Docker Weekend Task - 1

1. What is the need of DevOps?

-> DevOps is needed to:

- Bridge the gap between development and operations teams.
- Improve deployment speed through automation and CI/CD.
- Increase software quality via continuous testing and monitoring.
- Enhance collaboration and communication across the lifecycle.
- Enable faster feedback, bug fixes, and customer satisfaction.
- Reduce failures and mean time to recovery (MTTR).
- Support Agile and lean practices for faster innovation.

2. Difference between Virtualization vs Containerization vs Bare Metal

Feature	Bare Metal	Virtualization	Containerization
Definition	Physical machine without any layer	Running multiple OS on a single machine via hypervisor	Running apps in isolated user-space environments
Resource Usage	Full resources used by one OS	More overhead due to multiple OS	Lightweight – shares OS kernel
Speed	Fastest (no abstraction layer)	Slower than bare metal	Faster than VMs
Isolation	N/A (only one environment)	Each VM has its own OS	Process-level isolation, uses host OS
Scalability	Limited	Scalable, but with resource overhead	Highly scalable, low resource footprint
Startup Time	Slow (booting physical machine)	Minutes (boot guest OS)	Seconds (start container)

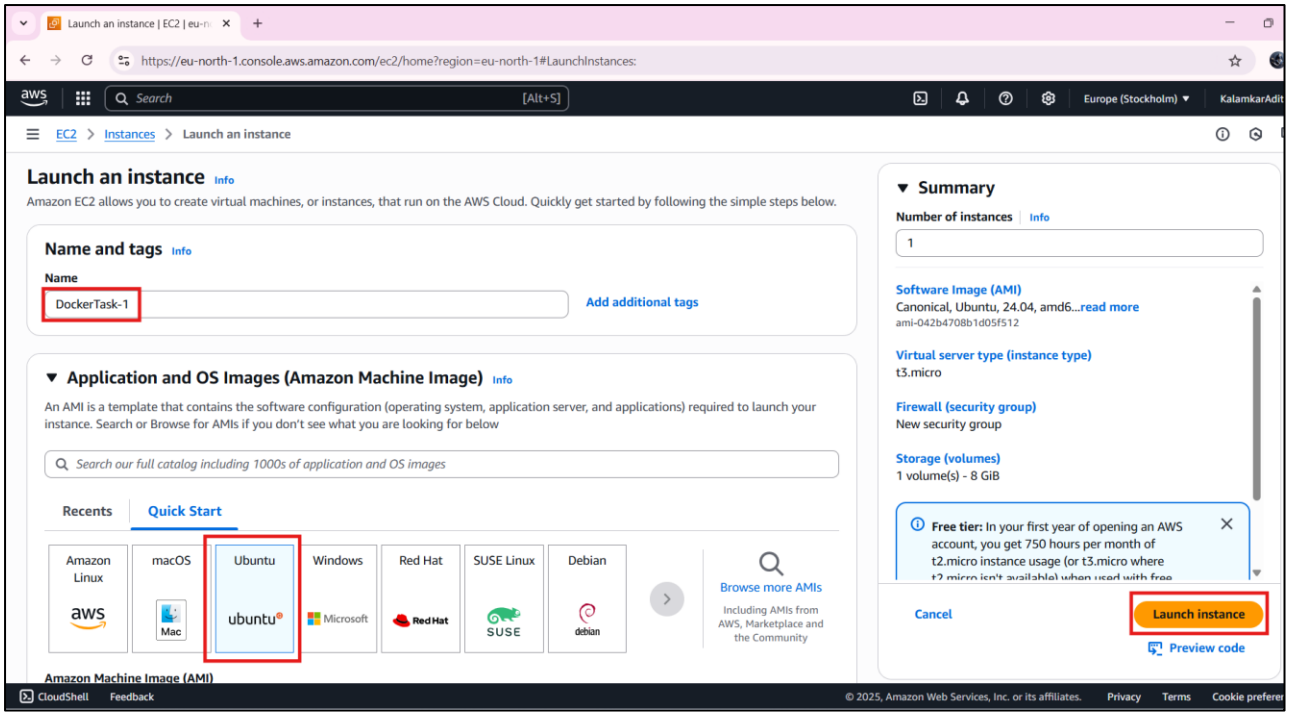
Use Case	High-performance apps, databases	Multi-OS needs, legacy apps	Microservices, CI/CD pipelines, cloud-native applications

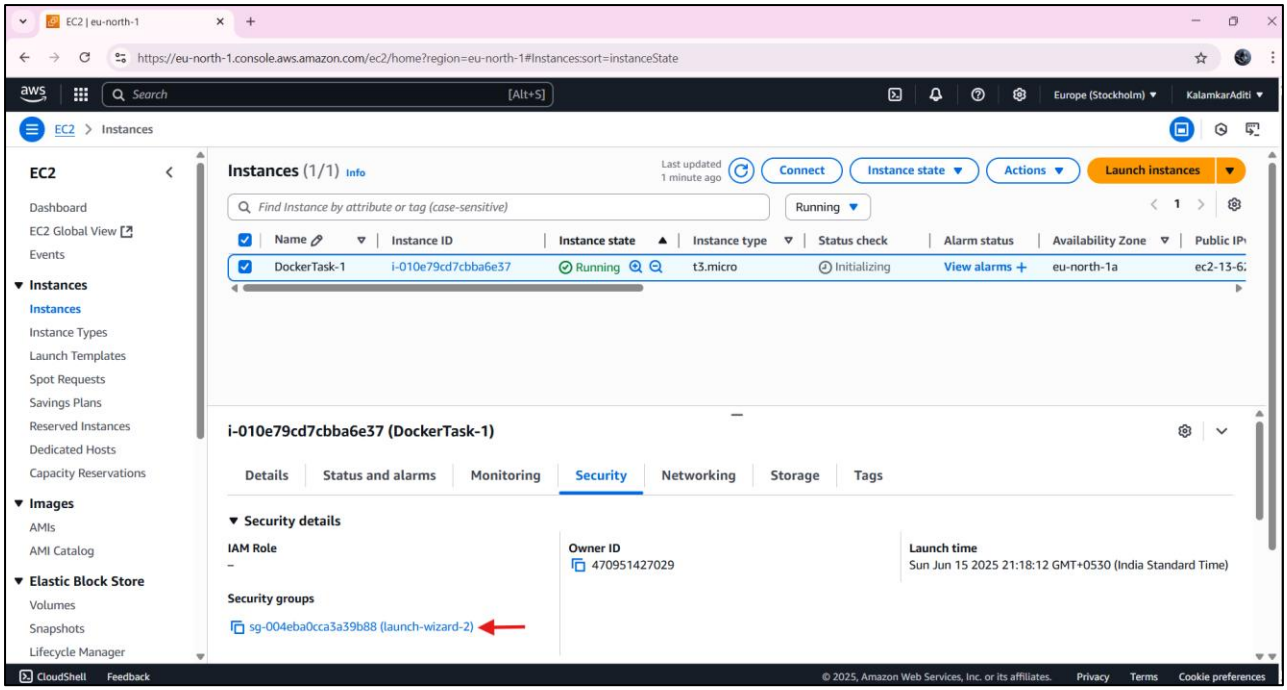
3. Create an httpd (apache) container, inside that container create 2 more html pages (eg. home.html and about.html) which will be accessible from the browser.

