

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)

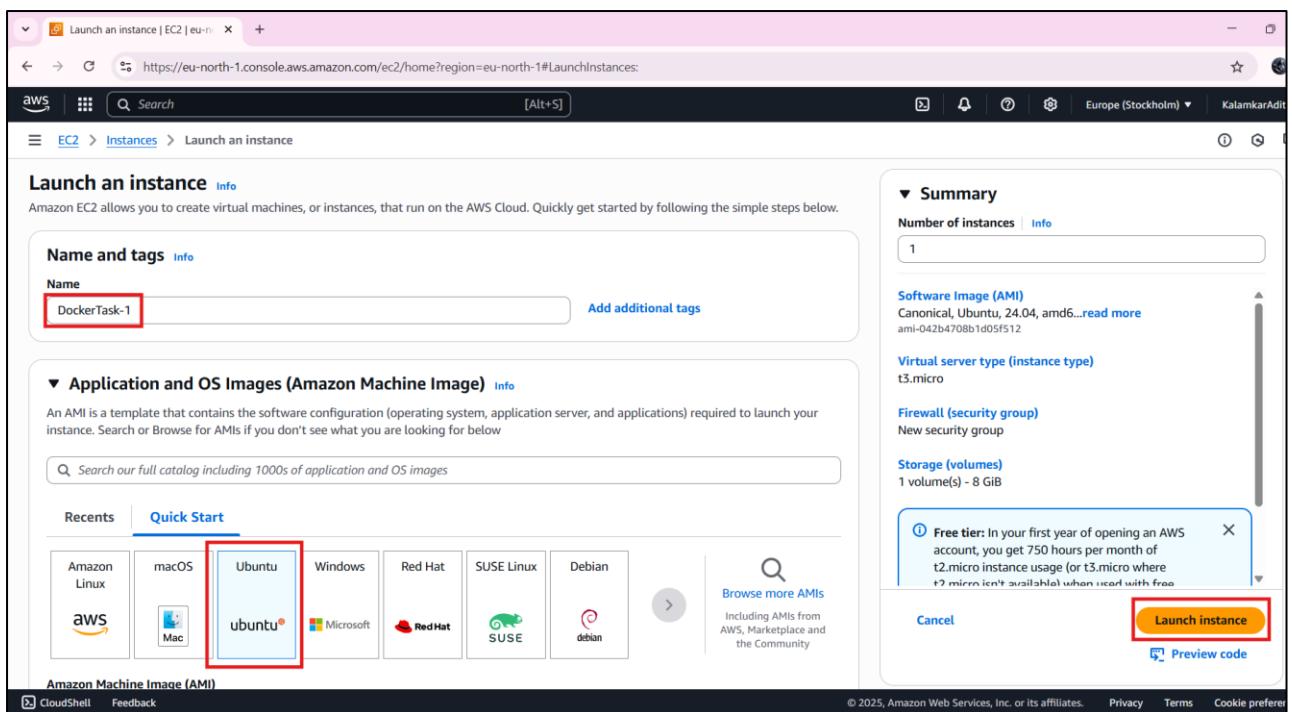
Use Case	High-performance apps, databases	Multi-OS needs, legacy apps	Microservices, CI/CD pipelines, cloud-native applications
-----------------	----------------------------------	-----------------------------	---

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.



The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Images, and Elastic Block Store. The main area displays a table of instances. One instance, "DockerTask-1" (ID: i-010e79cd7cbba6e37), is selected and shown in more detail. The "Security" tab is active, showing the "Security groups" section which lists "sg-004eba0cca3a39b88 (launch-wizard-2)". A red arrow points to this entry. Other tabs include Details, Status and alarms, Monitoring, Networking, Storage, and Tags.

The screenshot shows the "Edit inbound rules" page for a security group. It lists two rules: one for SSH (TCP port 22) and another for HTTP (TCP port 80). A red box highlights the "HTTP" rule. Below the table, there's an "Add rule" button with a red arrow pointing to it. At the bottom, there's a warning message about allowing all IP addresses and buttons for "Cancel", "Preview changes", and "Save rules".

Connect to EC2 via SSH using Powershell.

