

Create and Manage Docker Containers on Linux

Objective

Learn to create, manage, and deploy Docker containers on a Linux system. This task demonstrates Docker's core concepts, container lifecycle, and basic commands.

1. Setup Environment

Install Docker on Ubuntu and ensure it runs properly.

Commands:

```
sudo apt update
```

```
sudo apt install -y docker-ce docker-ce-cli containerd.io or sudo apt install -y docker.io
```

```
sudo systemctl enable docker
```

```
sudo systemctl start docker
```

```
sudo systemctl status docker
```

```
docker --version.
```

```
See "man sudo_root" for details.

ubuntu@ip-172-31-12-81:~$ sudo apt update
Hit:1 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
]
Get:3 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Reading state information... Done
41 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-12-81:~$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx
  docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
```

```
No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-12-81:~$ sudo systemctl start docker
ubuntu@ip-172-31-12-81:~$ sudo systemctl enable docker
ubuntu@ip-172-31-12-81:~$
```

2. Basic Docker Commands

Test Docker installation:

```
sudo docker run hello-world or[ sudo usermod -aG docker $USER
newgrp docker ] this allows running docker without sudo.
```

```
Pan 'docker run --help' for more information.  
ubuntu@ip-172-31-12-81:~$ sudo usermod -aG docker $USER  
ubuntu@ip-172-31-12-81:~$ newgrp docker  
ubuntu@ip-172-31-12-81:~$ docker run hello-world  
Hello from Docker!
```

3. Explore Docker Images

Search for an image:

```
sudo docker search nginx
```

```
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
  
ubuntu@ip-172-31-12-81:~$ docker search nginx  
NAME          DESCRIPTION  
STARS        OFFICIAL  
nginx         Official build of Nginx.  
21011        [owl]
```

Pull an image:

```
sudo docker pull nginx
```

```
docksal/nginx          Nginx service image for Docksal  
1  
ubuntu@ip-172-31-12-81:~$ docker pull nginx  
Using default tag: latest  
latest: Pulling from library/nginx  
8c7716127147: Pull complete  
250b90fb2b9a: Pull complete
```

List images:

```
sudo docker images
```

4. Run Your First Container

Run nginx container:

```
sudo docker run -d -p 8080:80 --name mynginx nginx
```

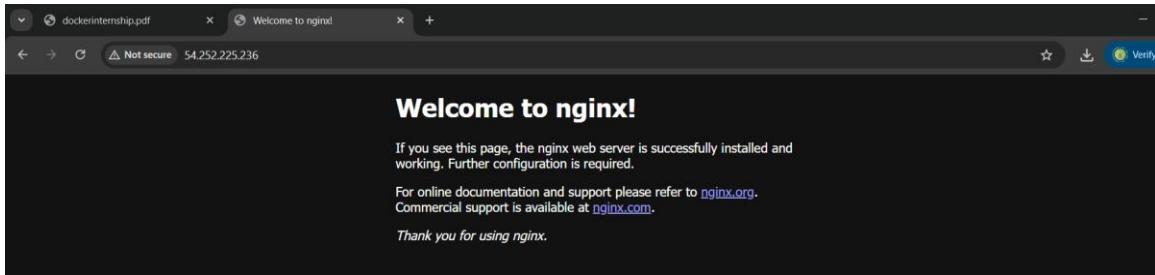
Verify running containers:

```
sudo docker ps
```

Access in browser:

<http://localhost:8080>

```
CONTAINER ID  IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES  
ubuntu@ip-172-31-12-81:~$ docker run -d -p 80:80 --name mynginx nginx  
1308d14c657785fe645adec227e708231ef34fc3dab5daea2a9a2fe851f99e88bd1  
ubuntu@ip-172-31-12-81:~$ sudo docker ps  
CONTAINER ID  IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES  
1308d14c6577  nginx     "/docker-entrypoint..."  10 seconds ago  Up 10 seconds  0.0.0.0:80->80/tcp, [::]:80->80/tcp  mynginx  
ubuntu@ip-172-31-12-81:~$ sudo docker ps -a  
CONTAINER ID  IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES  
1308d14c6577  nginx     "/docker-entrypoint..."  18 seconds ago  Up 17 seconds  0.0.0.0:80->80/tcp, [::]:80->80/tcp  mynginx  
301a7a548615  hello-world  "/hello"   8 minutes ago  Exited (0) 8 minutes ago  
ubuntu@ip-172-31-12-81:~$ |
```



