Docker Task1

Date:26/4/24

Q.1 Write a note on difference between Virtualization vs Containerization vs Bare Metal

Virtualization:

Virtualization involves creating virtual instances of hardware platforms, operating systems, storage devices, or network resources. It enables running multiple operating systems and applications on a single physical machine. Each virtual instance, known as a virtual machine (VM), operates independently, with its own allocated resources such as CPU, memory, storage, and networking. Hypervisors manage the virtualization process, allowing multiple VMs to run concurrently on the same hardware. Virtualization offers benefits like hardware resource optimization, isolation between VMs, and the ability to run legacy applications.

Containerization:

Containerization is a lightweight form of virtualization that encapsulates an application and its dependencies into a single package called a container. Containers share the host operating system's kernel and resources, making them more lightweight and efficient compared to VMs. They provide consistency across different environments, enabling applications to run reliably from development to production. Containerization platforms like Docker and Kubernetes have gained popularity due to their ability to streamline the development, deployment, and scaling of applications. Containers offer benefits such as portability, scalability, and faster deployment times compared to traditional virtualization.

Bare Metal:

Bare metal refers to running applications directly on the physical hardware without any intervening layer of abstraction like virtualization or containerization. In a bare metal environment, the operating system interacts directly with the underlying hardware, maximizing performance and resource utilization. Bare metal setups are often used in high-performance computing (HPC), real-time applications, and scenarios where minimal overhead and maximum control are required. While bare metal environments offer the highest level of performance and control, they may require more management effort compared to virtualized or containerized environments.

Parameters	Virtualization	Containerization	Bare Metal
Resource Allocation	Virtualization divides physical resources into virtual ones, allowing multiple isolated instances to share the same hardware.	Containerization shares the host operating system's resources among containers, providing lightweight isolation.	Bare metal environments utilize the entire physical hardware for each application.
Performance Overhead	Virtualization introduces **some overhead** due to the hypervisor layer	Containerization has **less overhead** since containers share the host OS kernel.	Bare metal environments offer the best performance with **minimal overhead**.
Isolation	Virtualization provides **strong isolation** between VMs since each VM operates independently.	Containerization offers **lighter isolation**, sharing the host OS kernel but providing separate user spaces.	Bare metal environments offer **no isolation** between applications running directly on the hardware
Portability	It is portable but require additional management overhead.	Containers are highly portable, allowing applications to run consistently across different environments.	Bare metal setups are less portable since applications are tightly coupled with the underlying hardware.
Management Complexity	It require less manual management compared to Bare metal or containerized envirinments.	It require medium manual management compared to virtulized envirinments.	Bare metal environments require more manual management compared to virtualized or containerized environments.

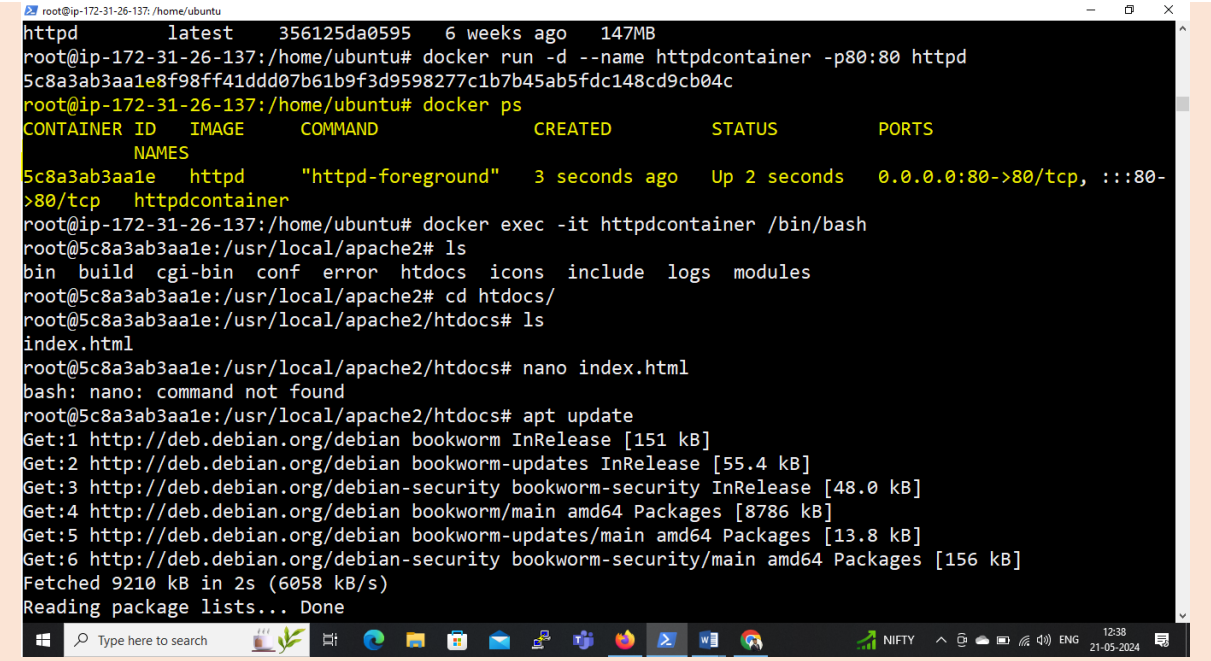
Q.2 Create an httpd container, inside that container create 2 more html pages (eg. Home.html and about.html) which will be accessible from browser.