-Command to connect via ssh:

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

```
ubuntu@ip-172-31-26-190: ~ + - 
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\user7> ssh -i '.\Downloads\dockerlearning (1).pem' ubuntu@13.62.47.196
The authenticity of host '13.62.47.196 (13.62.47.196)' can't be established.
ED25519 key fingerprint is SHA256:c9Dm6c137AmV/vqwf8HdmVh4zDxJH83tAChxFKBFU.
This key is not known by me. Are you sure you want to continue connecting (yes/no/[fingerprint])? ye
Please type 'yes' 'no' or the fingerprint: yes
Warning: Permanently added '13.62.47.196' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

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

System information as of Sun Jun 15 15:55:24 UTC 2025

System load: 0.0 Temperature: -273.1 C
Usage of /: 25.4% of 6.71GB Processes: 105
Memory usage: 23% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.26.196

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 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>".
See "man sudo_root" for details.

ubuntu@ip-172-31-26-190:~$ |
```

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:~$ 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:~$ sudo apt-get install docker.io -y
Reading package lists... Done
ubuntu@ip-172-31-26-190:~$ 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 main amd64 bridge-utils amd64 1.7.1-lubuntu2 [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 main amd64 dnsmasq-base amd64 2.98-2build2 [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-lubuntu2_amd64.deb ...
Unpacking bridge-utils (1.7.1-lubuntu2) ...
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) ...
```