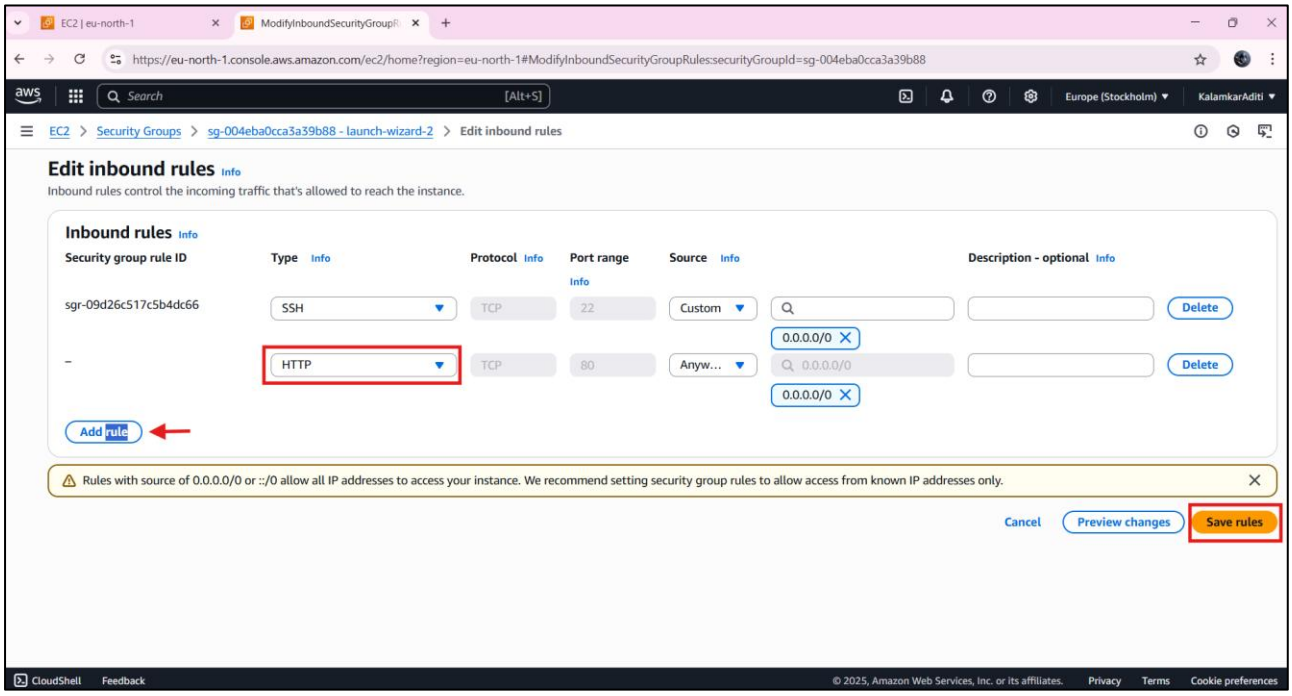
✅ **Goal**

- **Run an httpd(apache) container on EC2 Linux**
- **Serve home.html and about.html**
- **Use port 80**
- **Accessible via http://<EC2_PUBLIC_IP>/<pages>.html**

Step 1: Launch EC2 and allow inbound traffic on port 80.



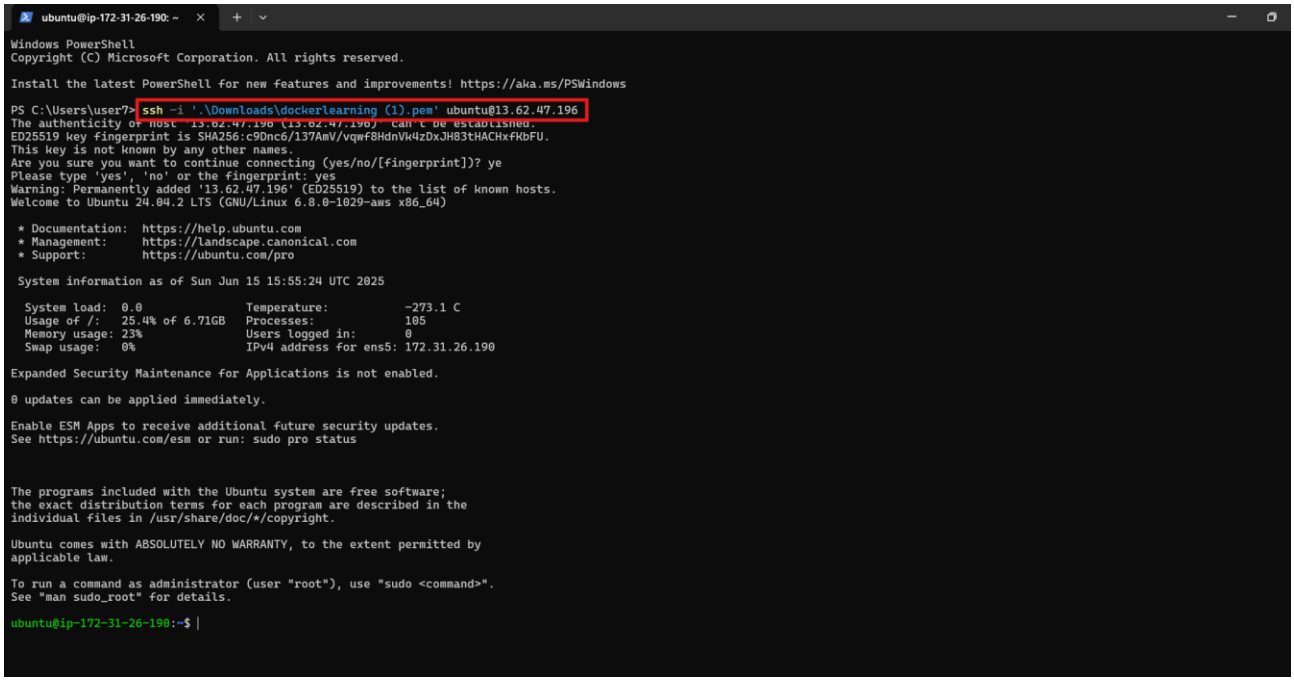




Connect to EC2 via SSH using Powershell.

