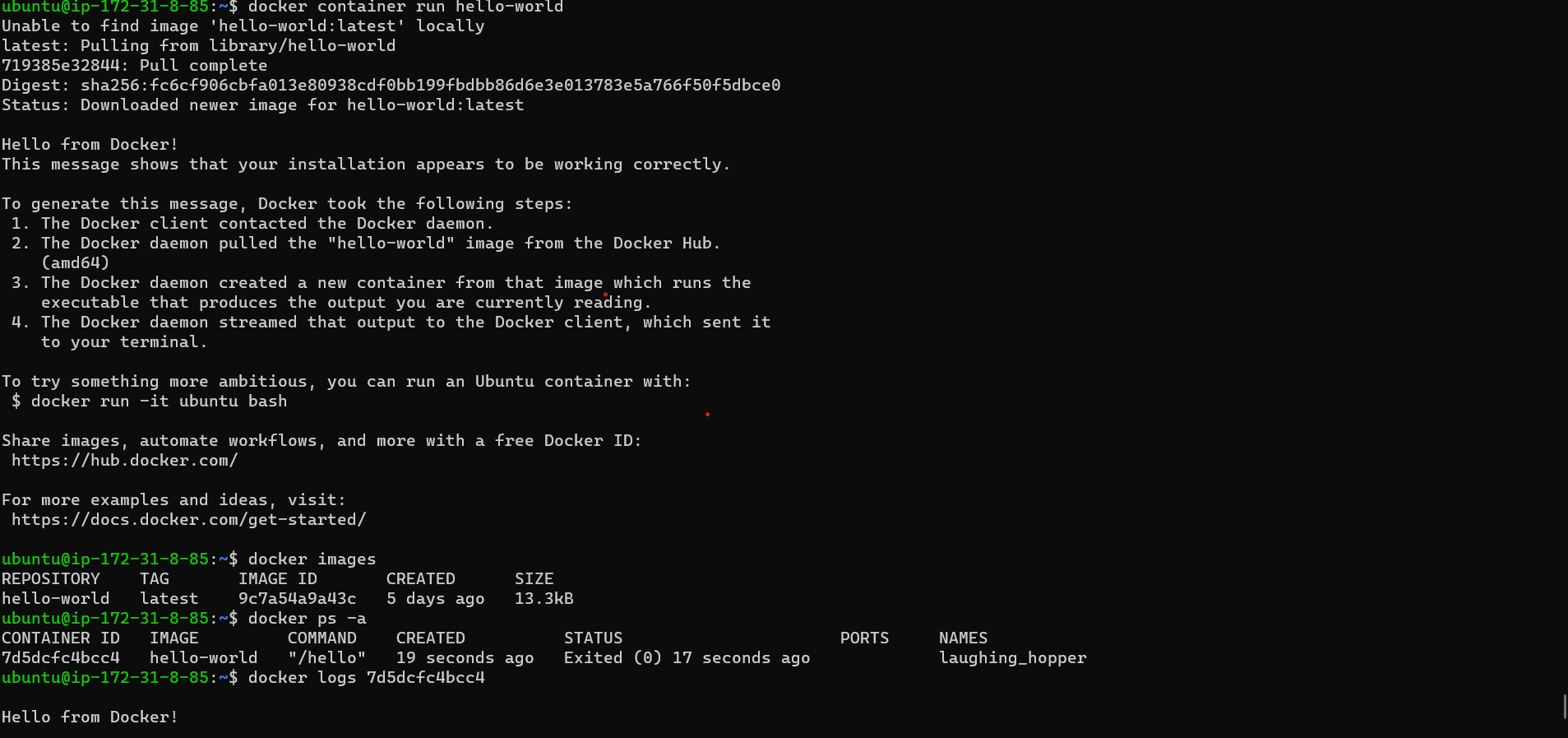
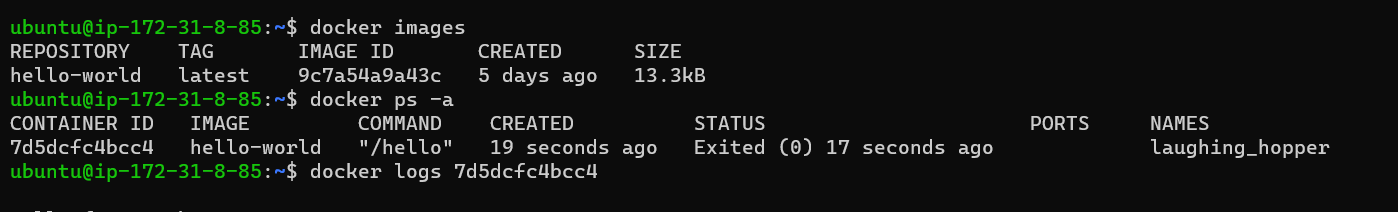
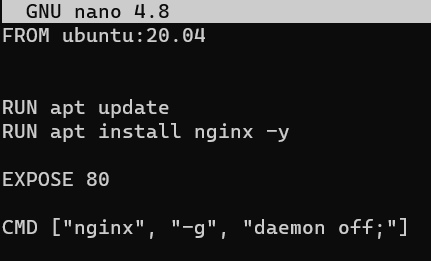
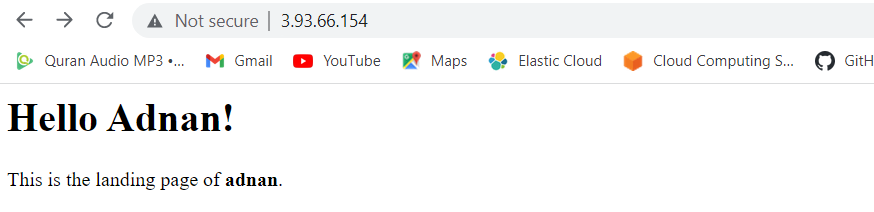
**Run hello-world docker container and observe the container status**