```

ubuntu@ip-172-31-26-190:~ + 
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-172-31-26-190:~$ sudo systemctl start docker.service
ubuntu@ip-172-31-26-190:~$ sudo systemctl enable docker.service
ubuntu@ip-172-31-26-190:~$ sudo systemctl status docker.service
● docker.service - Docker Application Container Engine
    Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
      Active: active (running) since Sun 2025-06-15 16:05:58 UTC; 56s ago
TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 1978 (dockerd)
      Tasks: 9
     Memory: 29.1M (peak: 29.4M)
        CPU: 319ms
       CGroup: /system.slice/docker.service
           └─1978 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jun 15 16:05:49 ip-172-31-26-190 systemd[1]: Starting docker.service - Docker Application Container Engine...
Jun 15 16:05:49 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:49.980130606Z" level=info msg="Starting up"
Jun 15 16:05:49 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:49.980913769Z" level=info msg="OTEL tracing is not configured, using no-op tracer pr
Jun 15 16:05:49 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:49.981067265Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-reso
Jun 15 16:05:50 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:50.185723137Z" level=info msg="Loading containers: start "
Jun 15 16:05:50 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:50.511367758Z" level=info msg="Loading containers: done."
Jun 15 16:05:50 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:50.537642867Z" level=info msg="Docker daemon" commit="27.5.1-0ubuntu3~24.04.2" conta
Jun 15 16:05:50 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:50.537873332Z" level=info msg="Daemon has completed initialization"
Jun 15 16:05:50 ip-172-31-26-190 dockerd[1978]: time="2025-06-15T16:05:50.588530037Z" level=info msg="API listen on /run/docker.sock"
Jun 15 16:05:50 ip-172-31-26-190 systemd[1]: Started docker.service - Docker Application Container Engine.
ubuntu@ip-172-31-26-190:~$ history
1 sudo apt-get update
2 sudo apt-get install docker.io -y
3 sudo systemctl start docker.service
4 sudo systemctl enable docker.service
5 sudo systemctl status docker.service
6 history
ubuntu@ip-172-31-26-190:~$ 

```

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

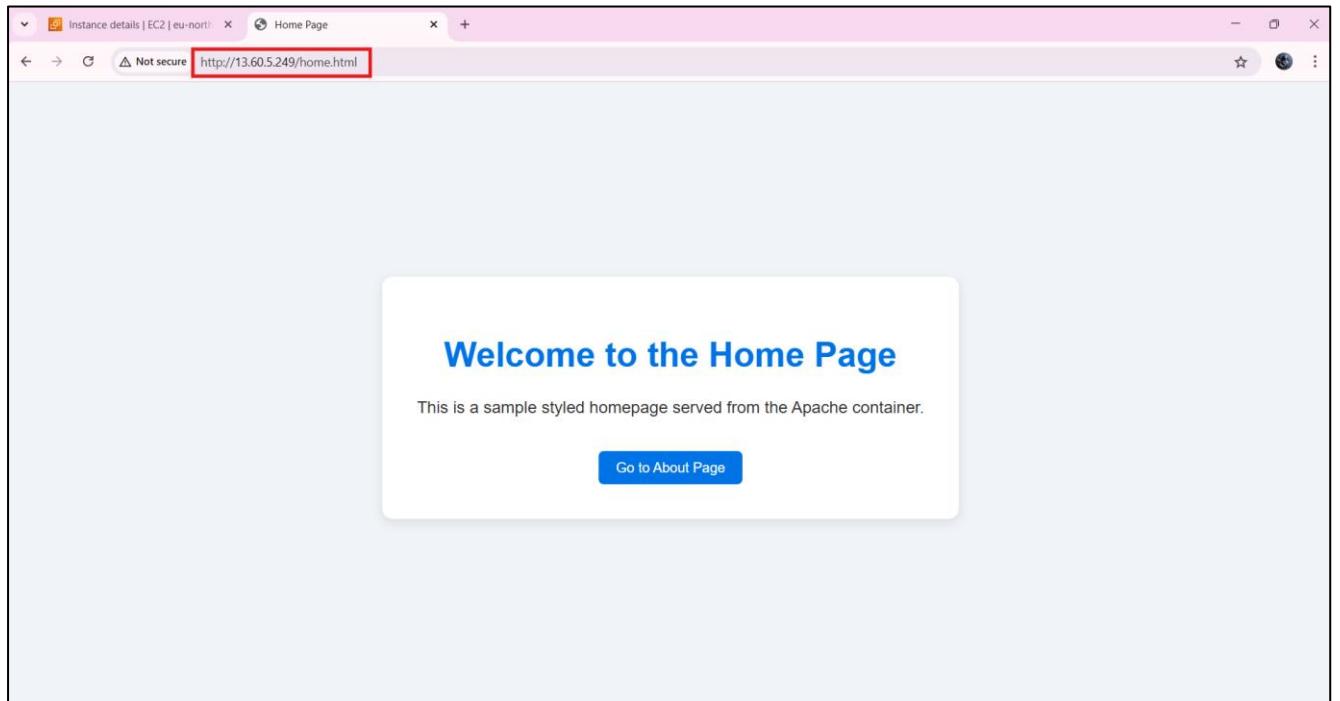
Save with Ctrl + X, then Y, then exit with ENTER.

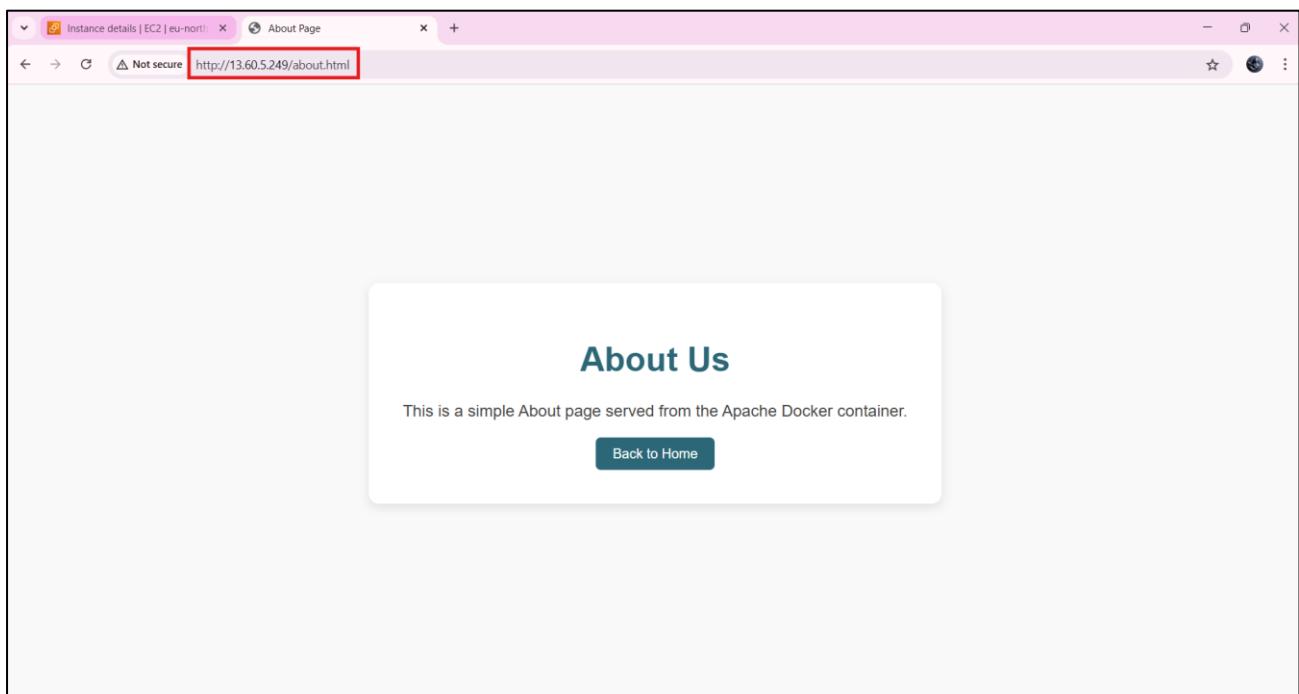
```
root@ip-172-31-26-190:/home/ubuntu$ sudo su
root@ip-172-31-26-190:/home/ubuntu# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
dad67da3f26b: Pull complete
d0a755bf09a1: Pull complete
4f4fb700ef54: Pull complete
be5c5a616c3a: Pull complete
d1042d58e186: Pull complete
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
45a001392e77e996b1c856bb9556445528b87d73d1928c738dc8bd5938624c75
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>

