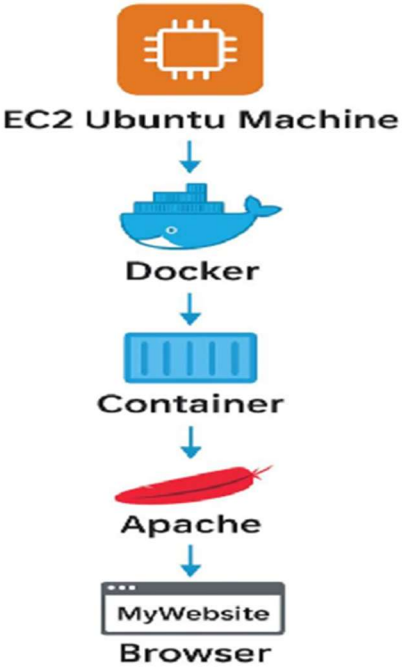
Lab: Deploy Docker on an EC2 Ubuntu Machine and Run a Web App in a Container

Objective

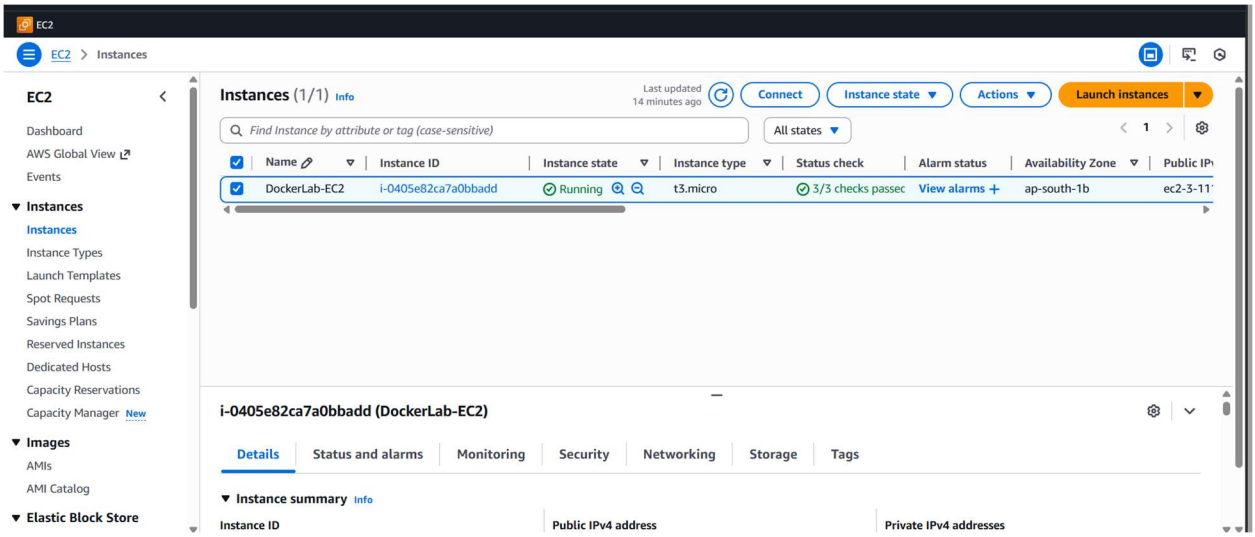
By the end of this lab, you will:

- Launch an Ubuntu EC2 instance.
- Install Docker on the instance.
- Run an Ubuntu container.
- Install Apache2 inside the container.
- Serve a simple web page through Apache running in the container.



Step 1: Launch EC2 Instance

1. Go to AWS Management Console → EC2 → Launch Instance.
2. Name it: DockerLab-EC2.
3. AMI: Ubuntu Server 24.04 LTS.
4. Instance type: t2.micro (Free tier eligible).
5. Key Pair: Select or create a new .pem key.
6. Network: Allow SSH (port 22) and HTTP (port 80) in Security Group.
7. Launch the instance.
8. Copy the Public IPv4 address.