 # To Create an httpd container use the following command:

- docker run –d –name<Containername> -p80:80 <imagename>
- eg. docker run -d --name httpdcontainer -p80:80 httpd

 # To see container :

- docker ps



 # To go inside the container use following command:

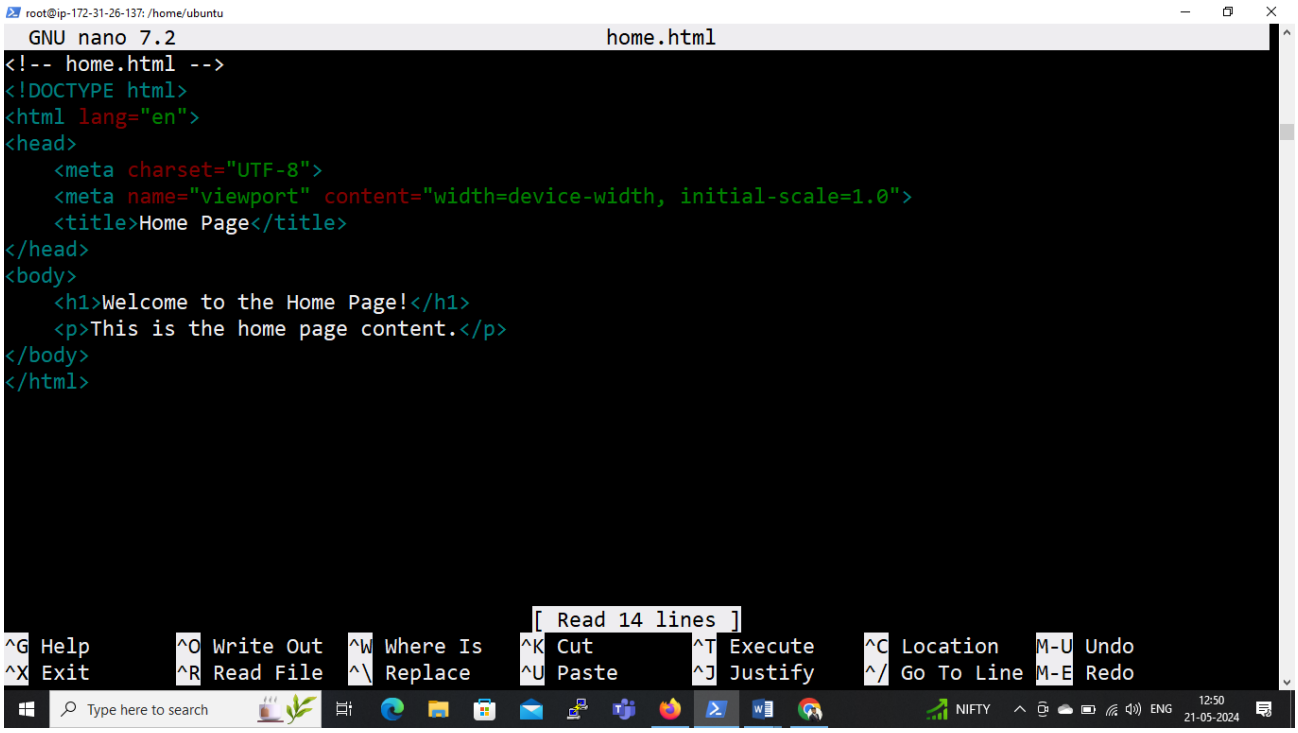
- docker exec -it <container name> /bin/bash
- eg. docker exec -it httpdcontainer /bin/bash