-Command to connect via ssh:

ssh -i <path-to-pem-file> <user>@<ec2-public-ip>



Step 2: Install and Start Docker

sudo apt-get update

→ Updates the system's package list.

sudo apt-get install docker.io -y

→ Installs Docker engine with automatic confirmation.

sudo systemctl start docker.service *or* sudo service docker start

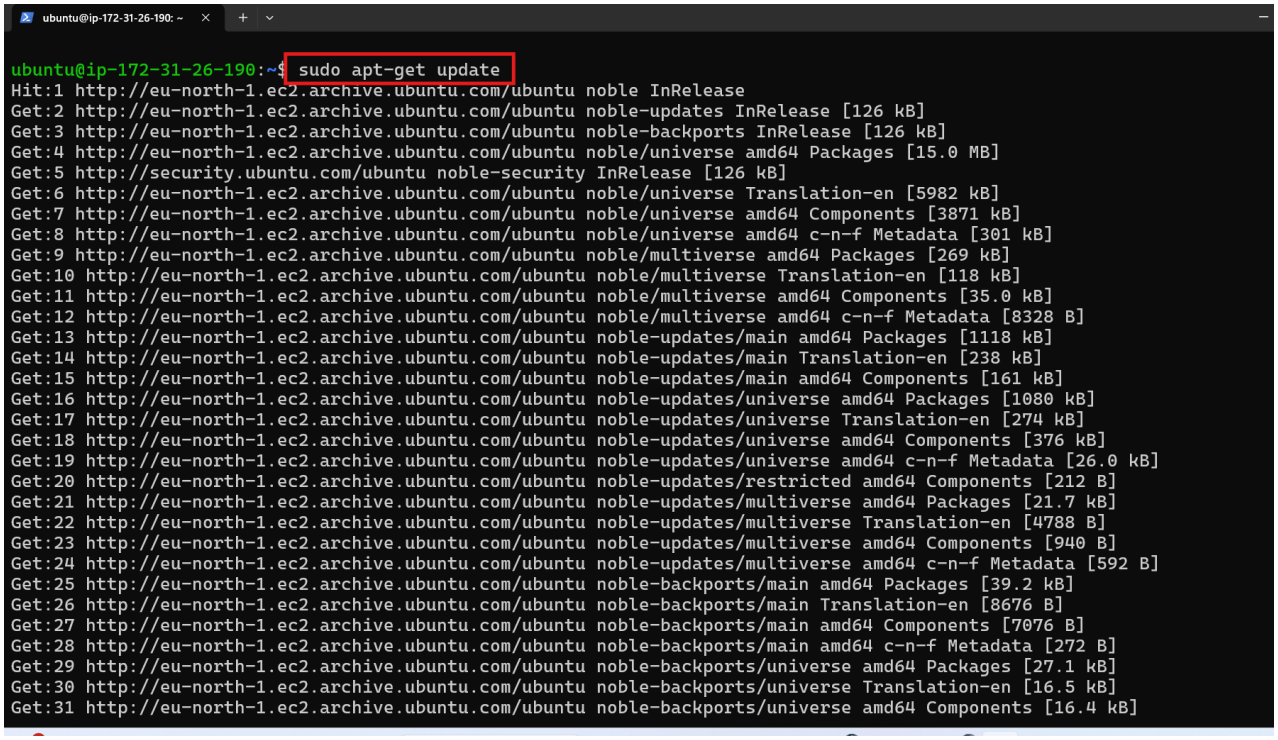
→ Starts the Docker service.

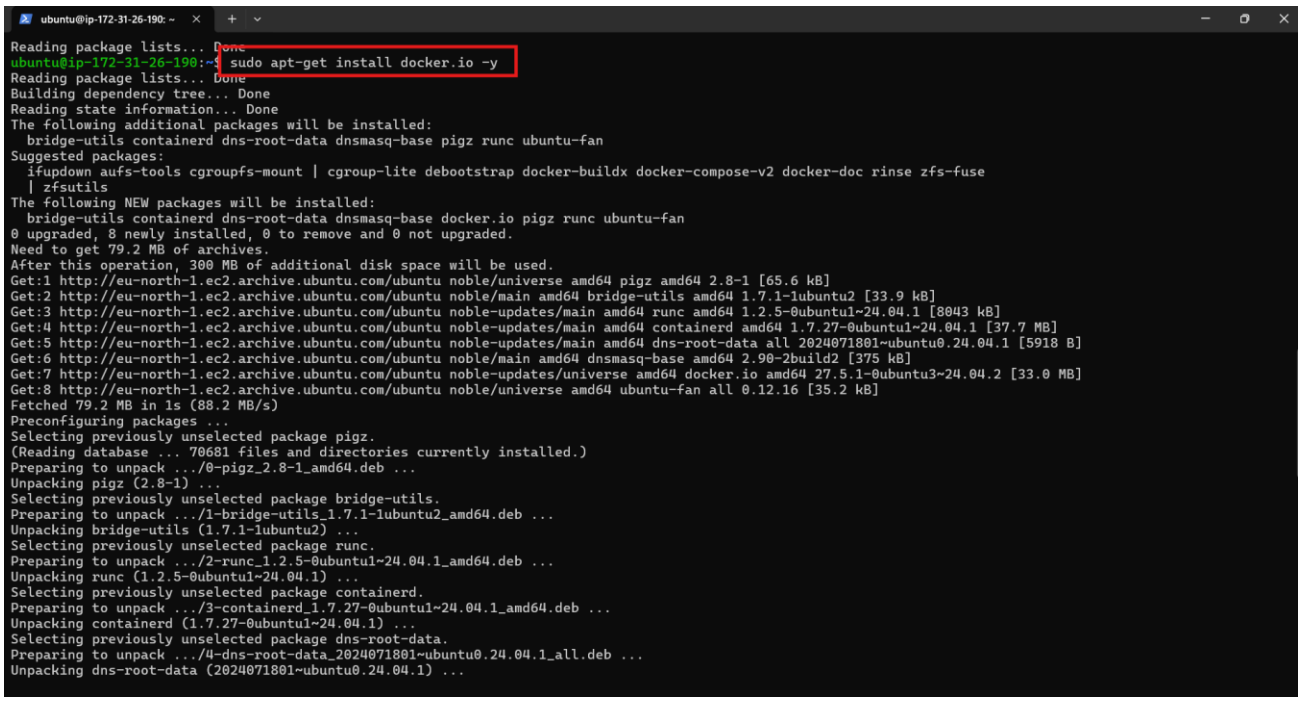
sudo systemctl enable docker.service *or* sudo service docker enable

