1. Create an ec2 instance with http, ssh and allow security groups
2. Connect to the ec2 instance using “**ssh -i xyz.pem ec2-user@public-ip”**
3. Install docker and add user to docker group using the following commands

* sudo yum update -y
* sudo yum install docker -y
* sudo service docker start
* sudo systemctl enable docker
* sudo usermod -aG docker ec2-user 🡪logout of the instance and login back again to add the user to docker group inorder to get access to the docker commands

1. Install nginx into ec2 instance

* docker pull nginx: version 🡪 pulls the particular version mentioned. If no version specified then it will pull the latest version of nginx from docker hub
* docker images 🡪 to check the image is created or not

image will be created and now we need to create a container for the above image

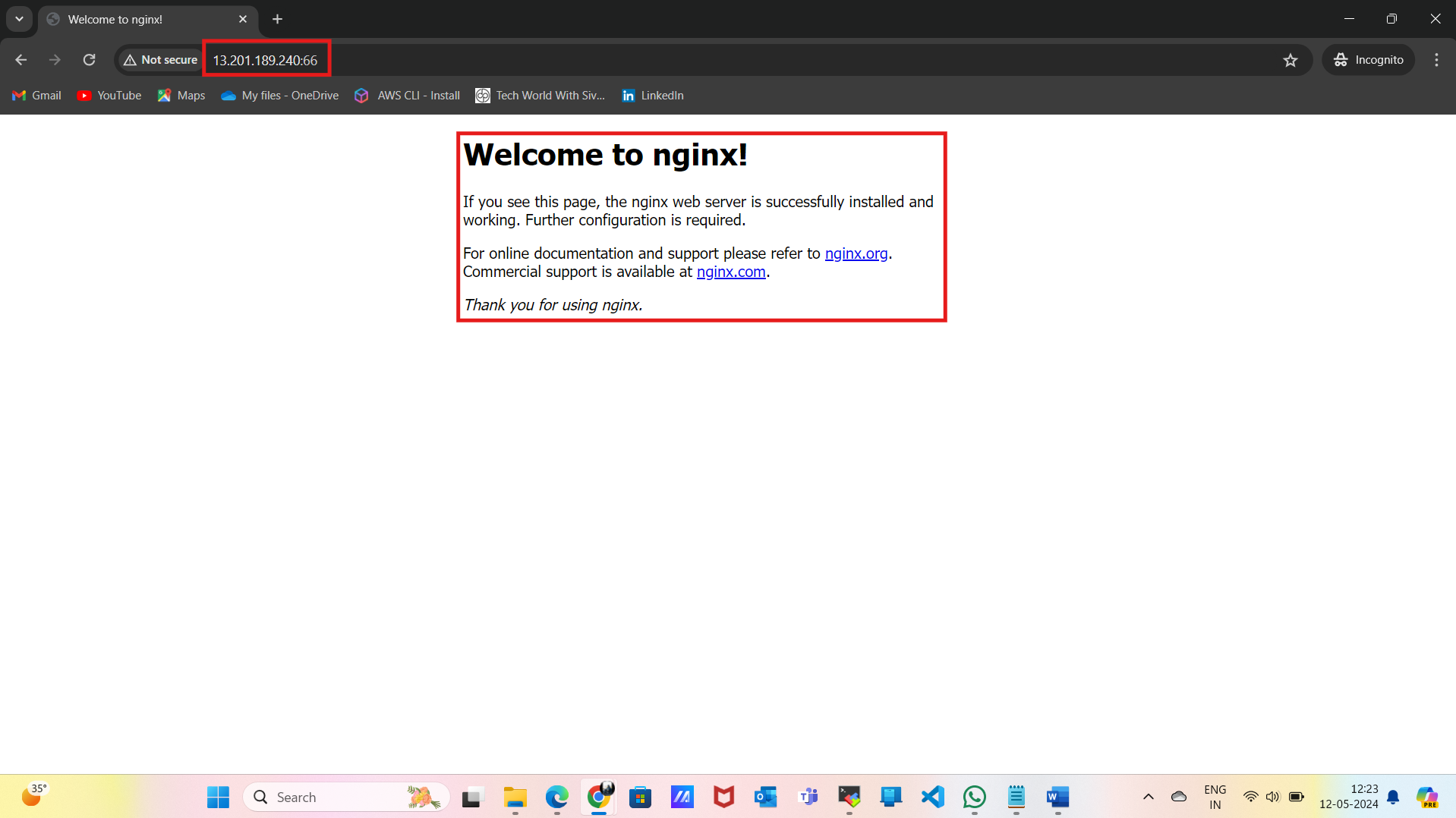
* docker create <**image id**> 🡪 create a container from the image
* docker ps -a 🡪 to check the container is created or not