****

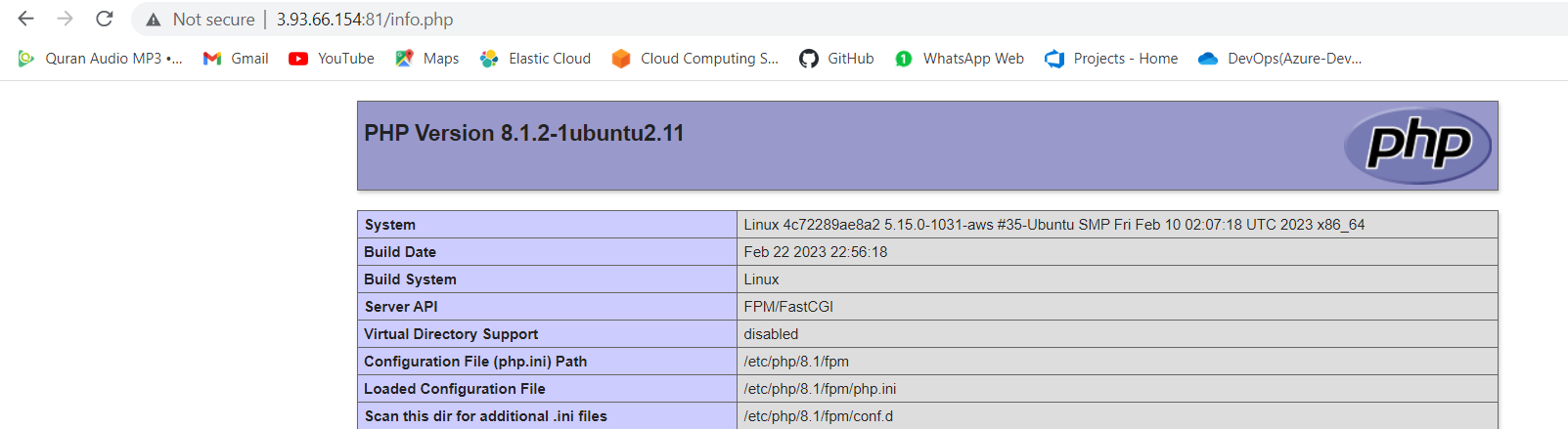
**Check the docker images and write down the size of hello-world image**

****

**Run the nginx container with name as nginx1 and expose it on 8080 port on docker **

****

****

****

**What is the lifecycle of a docker container?**

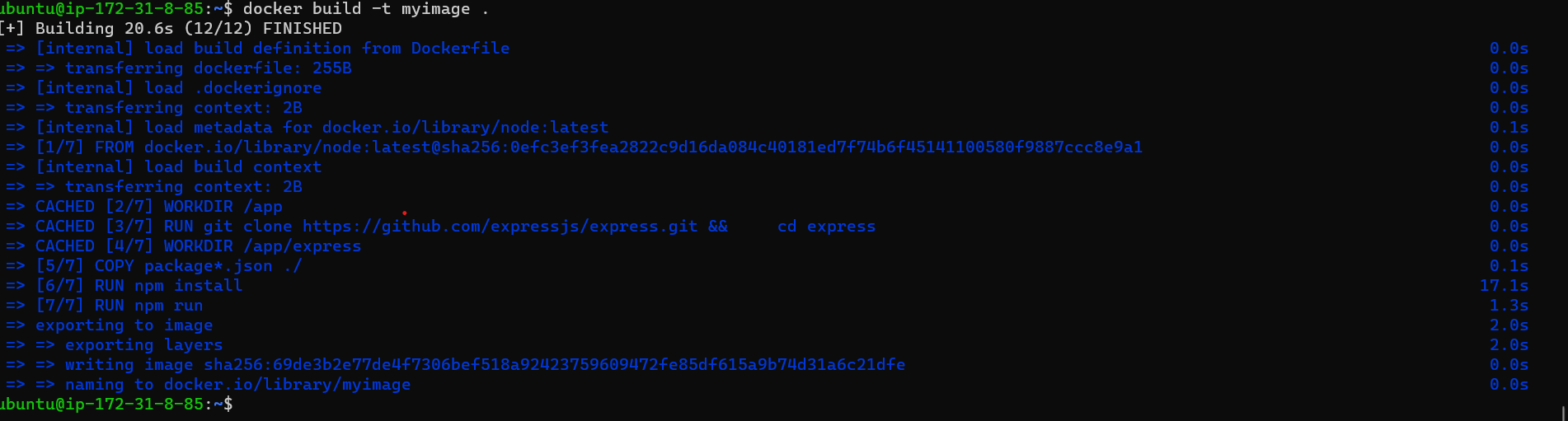
* The lifecycle of a docker container consists of five states: In the created state, a docker container is created from a docker image. In the running state, the docker container starts executing the commands mentioned in the image. To run a docker container, use the docker run command. The docker run command creates a container if it is not present.
* **Explain what happens when you run the docker container**
* When you run a container, Docker creates a set of namespaces for that container. These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace.