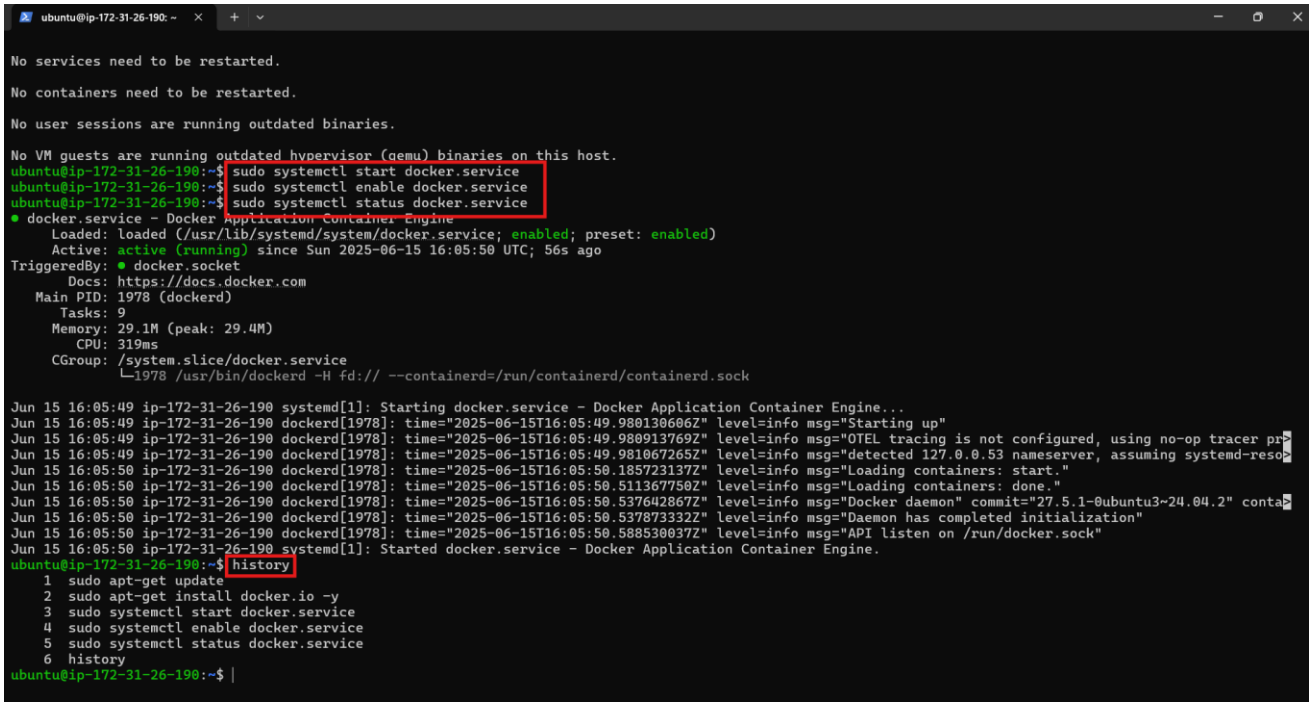
→ Enables Docker to start automatically on system boot.

sudo systemctl status docker.service *or* sudo service docker status

→ Verifies the current status of the Docker service.


ubuntu@ip-172-31-26-190: ~
ubuntu@ip-172-31-26-190:~$ sudo apt-get update
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1118 kB]
Get:14 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [238 kB]
Get:15 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Get:16 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1080 kB]
Get:17 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [274 kB]
Get:18 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [376 kB]
Get:19 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [26.0 kB]
Get:20 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:21 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [21.7 kB]
Get:22 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [4788 B]
Get:23 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:24 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [592 B]
Get:25 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.2 kB]
Get:26 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [8676 B]
Get:27 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7076 B]
Get:28 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:29 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [27.1 kB]
Get:30 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [16.5 kB]
Get:31 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [16.4 kB]


ubuntu@ip-172-31-26-190: ~
Reading package lists... Done
ubuntu@ip-172-31-26-190:~$ sudo apt-get install docker.io -y
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
The following NEW packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 79.2 MB of archives.
After this operation, 300 MB of additional disk space will be used.
Get:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.2.5-0ubuntu1~24.04.1 [8043 kB]
Get:4 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.27-0ubuntu1~24.04.1 [37.7 MB]
Get:5 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801~ubuntu0.24.04.1 [5918 B]
Get:6 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]
Get:7 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 27.5.1-0ubuntu3~24.04.2 [33.0 MB]
Get:8 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 79.2 MB in 1s (88.2 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 70681 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7.1-1ubuntu2_amd64.deb ...
Unpacking bridge-utils (1.7.1-1ubuntu2) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.2.5-0ubuntu1~24.04.1_amd64.deb ...
Unpacking runc (1.2.5-0ubuntu1~24.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.27-0ubuntu1~24.04.1_amd64.deb ...
Unpacking containerd (1.7.27-0ubuntu1~24.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2024071801~ubuntu0.24.04.1_all.deb ...
Unpacking dns-root-data (2024071801~ubuntu0.24.04.1) ...



Step 3 : Create an Apache (httpd) container and add HTML pages (home.html and about.html)

sudo docker pull httpd

→ Download the latest httpd (Apache) image

sudo docker run -d --name apache -p 80:80 httpd

→ Start Apache container in detached interactive mode

sudo docker exec -it apache bash

→ Open terminal inside the Apache container

apt update && apt install nano -y

→ Make nano available to edit files inside the container

cd /usr/local/apache2/htdocs/

→ Navigate to the directory where Apache serves web content, Change to Apache's web root directory

nano home.html

→ Open nano editor to create/edit the home page

nano about.html

→ Open nano editor to create/edit the about page

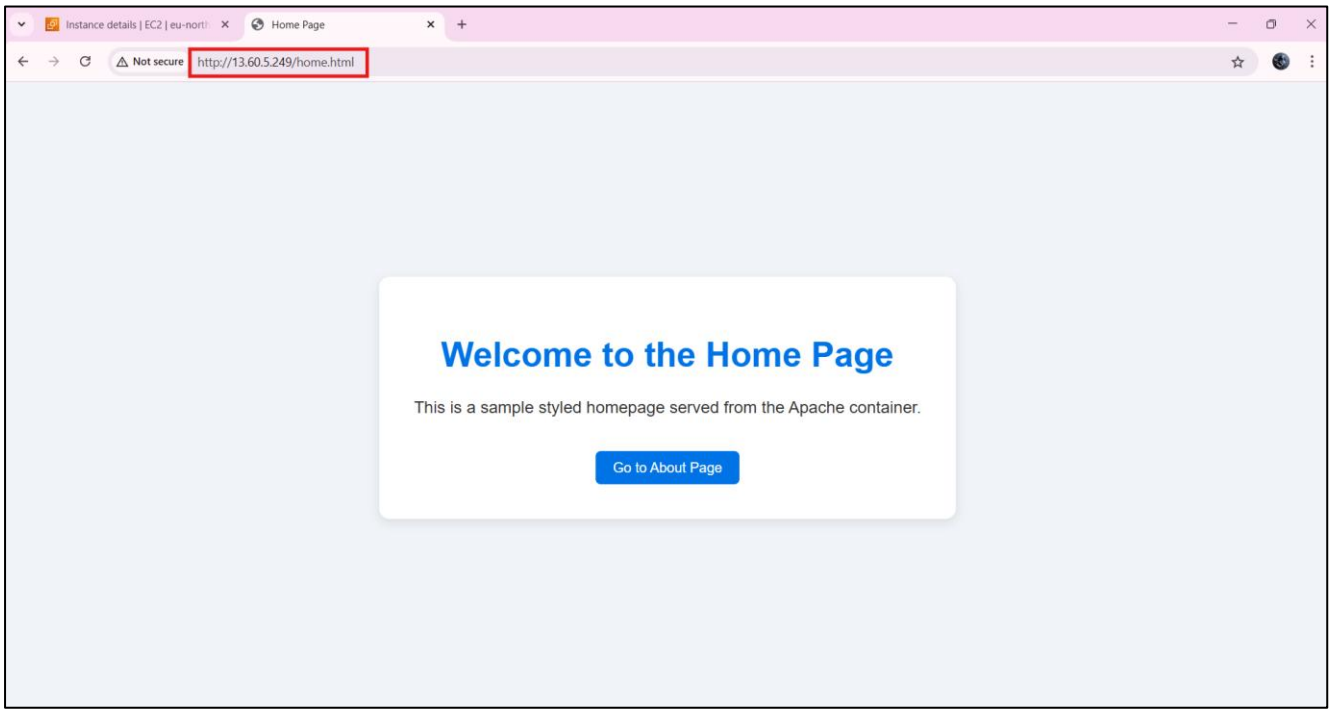
Save with Ctrl + X, then Y, then exit with ENTER.