Step 2: Connect to EC2 via SSH

ssh -i "john.pem" ubuntu@ec2-3-111-36-191.ap-south-1.compute.amazonaws.com

```
C:\Users\chauh\Downloads>
C:\Users\chauh\Downloads>ssh -i "john.pem" ubuntu@ec2-3-111-36-191.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-3-111-36-191.ap-south-1.compute.amazonaws.com (64:ff9b::36f:24bf)' can't be established.
ED25519 key fingerprint is SHA256:6iIZyKCkqF0YtgCDarUA44uvw/xh2piU6QuJTwY58PM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-111-36-191.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Oct 30 15:14:03 UTC 2025

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   25.5% of 6.71GB   Processes:             110
  Memory usage: 23%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.6.140

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
```

Step 3: Install Docker

1. **Update OS**
   sudo apt update
   sudo apt upgrade

(Why: refresh package lists and install available security/bug fixes.)

2. Install prerequisites/dependencies:

sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

(Why: needed to add external apt repos over HTTPS and manage keys.)

3.**Create keyrings dir and download Docker's GPG key:**

sudo mkdir -p /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

(Why: apt uses this to verify Docker packages.)

4. Add the Docker apt repository (this picks the correct Ubuntu codename):

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \

https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" \

| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

(Why: tells apt where to fetch Docker packages.)

5.Update package index and install Docker packages:

sudo apt update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

(Why: installs Docker Engine, containerd, Buildx, and compose plugin.)

6.Verify Docker works:

sudo docker --version

(Expected: something like Docker version 24.x.x, build ...)

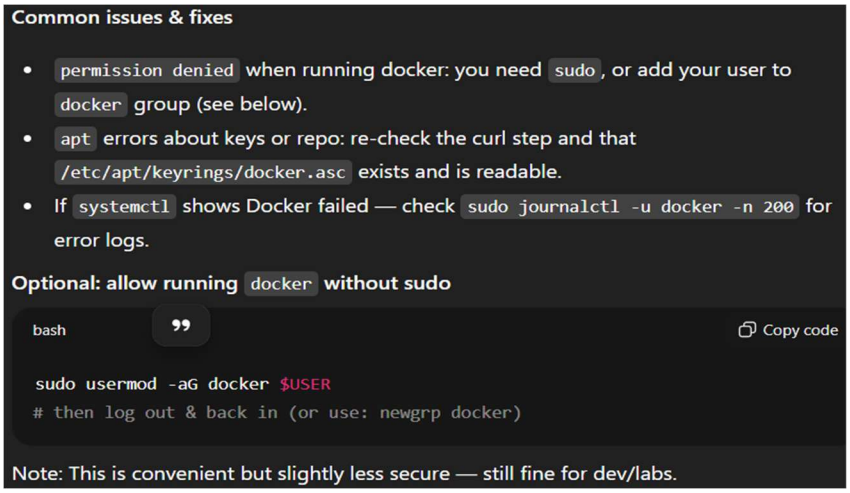
7.Enable & start Docker daemon:

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl status docker # check health

(Why: ensures Docker starts on boot. status shows if it's running.)



Step 4 — Run a Docker container

You'll pull a plain Ubuntu image and run an ephemeral container to practise.

1. (Optional) Login to Docker Hub:

docker login

(You only need this if pulling/pushing private images.)

2. Pull Ubuntu image:

sudo docker pull ubuntu

(Why: pulls official minimal Ubuntu image.)

3. Run container detached interactive:

sudo docker run -itd --name myubuntu ubuntu

(What each flag means:

- -i keep STDIN open
- -t allocate a tty
- -d run detached (background)
- --name myubuntu friendly name)

4. Check containers:
   sudo docker ps -a

Step 5 — Install Apache2 inside the container (detailed)

You'll open a shell inside container and install Apache like on any Ubuntu.

1. Enter the container shell:
   sudo docker exec -it myubuntu /bin/bash
   (Now you are inside the container (root shell)).

2. Inside container — update & install apache:
   sudo apt update && sudo apt upgrade -y
   sudo apt install apache2 -y
   (Why: installs Apache package in container filesystem.)
3. Start Apache:
   sudo service apache2 start

Optional checks:
After that, verify Apache is running:
sudo service apache2 status

Step 6 — Create a web page
Inside the container:
cd /var/www/html
echo "MyWebsite - Running on Docker Container" | sudo tee /var/www/html/index.html
cat index.html

(This replaces the default page with your message.)

Step 7 — Expose container to EC2 public IP

Containers run isolated; to access Apache from outside you must map container port 80 to host port 80.

1. Stop and remove old container (if you want to recreate with ports):

sudo docker stop myubuntu

sudo docker rm myubuntu

(Why: you can't change port mapping of a running container — recreate it with -p.)