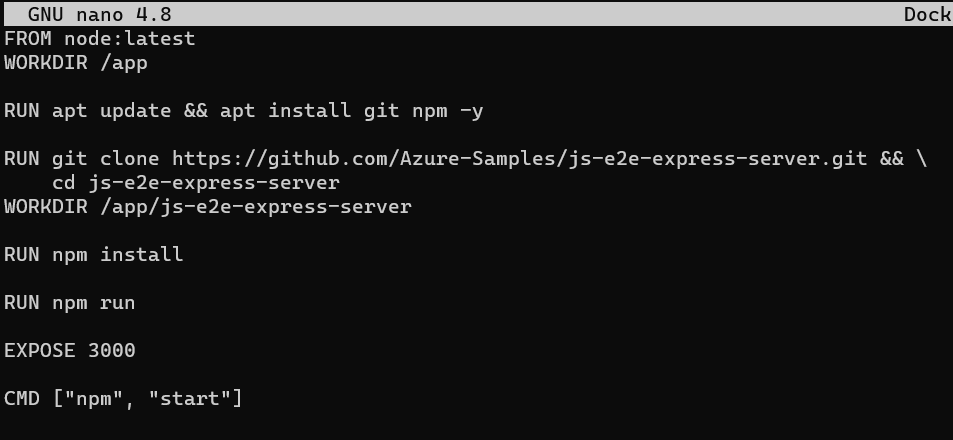
1. **Explain the Docker architecture**

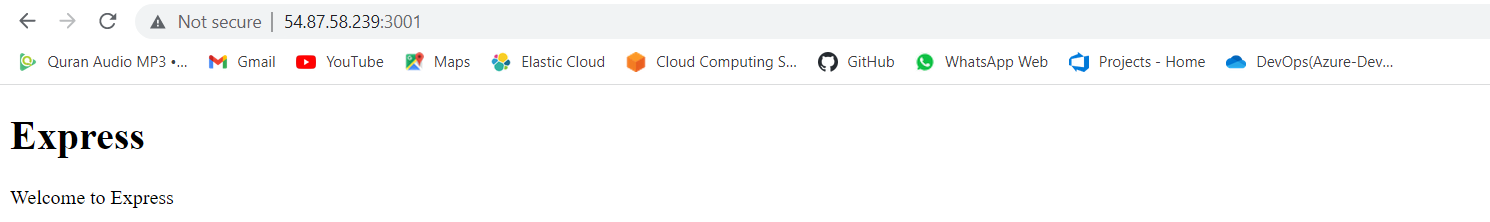
 Let me explain you the components of a docker architecture. It is the core part of the whole Docker system. Docker Engine is an application which follows client-server architecture. It is installed on the host machine. There are three components in the Docker Engine:

**DAY-2**

**write a Docker file for NodeJS application – expressjs**

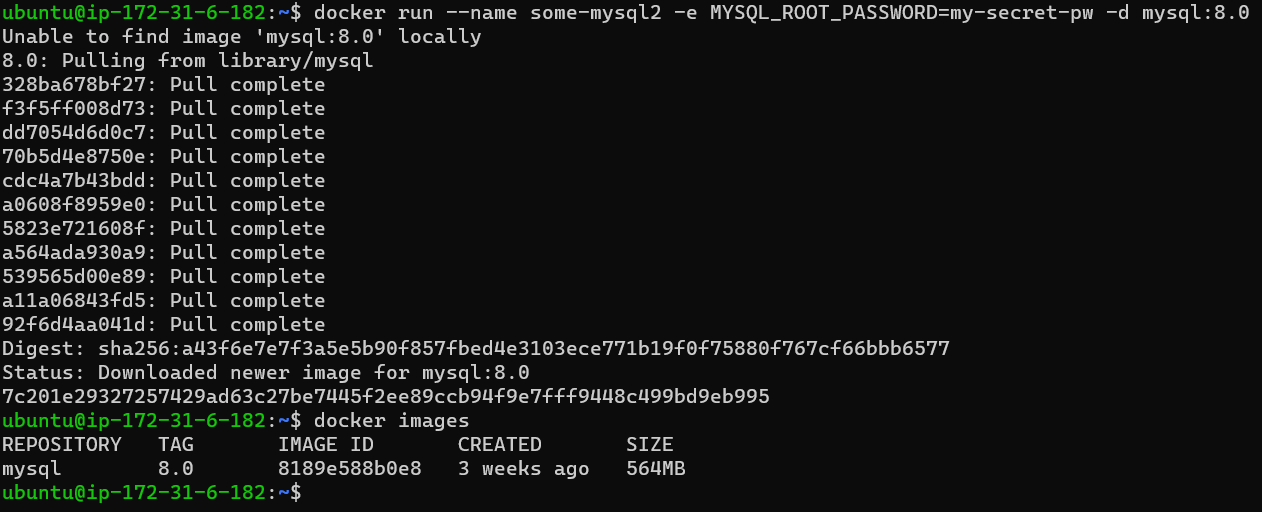
****

****

****

**DAY 3**

**create a MySQL dB container from official MySQL image**

****

**First we have to create a db container with CLI**

**{docker run --name some-mysql2 -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:8.0}**

**To check our image is build or not**

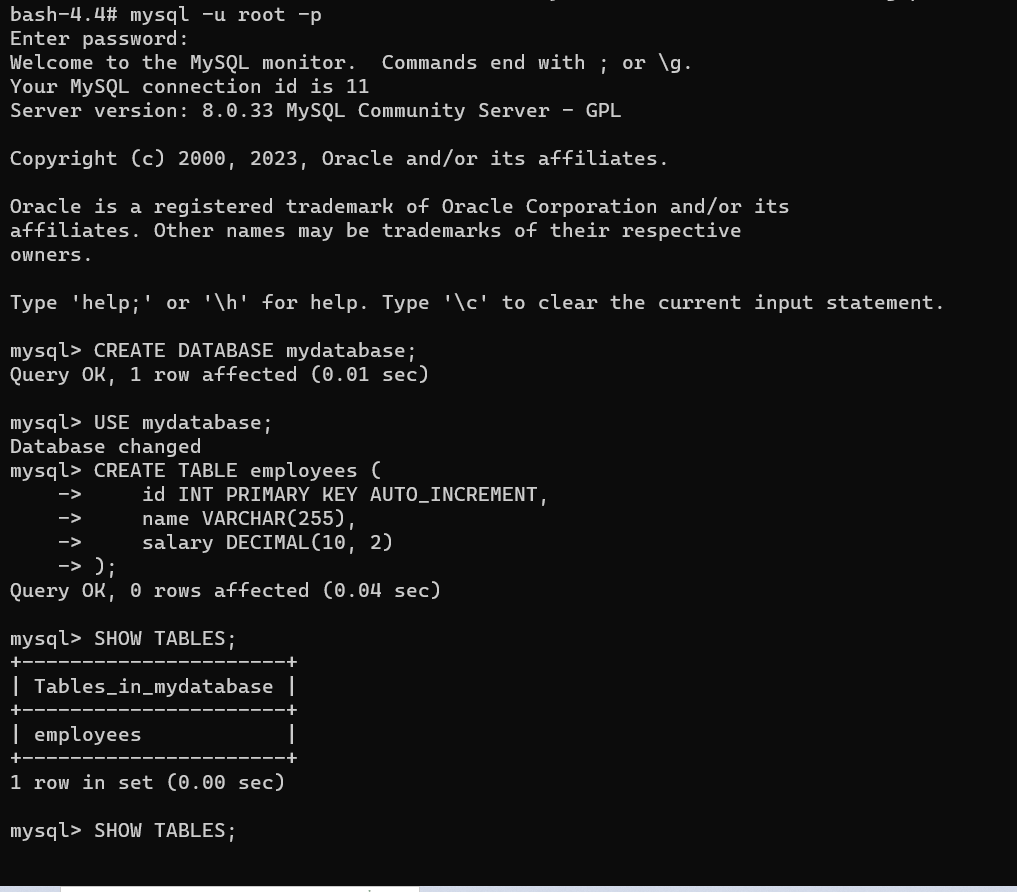
**{docker images}**

**Then we have to enter in this container**

**{** **docker exec -it some-mysql2 /bin/bash}**

**We can enter in container after giving our password for mysql**

**{mysql -u root -p}**

****

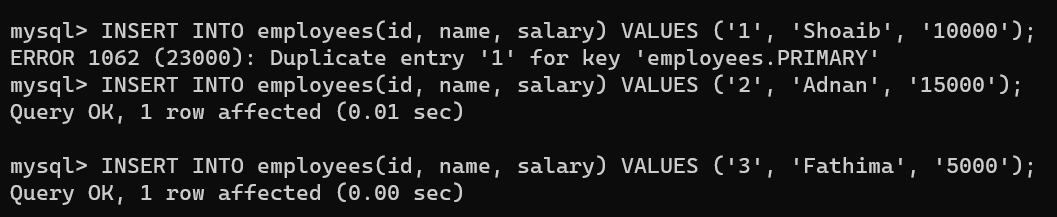
**Then we have to create a database with**

**CREATE DATABASE dtabasename**

**We have to use our database by this command**

**USE databasename**

**To see tables we have to type SHOW TABLES;**

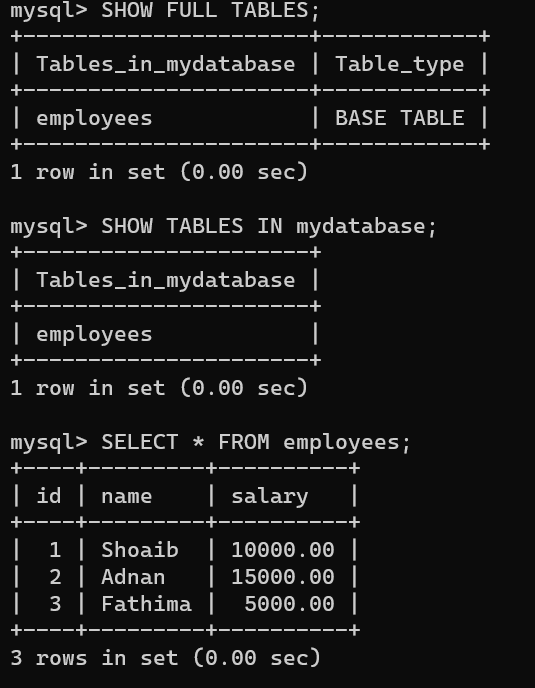
****

**To add content in table:**

**INSERT INTO employees(id, name, salary) VALUES ('2', 'Adnan', '15000');**

**To see full content of table**

**SELECT \* FROM employees;**

****

**why to mount one container to another ?**

Mounting one container to another can be useful in certain scenarios where you need to share or persist data between containers. Here are a few reasons why you might want to mount one container to another