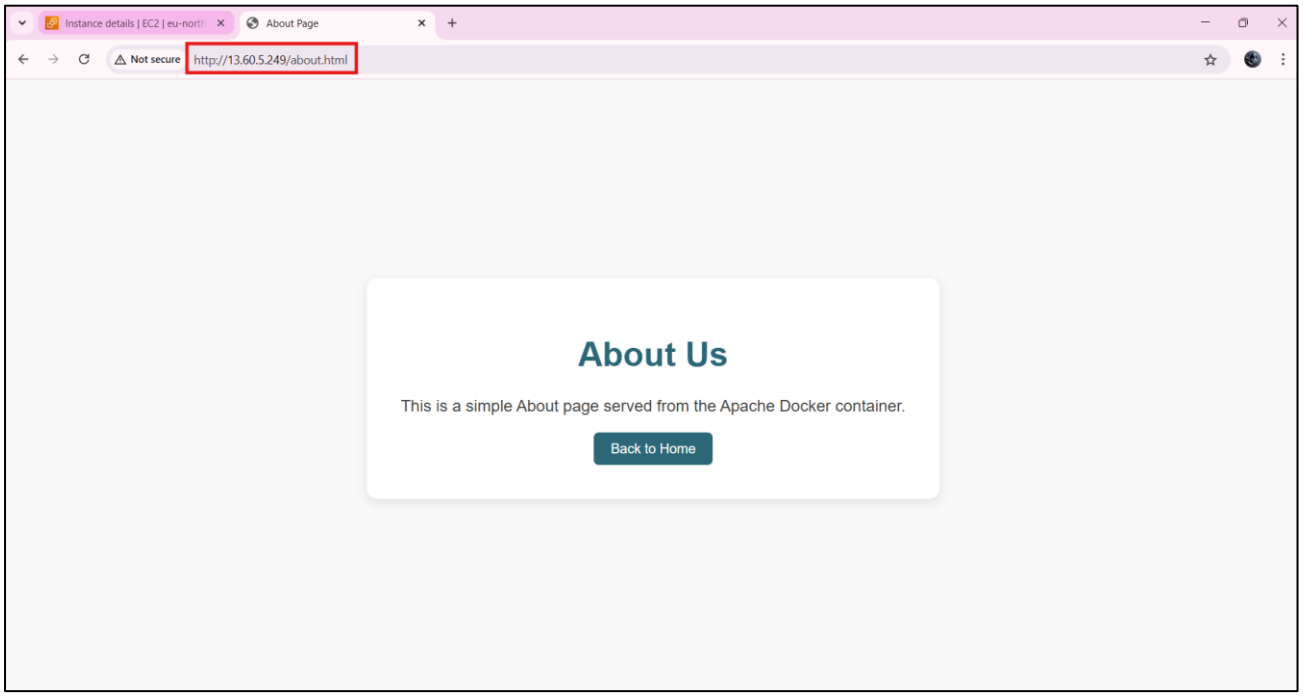
```
root@ip-172-31-26-190: /hom
ubuntu@ip-172-31-26-190:~$ sudo su
root@ip-172-31-26-190:/home/ubuntu# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
dad67da3f26b: Pull complete
d0a755bf09a1: Pull complete
4f4fb700ef54: Pull complete
be5c5a616c3a: Pull complete
d1042d58e186: Pull complete
c06cec1379c2: Pull complete
Digest: sha256:f6557a77ee2f16c50a5ccbb2564a3fd56087da311bf69a160d43f73b23d3af2d
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
root@ip-172-31-26-190:/home/ubuntu# docker run -d --name apache -p 80:80 httpd
45a001392e77e996b1c856bb9556445528b87b73b1928c738dc8bd5938624c75
root@ip-172-31-26-190:/home/ubuntu# docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    bf07fec943ec   4 months ago   148MB
root@ip-172-31-26-190:/home/ubuntu# docker exec -it apache bash
root@45a001392e77:/usr/local/apache2# apt update && apt install nano -y
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8793 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [756 B]
```

Step 4 : Access Pages in Browser

These should open in your browser, served by Apache running inside Docker on EC2.

- http://<EC2-public-IP>/home.html
- http://<EC2-public-IP>/about.html





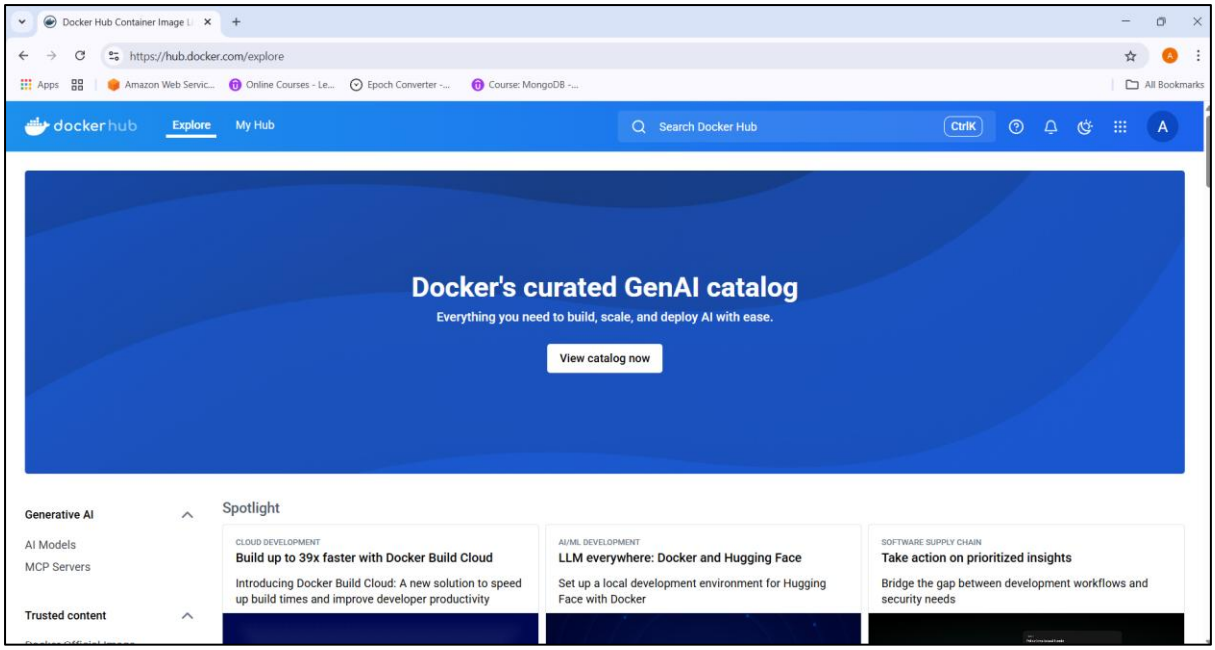
4. Create image from httpd container created from above question and push that image to

a. Docker hub

=> Step 1: Create Docker Hub Account & Repository

- **1. Create Docker Hub Account**

1. Open your browser and go to:
   👉 https://hub.docker.com
2. Click on “Sign Up” (top-right corner)
3. Fill the form:
   - Username (choose your Docker ID – unique, lowercase)
   - Email Address
   - Password (use a strong password)
4. Click “Sign Up”
5. Verify your email address (check your inbox for a confirmation link and click it)