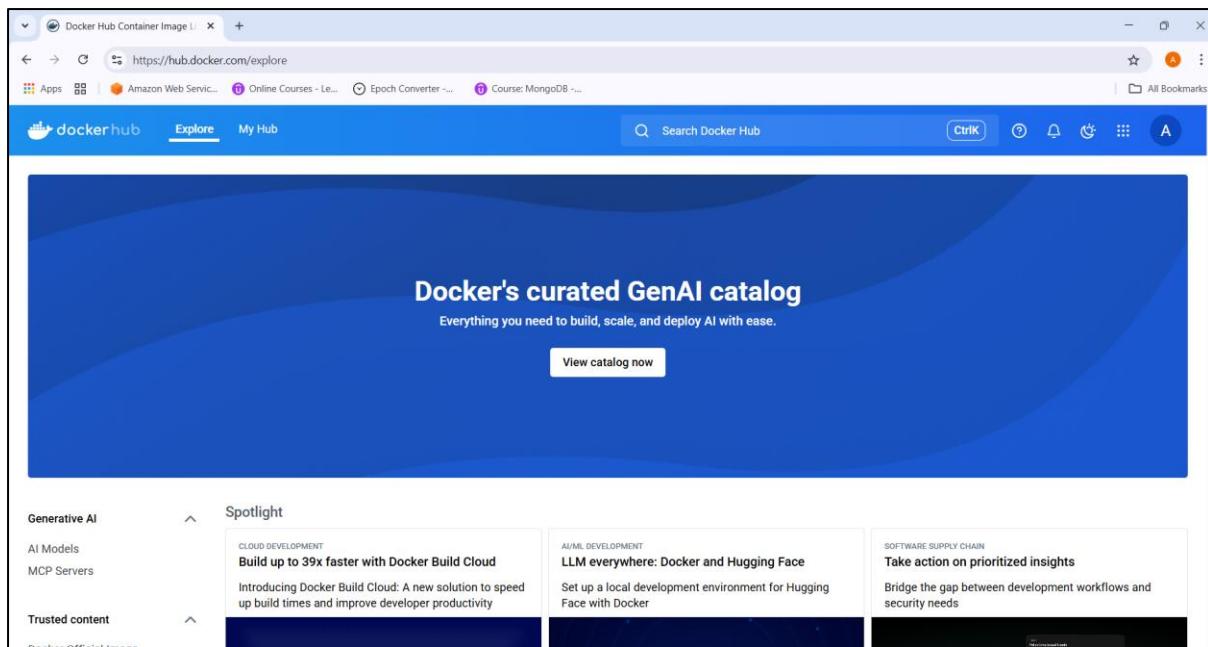
2. Click on “Sign Up” (top-right corner)

3. Fill the form:

- Username (choose your Docker ID – unique, lowercase)
- Email Address
- Password (use a strong password)