1 Data Sharing

2 Persistent Storage

3 Configuration Sharing

4 Development Testing

**To mount one container to other with mysql container**

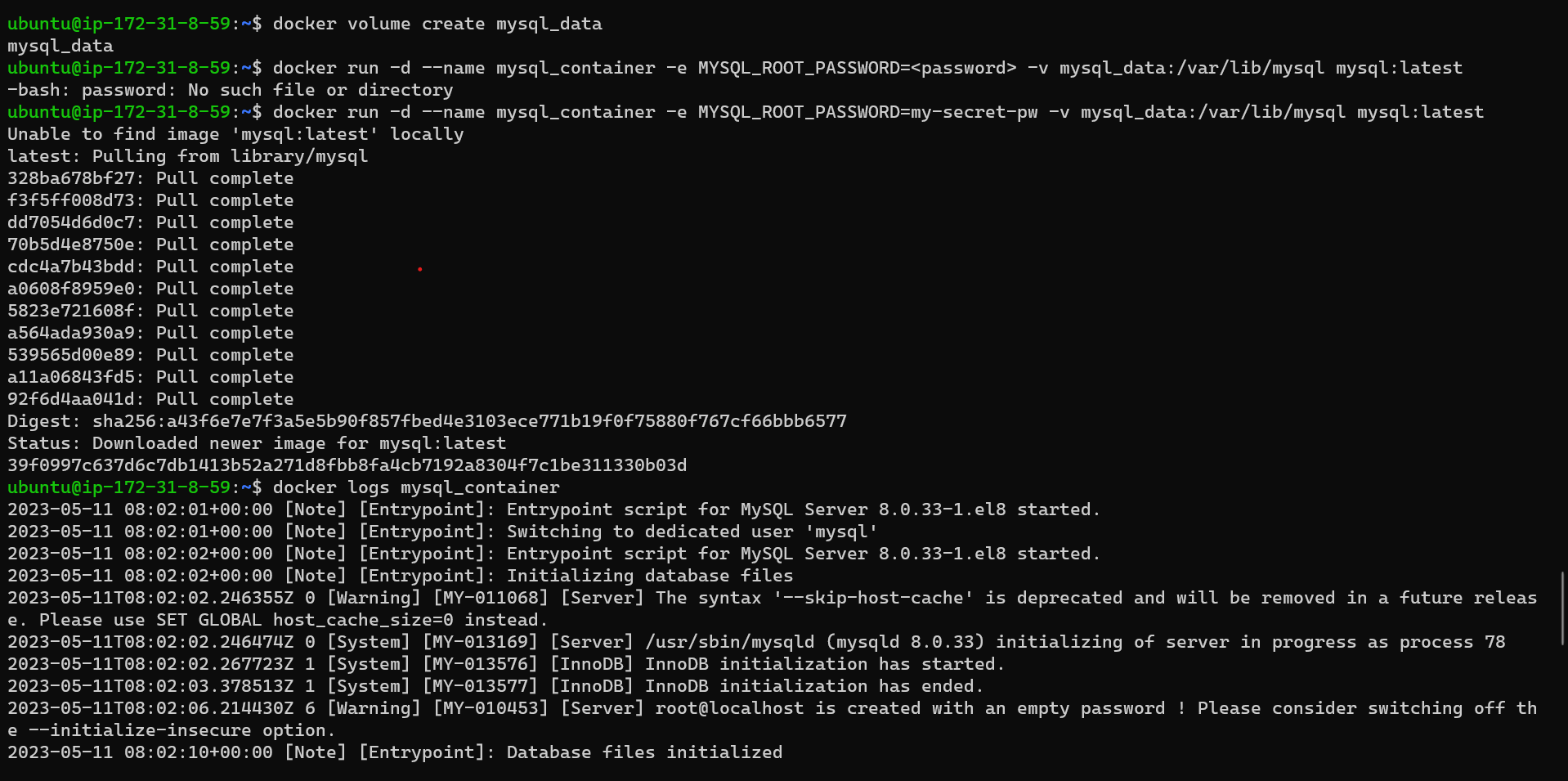
**Use this commands**

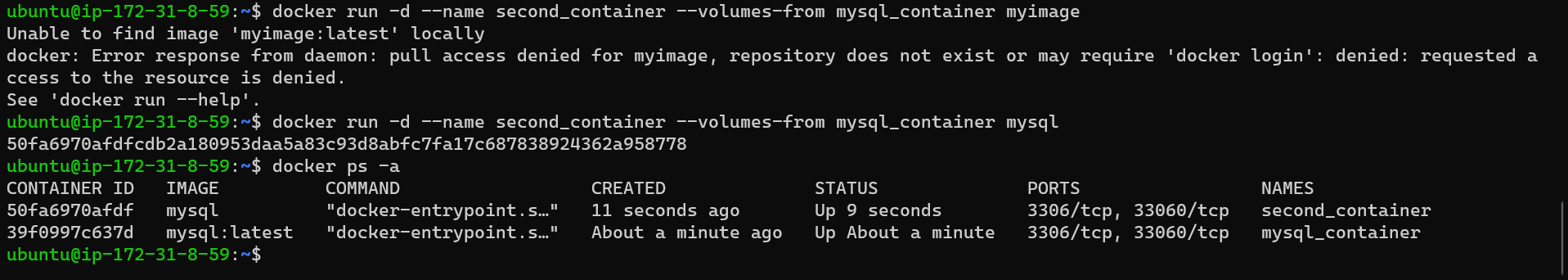
docker volume create mysql\_data

docker run -d --name mysql\_container -e MYSQL\_ROOT\_PASSWORD=<password> -v mysql\_data:/var/lib/mysql mysql:latest

docker logs mysql\_container

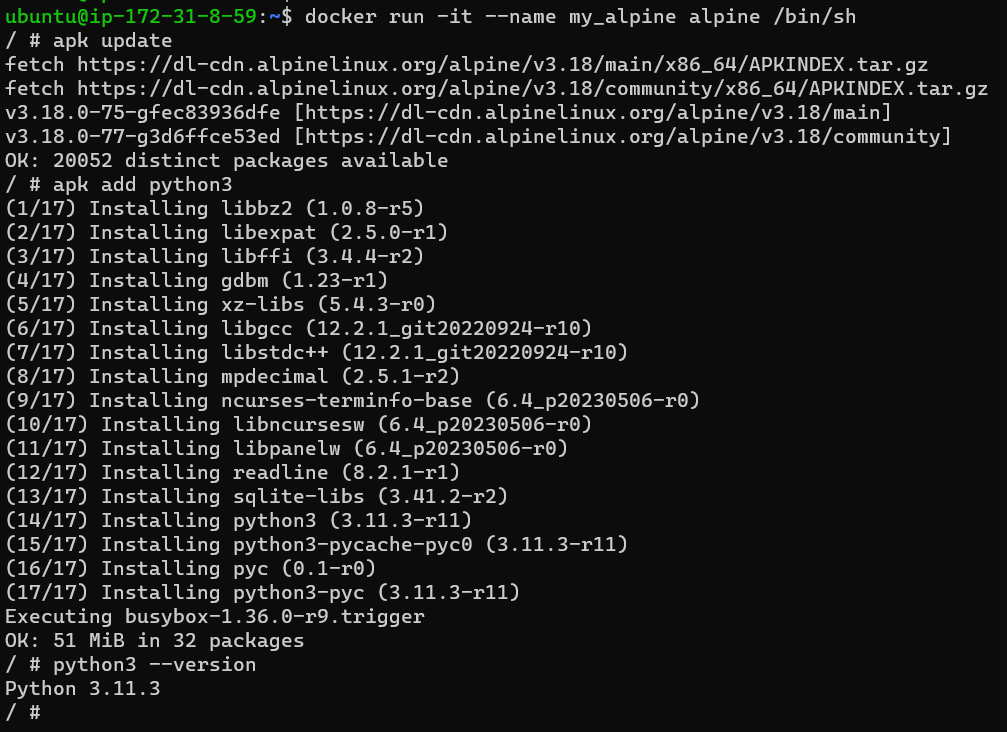
docker run -d --name second\_container --volumes-from mysql\_container <other\_container\_image>