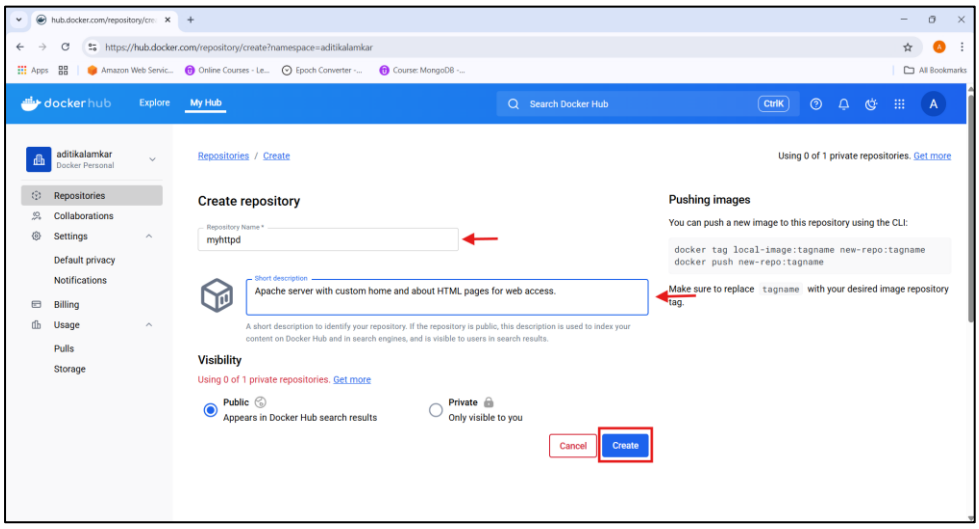


🔹 2. Create a Repository in Docker Hub

1. Once logged in, go to:

   👉 https://hub.docker.com/repositories

2. Click **“Create Repository”** (top-right corner)
3. Fill in the details:
4. **Repository Name** – e.g., myhttpd
5. **Visibility** – choose **Public**
6. Optional: Add **description** and **tags**
7. Click **“Create”**



Step 2: Commit, Tag, and Push Docker Image to Docker Hub

sudo su

→ Switch to the root user (superuser) for running Docker commands without sudo prefix.

docker container ls

→ List all running Docker containers, showing container IDs, names, ports, and status.

docker commit apache myhttpd:v1

→ Create a new Docker image (myhttpd:v1) from the current state of the apache container.

docker tag myhttpd:v1 dockerhub-username / docker-repo-name:tag

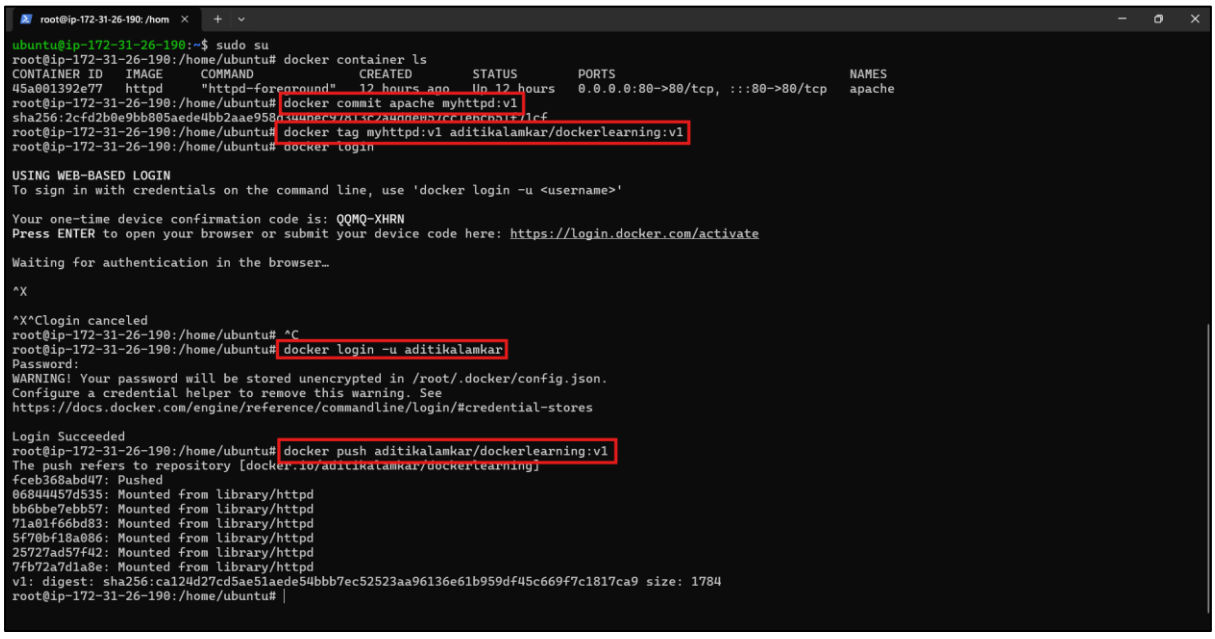
→ Tag the image with your Docker Hub username and repository name so it can be pushed (myhttpd:v1 → aditikalamkar/ docker-repo-name:tag).

docker login -u <dockerhub-username>

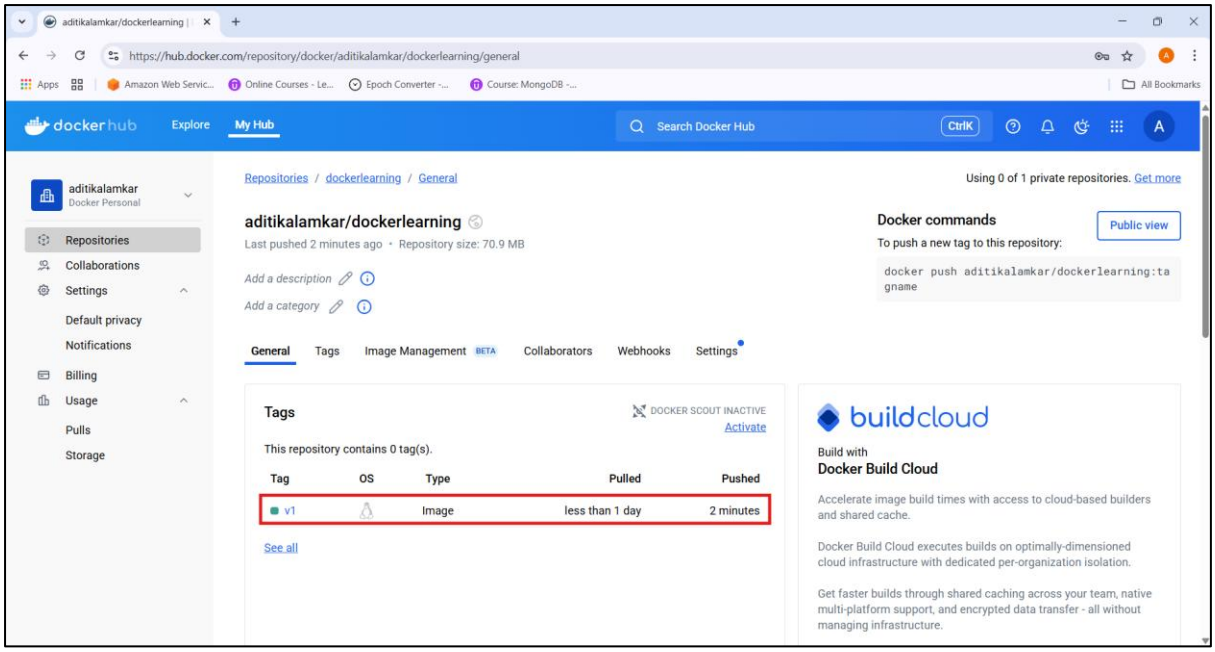
→Login to Docker Hub using your Docker ID (<docker-hub-username). You'll be prompted for your password.

docker push dockerhub-username /docker-repo-name:tag

→ Push the tagged image to your Docker Hub repository so it can be accessed or pulled from anywhere.