container will be created

1. docker commands

* docker ps 🡪 to get the list of running containers in the server
* docker ps -a 🡪 to get the list of all containers in the server
* docker images 🡪 to get the list of images in the server
* docker - -version 🡪 to get the version of docker
* docker start <**container id**> 🡪 to start the container
* docker inspect <**container id**> 🡪 to get the complete info about the container
* **docker run image: version 🡪 docker pull + docker create + docker start**
* docker stop <**container id**> 🡪 to stop the running container
* docker rm <**container id**> 🡪 to remove the container from server
* docker rm -f $(docker ps -aq) 🡪 to remove all the containers at once
* docker rmi <**image id**> 🡪 to remove the docker image from the server
* docker rmi -f $(docker images -aq) 🡪 to remove all the images at once (f stands for removing the image or container forcefully)
* docker run -d image: version 🡪 by default the container will run in the foreground, by using -d (detached mode) the container runs in background
* docker run -d -p 66:80 nginx: latest 🡪 exposing the port to outside world. here container port [80] is exposed to host port [66]. By using host port [66], we can access through outside world (**INTERNET**) **{public-Ip: host port}**

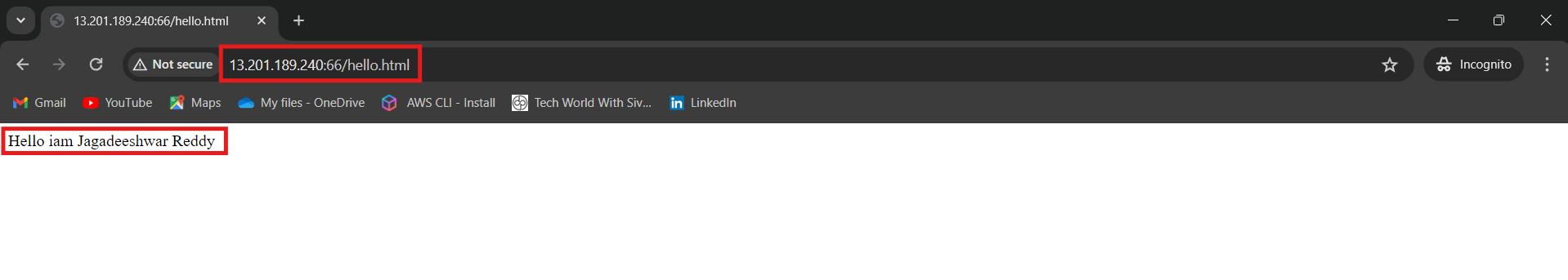
1. container is the running version of image
2. Image is a static single file; it can run any number of containers.



* docker exec -it < **container id** >bash 🡪 to enter inside the container. -it is interactive terminal
* CTRL + D 🡪 to come out of the container
* docker exec cat /etc/\*release 🡪 will give us the underlying OS.
* cd **usr/share/nginx/html/** 🡪 The default html location for nginx

EX: - echo “Hello i am Jagadeeshwar Reddy" > hello.html

we can access through (**INTERNET**) using **{public-Ip: host port/hello.html}**



|  |  |  |  |
| --- | --- | --- | --- |
|  | **PHYSICAL MACHINE** | **VIRTUAL MACHINE** | **CONTAINERS** |
| SCALABILITY | BAD | AVERAGE | GOOD |
| RESOURCE OPTIMISATION | BAD | AVERAGE | GOOD |
| HIGH AVAILABILITY | BAD | AVERAGE | GOOD |
| COST | BAD | AVERAGE | GOOD |
| PROVISIONING | BAD | AVERAGE | GOOD |
| PERFORMANCE | GOOD | AVERAGE | GOOD |
| SECURITY | GOOD | AVERAGE | BAD |
| PORTABILITY | BAD | AVERAGE | GOOD |
| CONSISTENT ENV | BAD | AVERAGE | GOOD |

* dockerfile: - Declarative approach to create our own images

DOCKERFILE

**FROM: -** FROM is the first instruction in dockerfile. It refers to the base OS, to create an image we always refer to the base OS

FROM <base OS>:<version>

* docker build -t <image-name>:<version> . ( . refers to the current directory, intimating the docker engine that the docker file is in current directory) [from local]
* docker build -t <Docker-hub-url>/username/image-name:version . (from docker hub)