****

****

**DAY - 4**

1. **Create an alpine container in interactive mode and install python**

****

**In this first we have to create alpine container with this command**

**{docker run -it –name my\_alpine alpine /bin/bash}**

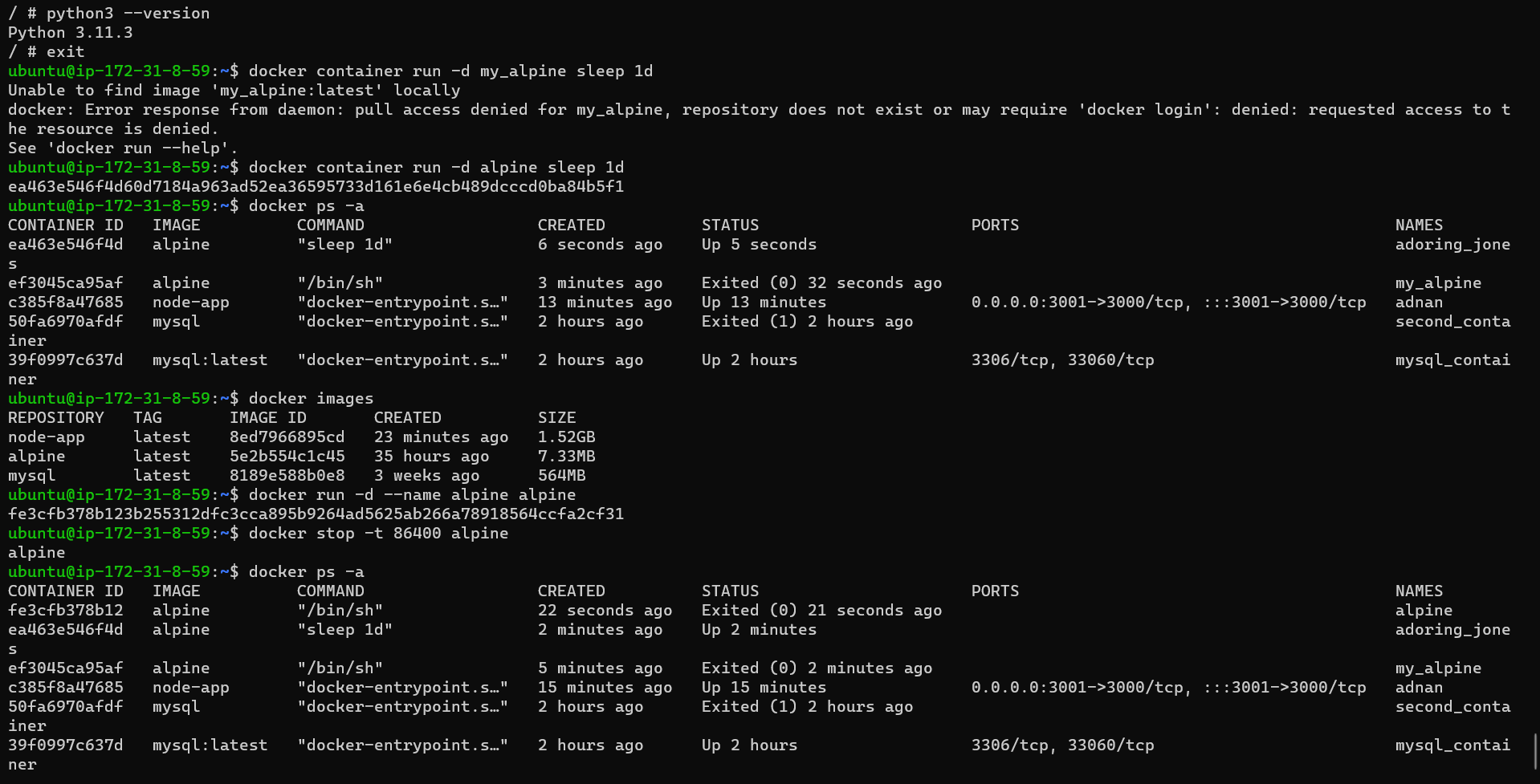
**After entering in container we have to install python**

**Apk update**

**Apk add python**

**Python3 --version**

1. **Create a ubuntu container with sleep 1d and then login using exec and install python**

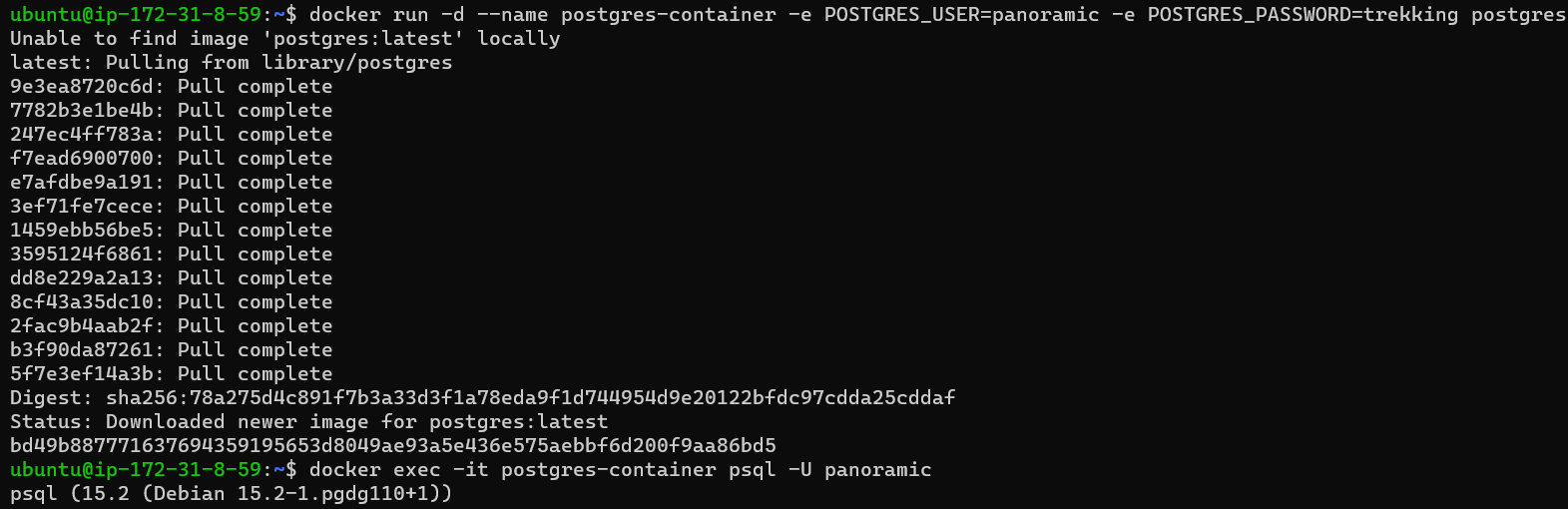
****

**In this we have to stop our running container**

**{docker stop -t 86400 alpine}**

**Here I stopped mycontainer for 86400 seconds**

**Create a postgres container with username panoramic and password as trekking. Try logging in and show the databases (query for psql)**