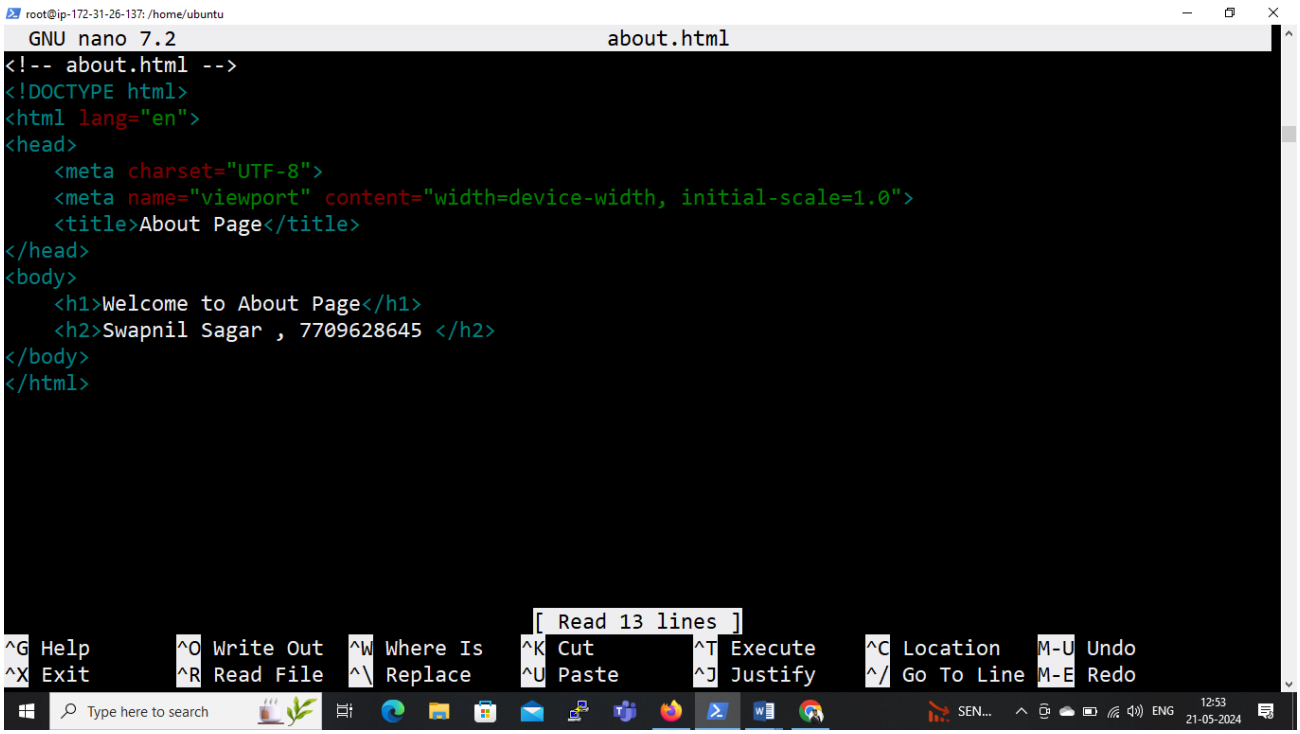
 # To create html pages:

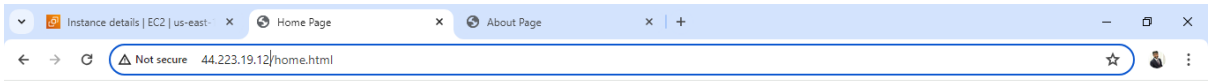
- Go to , cd /usr/local/apache2/htdocs
- apt update
- apt install nano -y
- create page , nano home.html (see below screenshot)
- create page , nano about.html (see below screenshot)

 # To see page on bowser:

- http://PUBLIC IP of server/page name
- http://44.223.19.12/home.html
- http://44.223.19.12/about.html

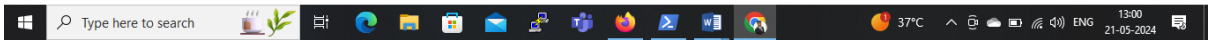

root@ip-172-31-26-137: /home/ubuntu
GNU nano 7.2 home.html
<!-- home.html -->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Home Page</title>
</head>
<body>
<h1>Welcome to the Home Page!</h1>
This is the home page content.
</body>
</html>
[Read 14 lines]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo
Type here to search
NIFTY
ENG
12:50
21-05-2024

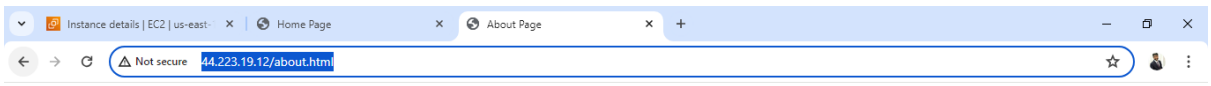

root@ip-172-31-26-137: /home/ubuntu
GNU nano 7.2 about.html
<!-- about.html -->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>About Page</title>
</head>
<body>
<h1>Welcome to About Page</h1>
<h2>Swapnil Sagar , 7709628645 </h2>
</body>
</html>
[Read 13 lines]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo
Type here to search
SEN...
ENG
12:53
21-05-2024



Welcome to the Home Page!

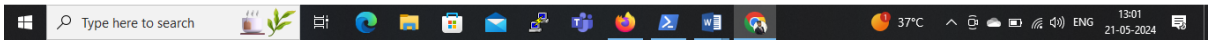
This is the home page content.





Welcome to About Page

Swapnil Sagar , 7709628645

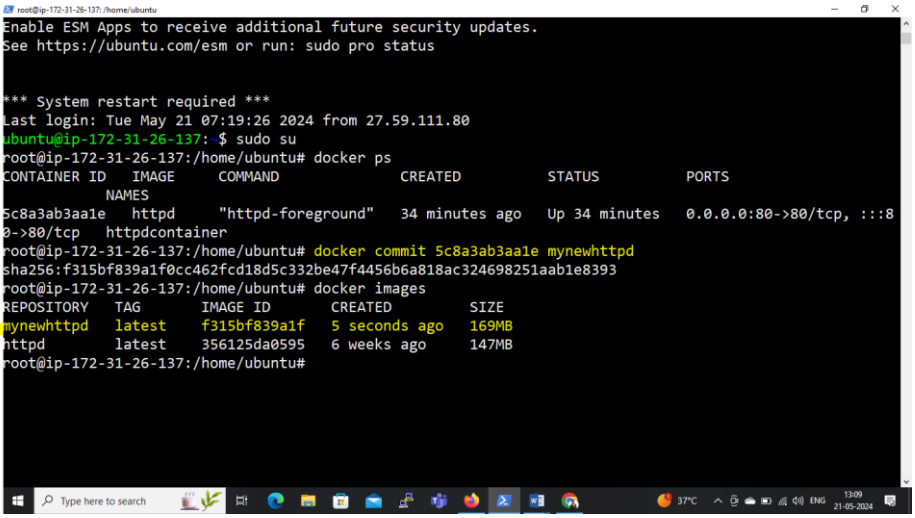


Q.3 Create image from httpd container and push that image to

a. **Docker hub**
b. **ECR (AWS students)**

 # To Create image from container use the command:

- docker commit <containerid/name> image name
- Eg. docker commit 5c8a3ab3aa1e mynewhttpd