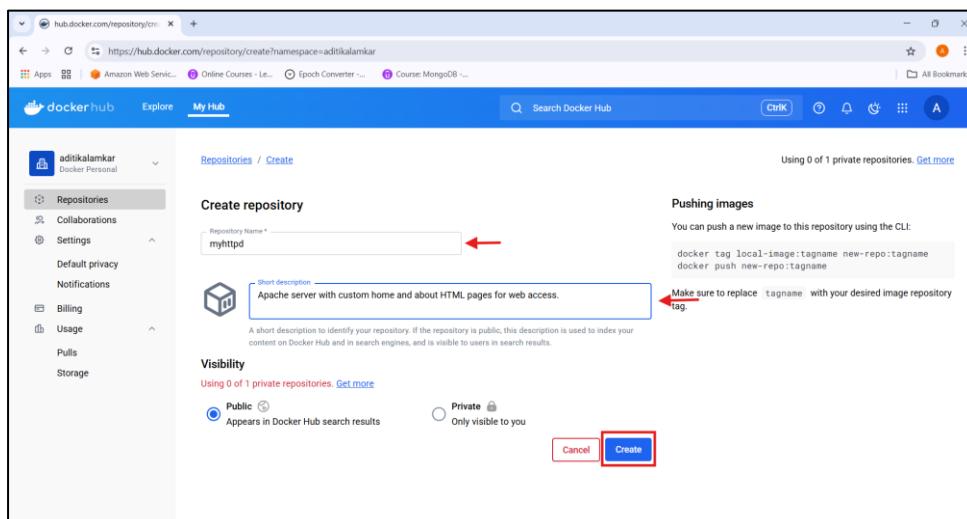
4. Click “Sign Up”

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 screenshot shows a terminal window with the following command history:

```
root@ip-172-31-26-190:/home/ubuntu# sudo su
root@ip-172-31-26-190:/home/ubuntu# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
45a001392e77        "httpd-foreground"   12 hours ago      In 12 hours       0.0.0.0:80->80/tcp, ::80->80/tcp   apache
root@ip-172-31-26-190:/home/ubuntu# docker commit apache myhttpd:v1
sha256:2cfdbbe9b805ae4bb2aae958a0a9c7a0d9e0c4f
root@ip-172-31-26-190:/home/ubuntu# docker tag myhttpd:v1 aditikalamkar/dockerlearning:v1
root@ip-172-31-26-190:/home/ubuntu# docker login
Using web-based login
To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: QMQQ-XHRN
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate
Waiting for authentication in the browser...
^X

^X*Login canceled
root@ip-172-31-26-190:/home/ubuntu# ^C
root@ip-172-31-26-190:/home/ubuntu# docker login -u aditikalamkar
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
root@ip-172-31-26-190:/home/ubuntu# docker push aditikalamkar/dockerlearning:v1
The push refers to repository [docker.io/aditikalamkar/dockerlearning]
fcb268abd7: Pushed
06944457d535: Mounted from library/httpd
bb6bbe7eb57: Mounted from library/httpd
71a01f66bd83: Mounted from library/httpd
5f70bf18a086: Mounted from library/httpd
25727ad57f42: Mounted from library/httpd
7fb72a7d1a8e: Mounted from library/httpd
v1: digest: sha256:ca124d27cd5ae51ae0e54bbb7ec52523aa96136e61b959df45c669f7c1817ca9 size: 1784
root@ip-172-31-26-190:/home/ubuntu#
```

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

The screenshot shows a Docker Hub repository page for 'aditikalamkar/dockerlearning'. The 'General' tab is active, displaying the repository's details: last pushed 2 minutes ago, repository size 70.9 MB. There are options to add a description or category. A table lists the tag 'v1' with columns Tag, OS, Type, Pulled, and Pushed. The 'Pushed' column shows '2 minutes'. A sidebar on the right is titled 'buildcloud' and describes building with Docker Build Cloud.

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**"

The screenshot shows the 'Create private repository' wizard in the AWS ECR console. In the 'General settings' step, the repository name 'dockerlearning' is specified. Below this, the 'Encryption settings' section is visible, containing a note that settings can't be changed once created. The 'Encryption configuration' section shows 'AES-256' selected as the encryption standard.