5. Manage Containers

List containers:

```
sudo docker ps -a
```

Stop a container:

```
sudo docker stop mynginx
```

Remove a container:

```
sudo docker rm mynginx
```

Remove an image:

```
sudo docker rmi nginx
```

```
, [::]:80->80/tcp mynginx
ubuntu@ip-172-31-12-81:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
TS
1308d14c6577        nginx              "/docker-entrypoint...."   18 seconds ago    Up 17 seconds          0.0
.0.0:80->80/tcp, [::]:80->80/tcp mynginx
301a7a548615        hello-world       "/hello"           8 minutes ago     Exited (0) 8 minutes ago
                           dreamy_merkle

ubuntu@ip-172-31-12-81:~$ docker stop mynginx
mynginx
ubuntu@ip-172-31-12-81:~$ docker rm mynginx
mynginx
ubuntu@ip-172-31-12-81:~$ docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:f79cdde317d4d172a39297834034eed6dff5728a8e6d7a42f507504c23ecf8b8
Deleted: sha256:07ccdb7838758a4d52a9761636c385125a327355c0c94a6acff9babff938
Deleted: sha256:71b75b17511f67932ccf71e2046c6d1b4fe17a594134c765bf71c874dedc7027
Deleted: sha256:c76da83ebfb3e35184d1a7105d03cecfb144d07cb4dae12378392040fb44615
Deleted: sha256:f825da5ac2d31a2c717db4fa159b6728b33a94bc7285bd1dfebcbbe23ebd185
Deleted: sha256:31333e0f1ecf5940213f8b1ed4eb3e4d78d77fa5fb59c1cd05a6672690ed133c
Deleted: sha256:e72eb931613f8c1b5ba39f864877beee0780af9f9d20c5a02790d0146f0b012
Deleted: sha256:775e3183eed457c31a0855ac7a55b6e0cb8d40554e9524cb6d503cb2351e0b10
Deleted: sha256:1d46119d249f7719e1820e24a311aa7c453f166f714969cff89504678eaa447
ubuntu@ip-172-31-12-81:~$
```

6. Dockerfile Creation

Using nginx create a Simple Web Server Dockerfile

1. Create project folder:

```
mkdir mynginxapp
cd mynginxapp
```
2. Create sample HTML file and index.html

Step 2: Create Dockerfile

Step 3: Build Docker Image

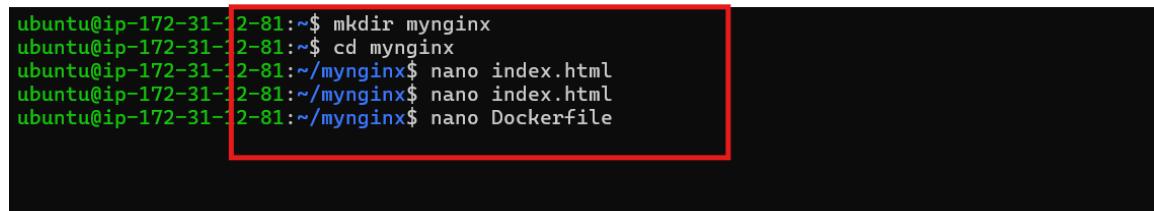
```
docker build -t mynginxapp .
```

Step 4: Run Docker Container

```
docker run -d -p 8080:80 mynginxapp
```

Open browser and go to <http://localhost:8080> to see the result.