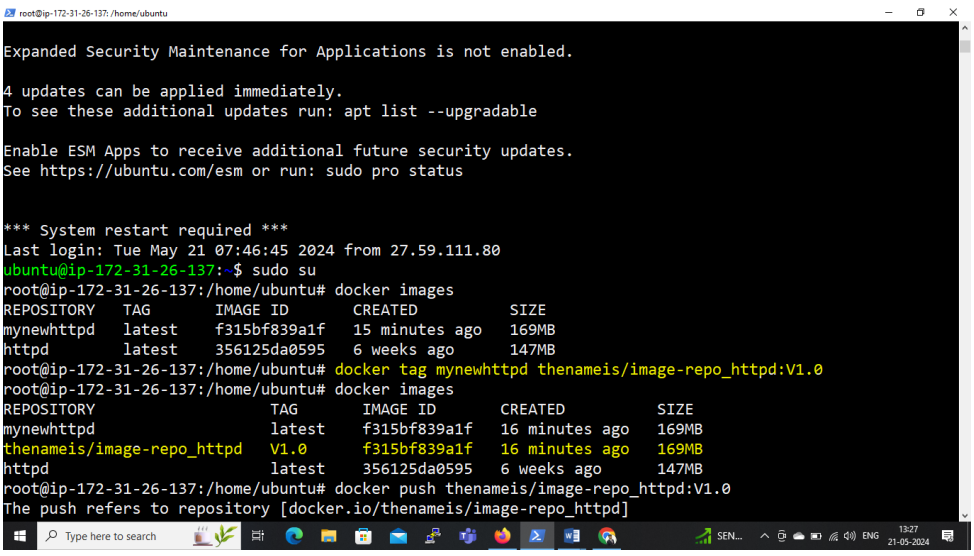
A. <u>To push this image to Docker Hub:</u>

 # Create a repository on docker-hub:

- Eg. image-repo_httpd

 # To give a tag to the image who want to push:

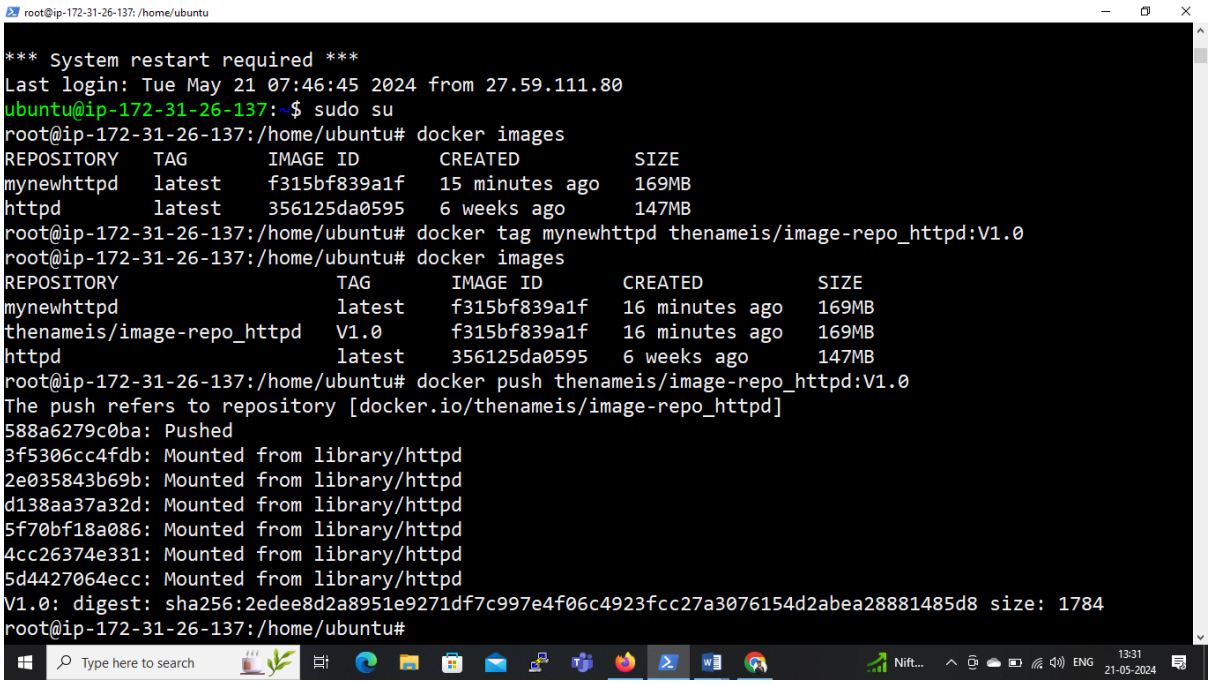
- docker tag <imagename> <dockerhubusername/repositoryname>:<tagname>
- Eg. docker tag mynewhttpd thenameis/image-repo_httpd:V1.0