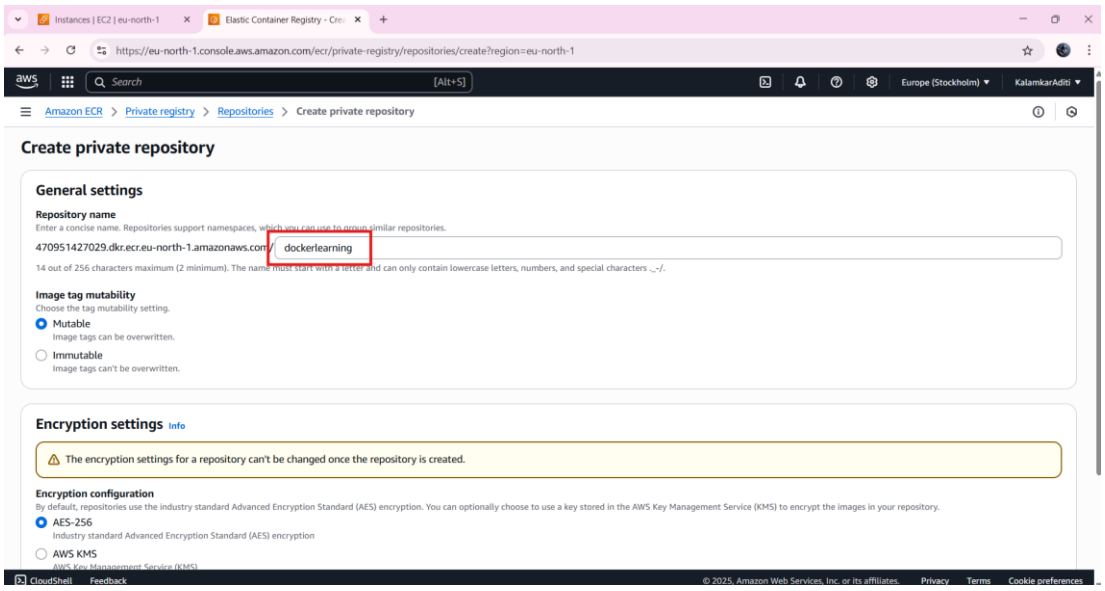
→ The final output will be visible in your Docker Hub repository, where the image will appear under your account.



b. ECR

Step 1: Create a Repository in ECR

1. Go to the AWS Management Console
2. Navigate to **ECR → Repositories**
3. Click **"Create repository"**
4. Choose:
    - Repository name: dockerlearning
    - Leave defaults and click **"Create repository"**



Step 2: Install AWS CLI on Ubuntu

sudo su

→ Switch to the root user (superuser) for running Docker commands without sudo prefix.

apt update

→Updates the local package list to get the latest version information.

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

→ Downloads the official AWS CLI v2 installation ZIP file from AWS and saves it as awscliv2.zip.

apt install unzip -y

→ Installs the unzip utility required to extract the downloaded ZIP file.

unzip awscliv2.zip

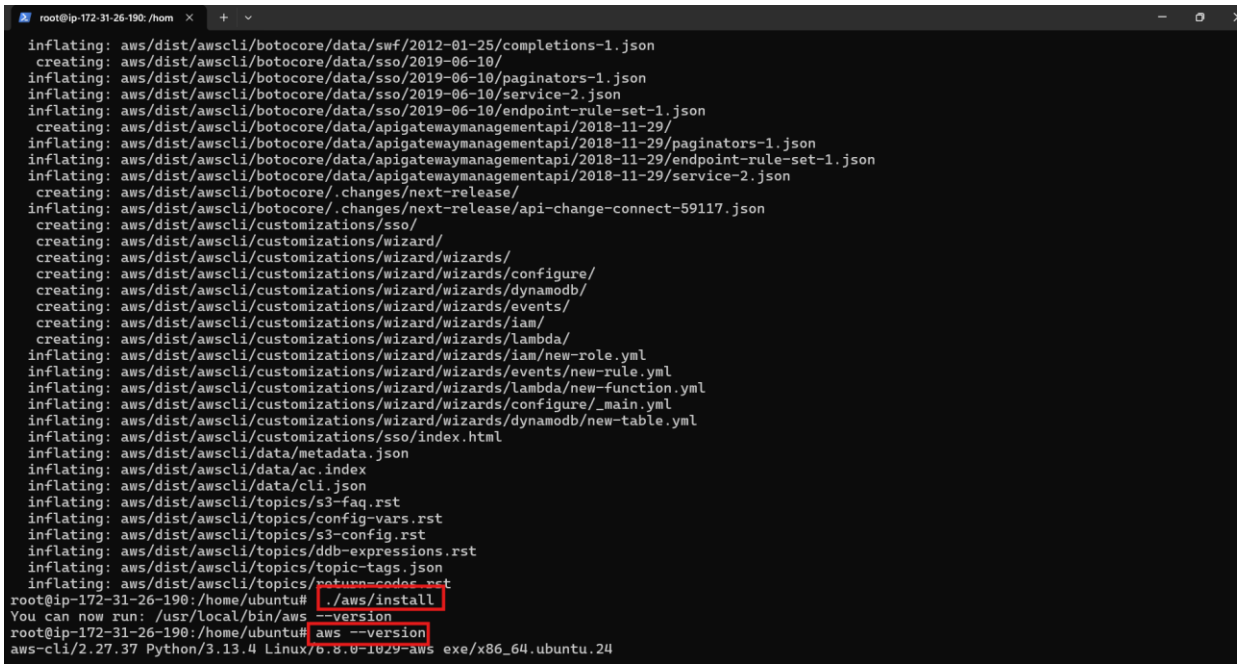
→ Unzips the AWS CLI installation package to a directory named aws/.

./aws/install

→ Installs AWS CLI v2 to the default location (/usr/local/bin/aws).

aws –version

→ Displays the installed AWS CLI version to confirm successful installation.