****

**Try to create a docker file which runs php info page, use ARG and ENV wherever appropriate on 1. Apache, 2. Nginx**

**TO COMPLETE THIS TASK**

First we have to install docker in our machine

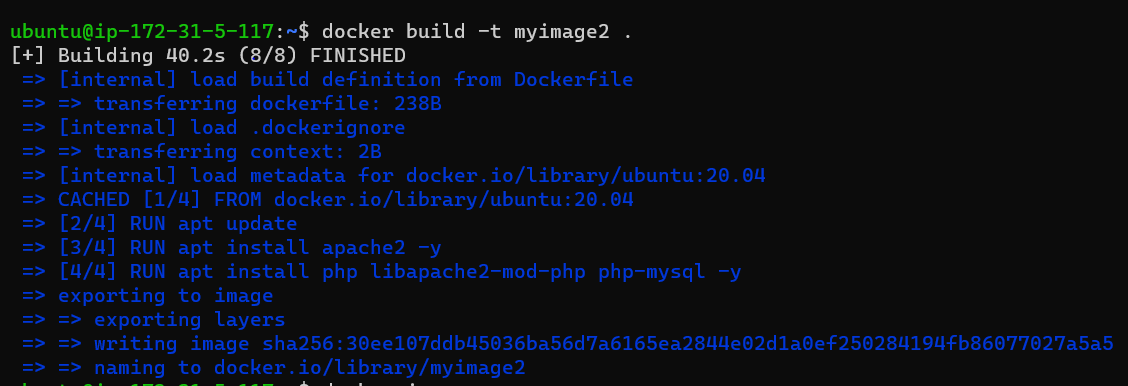
sudo apt update

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

sudo usermod -aG docker ubuntu

After this we need to add a Dockerfile to build our apache2 image

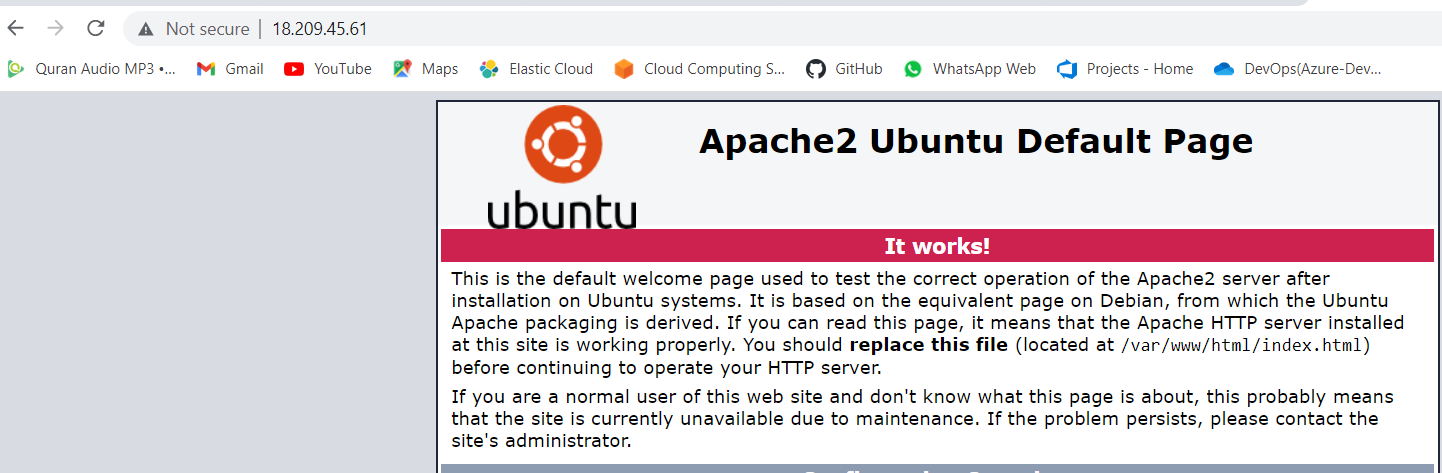


After completion of image building check whether image is build

(**docker images)**

Then create a container on 80:80 port

Docker run -d –name containername -p 80:80 (imagename)



**Nginx**

**TO COMPLETE THIS TASK**

First we have to install docker in our machine

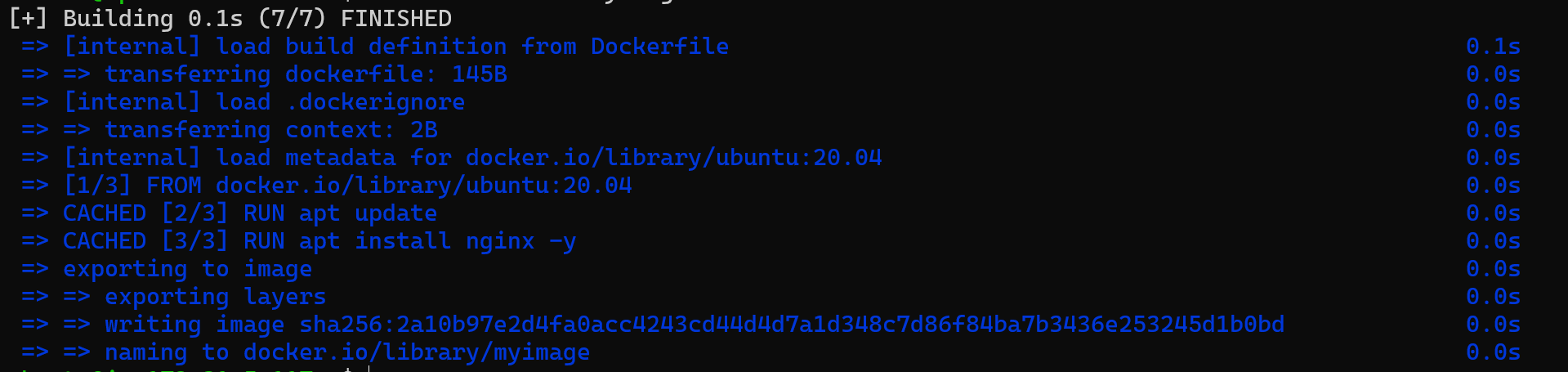
sudo apt update

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

sudo usermod -aG docker ubuntu

After this we need to add a Dockerfile and nginx.conf file to build our nginx image