2. Run a new container with port mapping:

sudo docker run -itd -p 80:80 --name webapp ubuntu

(-p 80:80 maps host port 80 → container port 80.

If EC2 already has a service on host port 80, pick another host port (e.g. -p 8080:80) and use that in browser.)

3. Install Apache inside new container:

sudo docker exec -it webapp bash

inside:

sudo apt update && sudo apt install apache2 -y

sudo service apache2 start

echo "MyWebsite - Dockerized Apache" >sudo tee /var/www/html/index.html

exit

Step 8 — Test the web app

From your local PC:

curl http://<EC2-Public-IP> (Example : curl http:// 3.111.36.191)

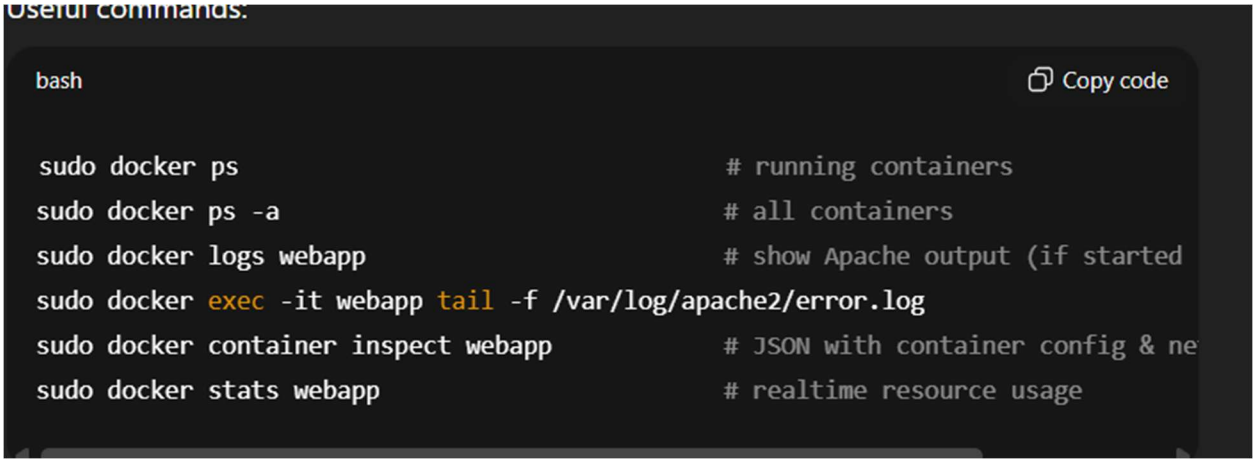
```
done.
root@1a2458b4e5db:/# service apache2 start
 * Starting Apache httpd web server apache2                                AH00558: apache2: Could not rel
in name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
 *
root@1a2458b4e5db:/# echo "MyWebsite - Dockerized Apache" > /var/www/html/index.html
root@1a2458b4e5db:/# exit
exit
ubuntu@ip-172-31-6-140:~$ sudo ufw status
Status: inactive
ubuntu@ip-172-31-6-140:~$ curl http:// 3.111.36.191
curl: (3) URL rejected: No host part in the URL
MyWebsite - Dockerized Apache
ubuntu@ip-172-31-6-140:~$ curl http://3.111.36.191
MyWebsite - Dockerized Apache
ubuntu@ip-172-31-6-140:~$
```

Or open browser:

http://<EC2-Public-ip> (Example: http://3.111.36.191)

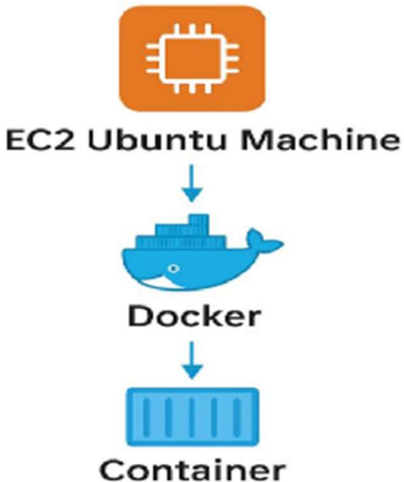
You should see:

Step 9 — Monitor the container (how & why)



Lab Completed!

You now have an EC2 instance running Docker, hosting an Apache-based web application inside a container.


EC2 Ubuntu Machine
Docker
Container