Step 3: Push Docker Image to AWS ECR

sudo su

→ Switch to the root user (superuser) for running Docker commands without sudo prefix.

docker images or docker image ls

→ Displays all locally available Docker images to confirm the image name and tag (e.g., httpd:latest).

aws configure

→ Configures your AWS credentials (Access Key, Secret Key, Region) for CLI authentication.

aws ecr get-login-password --region eu-north-1 | docker login --username AWS --password-stdin 470951427029.dkr.ecr.eu-north-1.amazonaws.com

→ Logs in Docker to your AWS ECR repository using your credentials and region.

docker tag <img-name>:latest 470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning:latest

#Example:

docker tag dockerlearning:latest 470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning:latest

→ Tags the local httpd:latest image with your ECR repository URI and tag (latest) so it can be pushed.

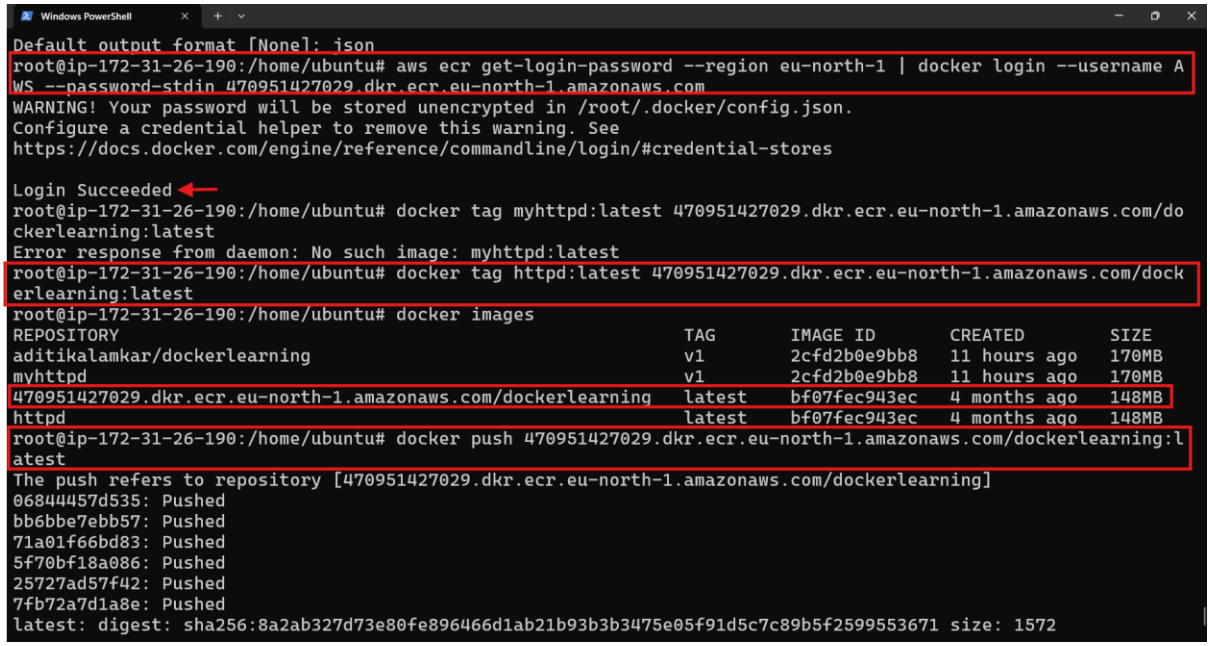
docker images

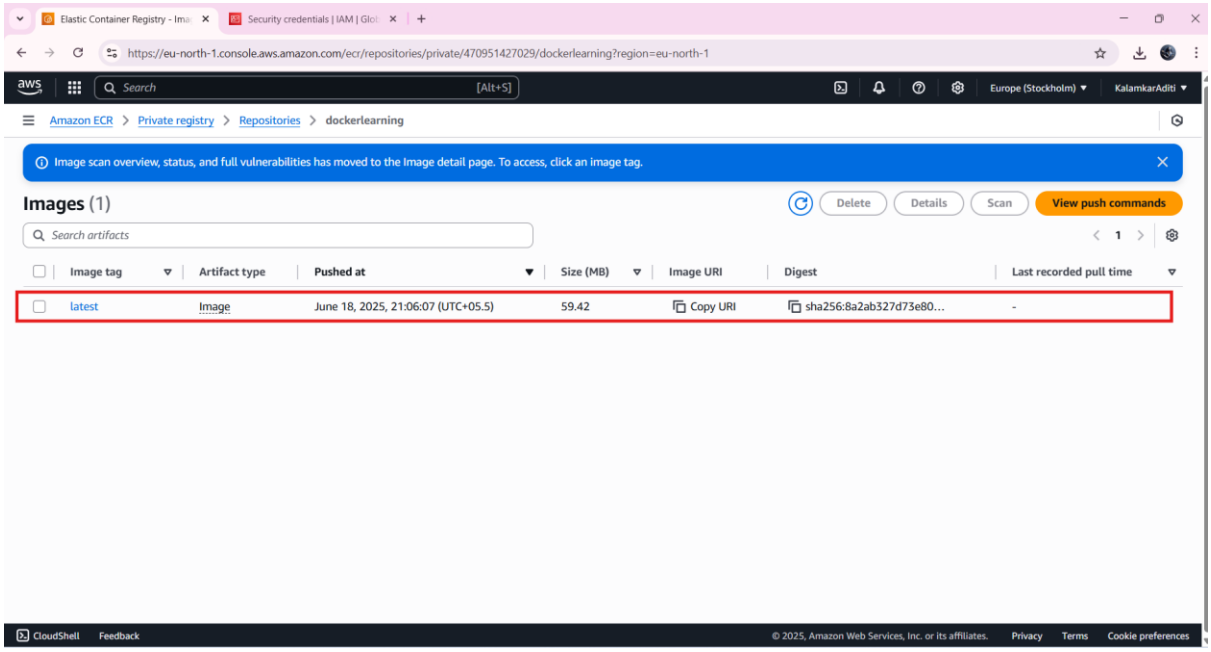
→ Confirms that the image has been correctly tagged with the ECR repository address.

(Optional) Verify the Tagged Image.

docker push 470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning:latest

→ Uploads the tagged Docker image to your AWS ECR repository for use in deployments.





5. Create the list of commands with its usage and examples.

- **Docker Installation & Service**

  sudo apt update

  → Update ubuntu OS.

  sudo apt install docker.io -y

  → Installs Docker Engine on Ubuntu.

sudo systemctl start docker

→ Starts the Docker service.

sudo systemctl enable docker

→ Enables Docker to start on boot.

sudo systemctl status docker

→ check status of the Docker service.

- **Docker Image Commands**

Sudo su

→ Switch to the root user (superuser) for running Docker commands without sudo prefix.

docker pull <image-name>

→ Pulls the official image from Docker Hub.

Example: (Apache HTTPD image)

docker pull httpd

docker images

→ Lists all locally available Docker images.

docker rmi <image_id> or docker rmi <image_name>

→ Removes a Docker image from your system.

docker run -d --name <container-name> -p <host-port>:<container-port> <image-name>

→ Runs an img container in detached mode with <container-name> and maps port 80.

#Example:

docker run -d --name apache -p 80:80 httpd

→ Runs an httpd container in detached mode with name apache and maps port 80.

docker container ls

→ Lists all running containers.

docker container ls -a

→ Lists **all** containers (running + stopped).

docker stop <container-name>

→ Stops the container

docker rm <container-name>

→ Removes the container.