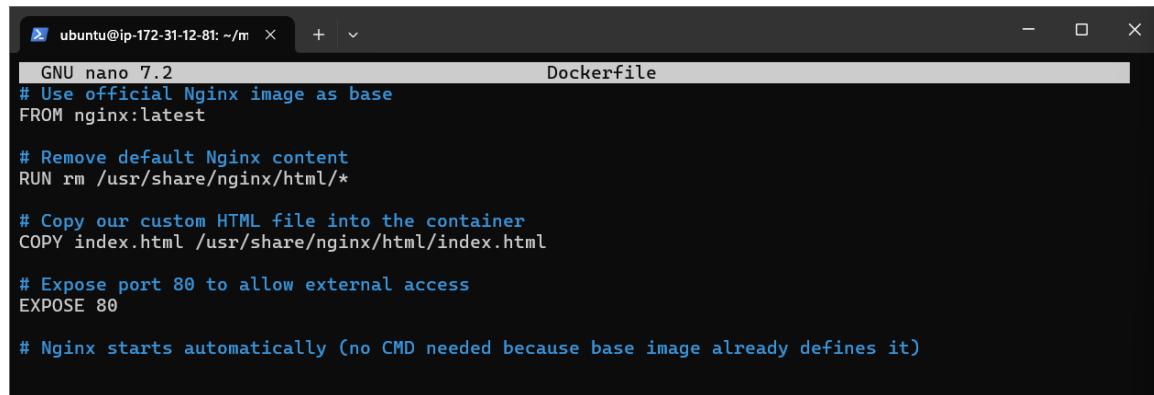
Screenshots;



```
ubuntu@ip-172-31-2-81:~$ mkdir mynginx
ubuntu@ip-172-31-2-81:~$ cd mynginx
ubuntu@ip-172-31-2-81:~/mynginx$ nano index.html
ubuntu@ip-172-31-2-81:~/mynginx$ nano Dockerfile
ubuntu@ip-172-31-2-81:~/mynginx$ nano Dockerfile
```



```
ubuntu@ip-172-31-12-81:~/m ~
GNU nano 7.2                               index.html
<!DOCTYPE html>
<html>
<head>
<title>My Custom Nginx</title>
</head>
<body>
    <h1>Hello from Nginx Docker Container!</h1>
</body>
</html>
```



```
ubuntu@ip-172-31-12-81:~/m ~
GNU nano 7.2                               Dockerfile
# Use official Nginx image as base
FROM nginx:latest

# Remove default Nginx content
RUN rm /usr/share/nginx/html/*

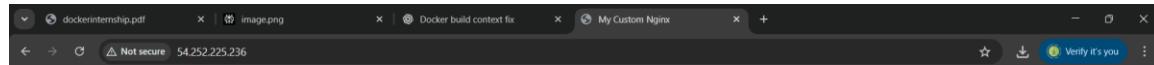
# Copy our custom HTML file into the container
COPY index.html /usr/share/nginx/html/index.html

# Expose port 80 to allow external access
EXPOSE 80

# Nginx starts automatically (no CMD needed because base image already defines it)
```



```
--> /16fe7da859
Successfully built 716fe7da859
Successfully tagged mynginx:latest
ubuntu@ip-172-31-12-81:~/mynginx$ docker run -d -p 80:80 mynginx
35824fed653b83e5b99dc1b0fb10d8fad4327e788cd6d8225ce78d57075c6b
ubuntu@ip-172-31-12-81:~/mynginx$ docker stop mynginx
```



Hello from Nginx Docker Container!

7. Docker Compose (Multi-container application)

First install docker compose.

```
35824fed653b
ubuntu@ip-172-31-12-81:~/myng.nx$ sudo apt install -y docker-compose
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-compose python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable
  python3-websocket
  python3=websocket
```

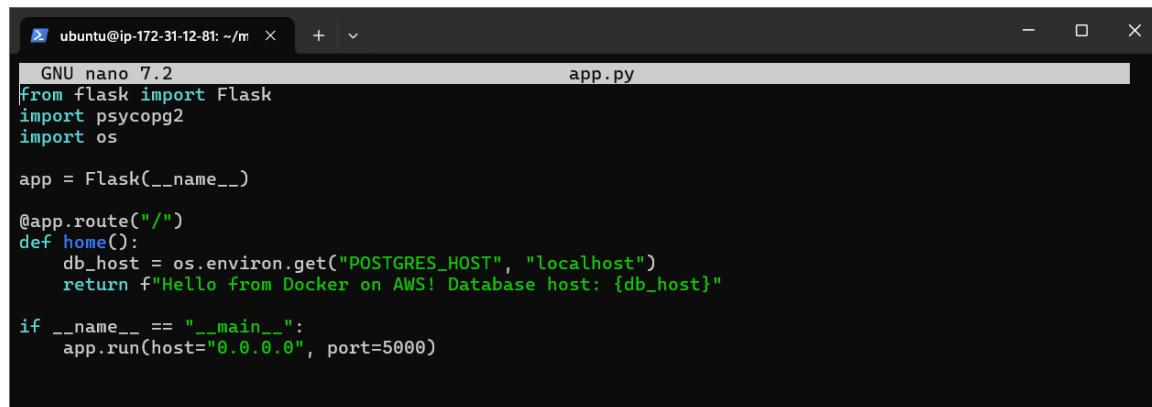
This sets up a Flask app + PostgreSQL database:

Create folder & change directory into it

Mkdir myapp

Cd myapp

Nano app.py



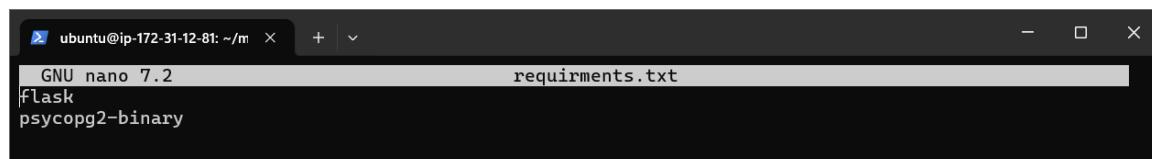
```
GNU nano 7.2                                         app.py
from flask import Flask
import psycopg2
import os

app = Flask(__name__)

@app.route("/")
def home():
    db_host = os.environ.get("POSTGRES_HOST", "localhost")
    return f"Hello from Docker on AWS! Database host: {db_host}"

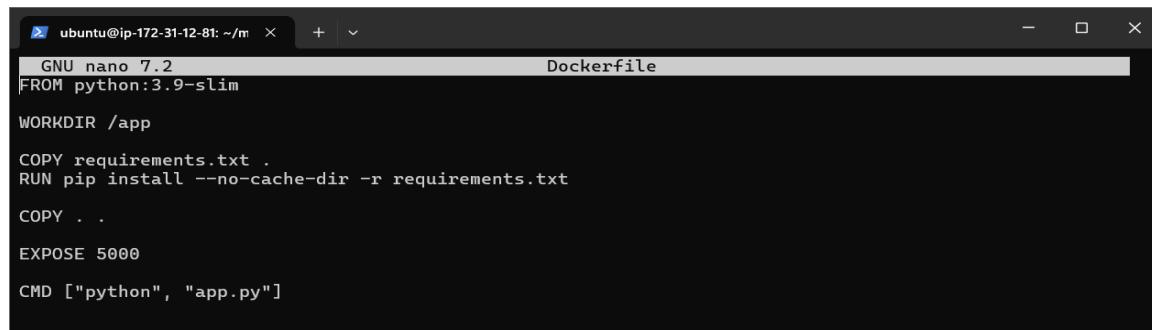
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Nano requirements.txt



```
GNU nano 7.2                                         requirements.txt
flask
psycopg2-binary
```

Nano Dockerfile



```
GNU nano 7.2                                         Dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
```

Nano docker-compose.yml

```
ubuntu@ip-172-31-12-81: ~ /m + - x
GNU nano 7.2                                            docker-compose.yml
version: "3.9"

services:
  web:
    build: .
    container_name: flask_app
    ports:
      - "80:5000"
    environment:
      - POSTGRES_HOST=db
    depends_on:
      - db

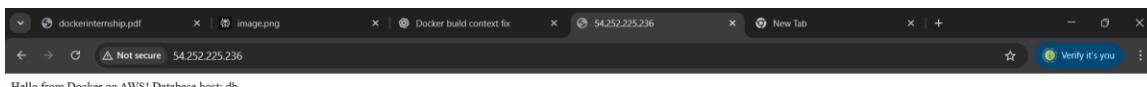
  db:
    image: postgres:15
    container_name: postgres_db
    restart: always
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: mydatabase
    ports:
      - "5432:5432"
    volumes:
      - db_data:/var/lib/postgresql/data

volumes:
  db_data:
```

Then start the application

Sudo docker-compose up -d or docker-compose up -d

```
Successfully built 2712110cc0c  
Successfully tagged myapp_web:latest  
ubuntu@ip-172-31-12-81:~/myapp$ docker-compose up -d  
Creating postgres_db ... done  
Creating flask_app ... done  
ubuntu@ip-172-31-12-81:~/myapp$ docker ps -a
```



Stop the application

Sudo docker compass down

8 Key Docker Concepts Summary

Concept	Description
Image	A lightweight, standalone package containing everything to run software.
Container	A running instance of an image.

Dockerfile	A script of instructions to build a Docker image.
Volume	Persistent storage for containers.
Port Mapping	Connects container ports to host machine ports.
Docker Compose	Defines and manages multi-container applications.