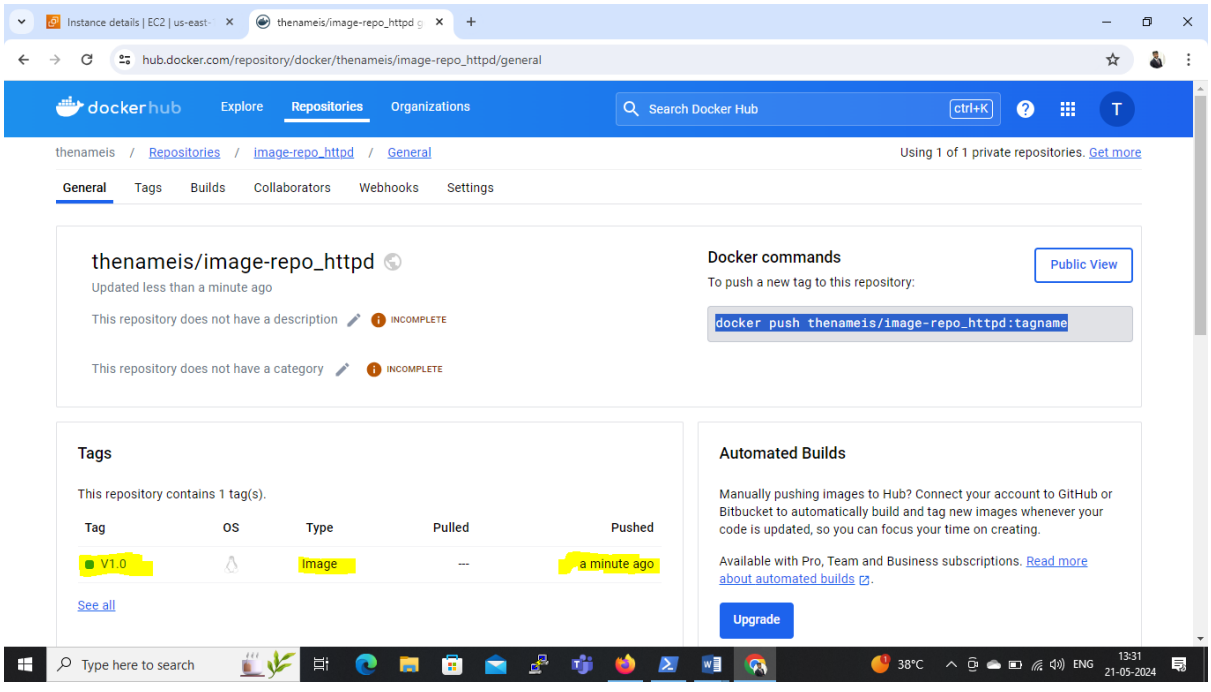


 # To Push the image to docker hub use the following command:

`docker push thenameis/image-repo_httpd:tagname`

- docker push thenameis/image-repo_httpd:V1.0





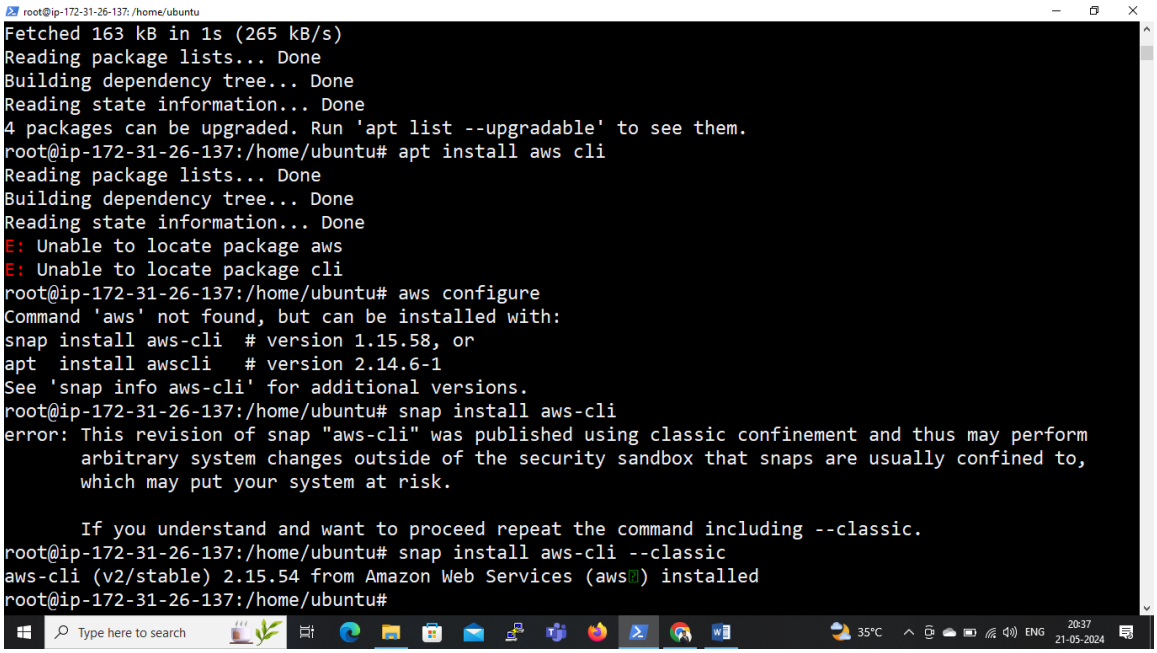
B. To push this image to ECR:

