhost:5000/pyramid/5
google
YouTube
twitter
bts
Gmail
YouTube
Maps
News
WhatsApp
ChatGPT
NeetCode.io
GRE
All Bookmarks
*
***
*****
****
****