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
```

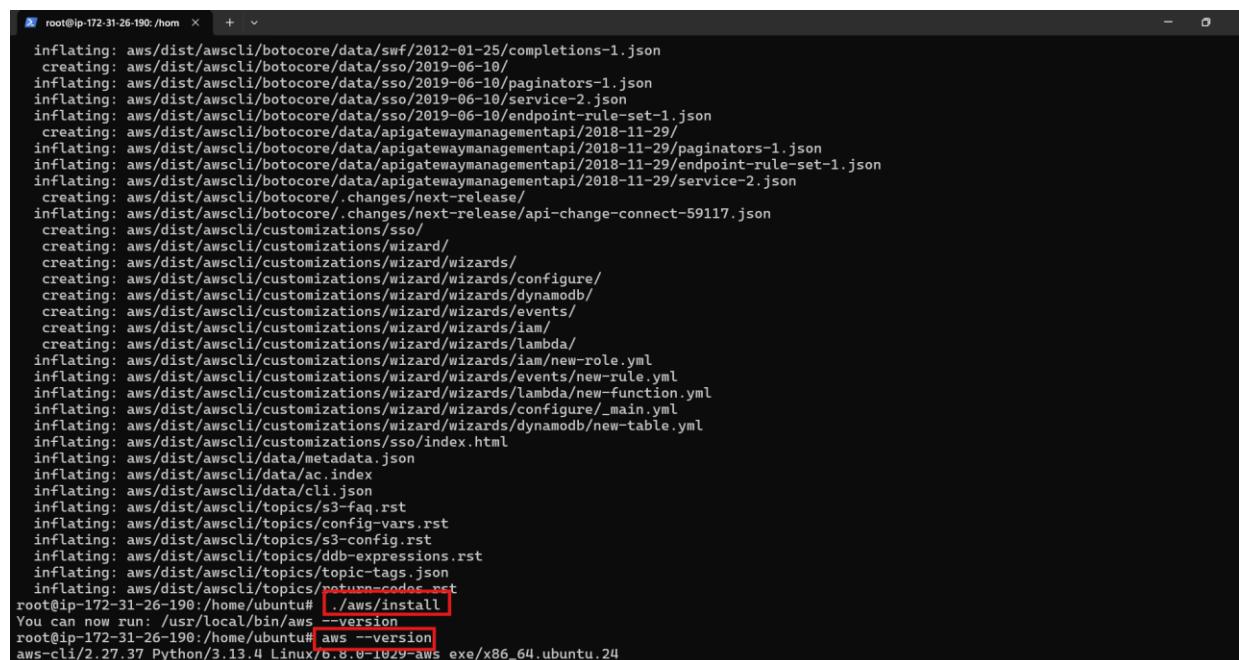
→ Unzips the AWS CLI installation package to a directory named aws/.

```
./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.



The screenshot shows a terminal window titled 'root@ip-172-31-26-190: /home' with several lines of command-line output. The output details the inflation and creation of various JSON files within the AWS CLI distribution directory, including files for SSO, pagination, and various AWS services like Lambda and IAM. At the bottom of the terminal, the command '/aws/install' is entered, followed by the message 'You can now run: /usr/local/bin/aws --version'. The terminal then displays the AWS CLI version information: 'aws-cli/2.27.37 Python/3.13.4 Linux/6.8.0-1029-aws exe/x86_64 ubuntu.24'.

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.

```

Windows PowerShell
Default output format [None]: json
root@ip-172-31-26-190:/home/ubuntu# aws ecr get-login-password --region eu-north-1 | docker login --username AWS --password-stdin 470951427029.dkr.ecr.eu-north-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded ←
root@ip-172-31-26-190:/home/ubuntu# docker tag myhttpd:latest 470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning:latest
Error response from daemon: No such image: myhttpd:latest
root@ip-172-31-26-190:/home/ubuntu# docker tag httpd:latest 470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning:latest
root@ip-172-31-26-190:/home/ubuntu# docker images
REPOSITORY                                TAG      IMAGE ID   CREATED        SIZE
aditikalamkar/dockerlearning               v1       2cf2b0e9bb8  11 hours ago  170MB
myhttpd                                    v1       2cf2b0e9bb8  11 hours ago  170MB
470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning   latest    bf07fec943ec  4 months ago  148MB
httpd                                      latest   bf07fec943ec  4 months ago  148MB
root@ip-172-31-26-190:/home/ubuntu# docker push 470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning:latest
The push refers to repository [470951427029.dkr.ecr.eu-north-1.amazonaws.com/dockerlearning]
06844457d535: Pushed
bb6bbe7eb57: Pushed
71a01f66bd83: Pushed
5f70bf18a086: Pushed
25727ad57f42: Pushed
7fb72a7d1a8e: Pushed
latest: digest: sha256:8a2ab327d73e80fe896466d1ab21b93b3b3475e05f91d5c7c89b5f2599553671 size: 1572

```

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest	Image	June 18, 2025, 21:06:07 (UTC+05:5)	59.42	Copy URI	sha256:8a2ab327d73e80...	-

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.