 # Create a Public Repository in AWS

- Repository Name: httpd-image-repo

 # Install AWS CLI with using following command:

- snap install aws-cli –classic

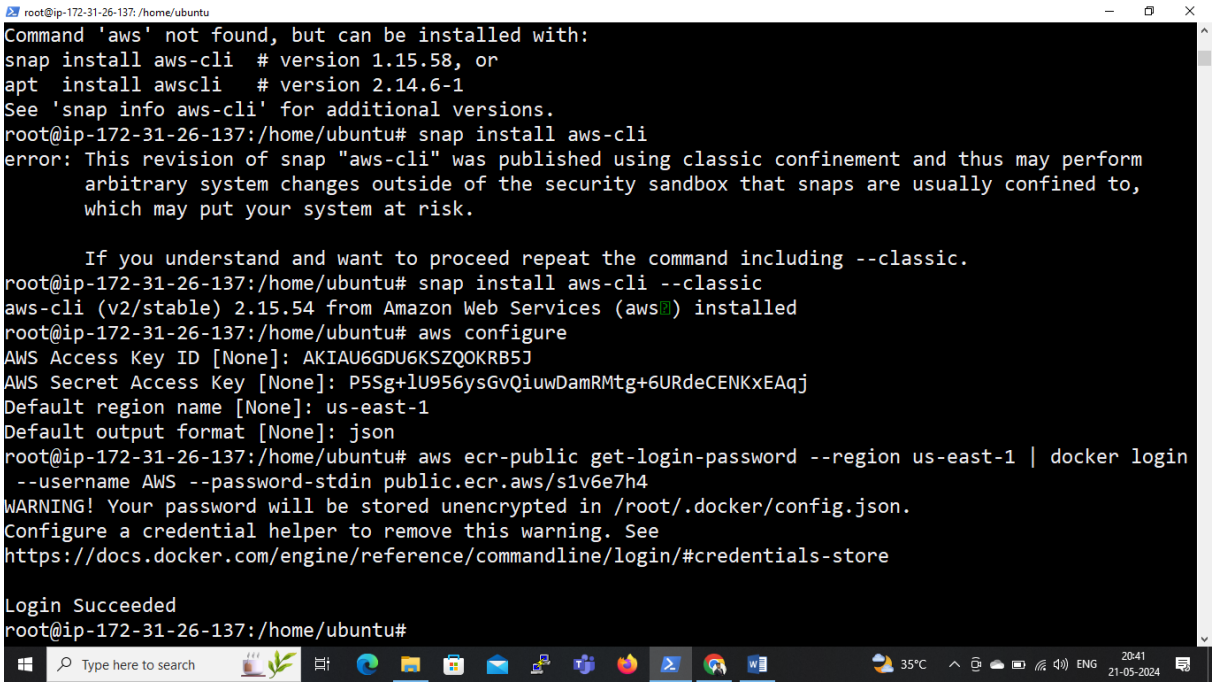


 # To configure AWS: aws configure

- Access key ID:
- Secret access key:
- Default region Name: us-east-1
- Default output format: json

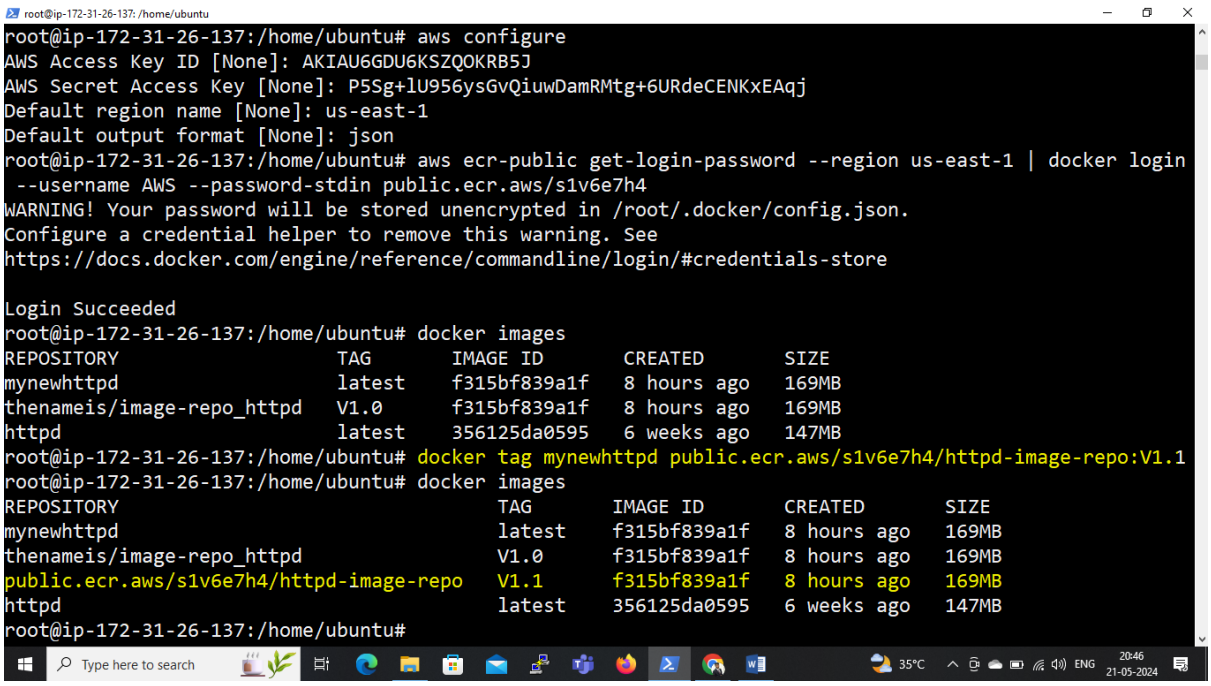
 # From view push commands on repository:

- aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/s1v6e7h4



 # To give a tag to the image who want to push:

- docker tag mynewhttpd public.ecr.aws/s1v6e7h4/httpd-image-repo:V1.1



 # To Push the image to ECR use the following command:

- docker push public.ecr.aws/s1v6e7h4/httpd-image-repo:V1.1