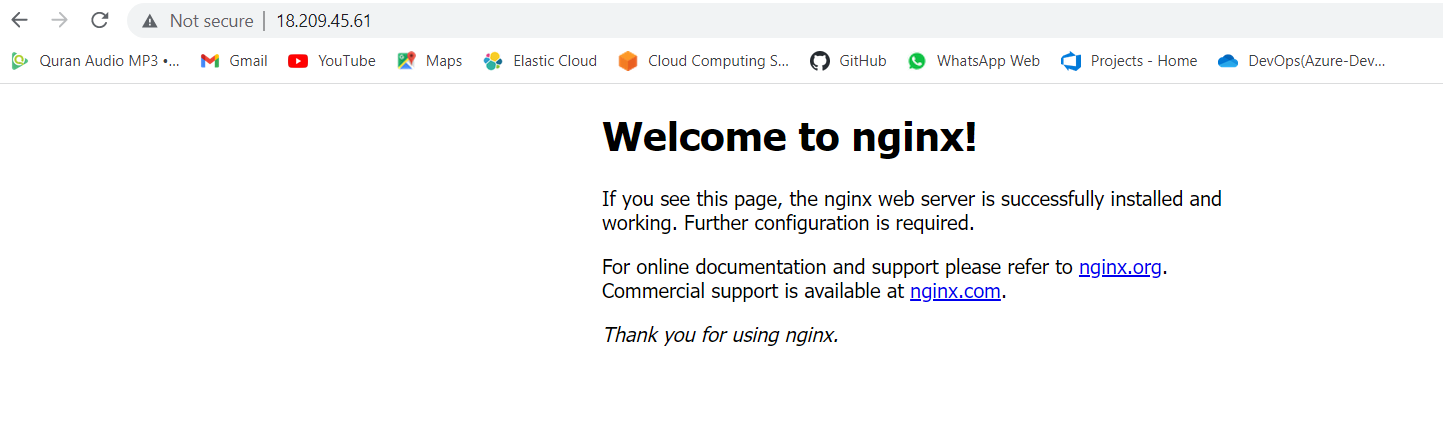
After completion of image building check whether image is build

**(docker images)**

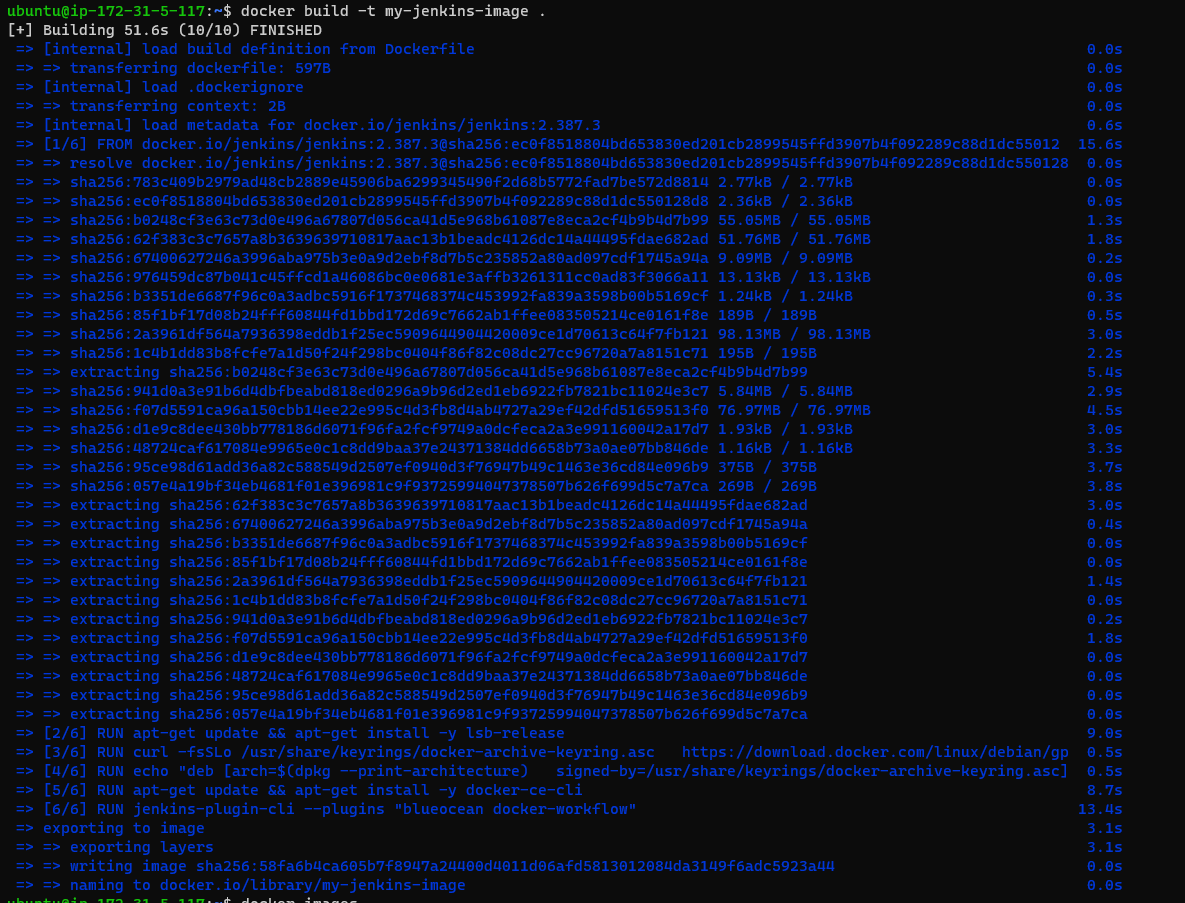
Then create a container on 80:80 port

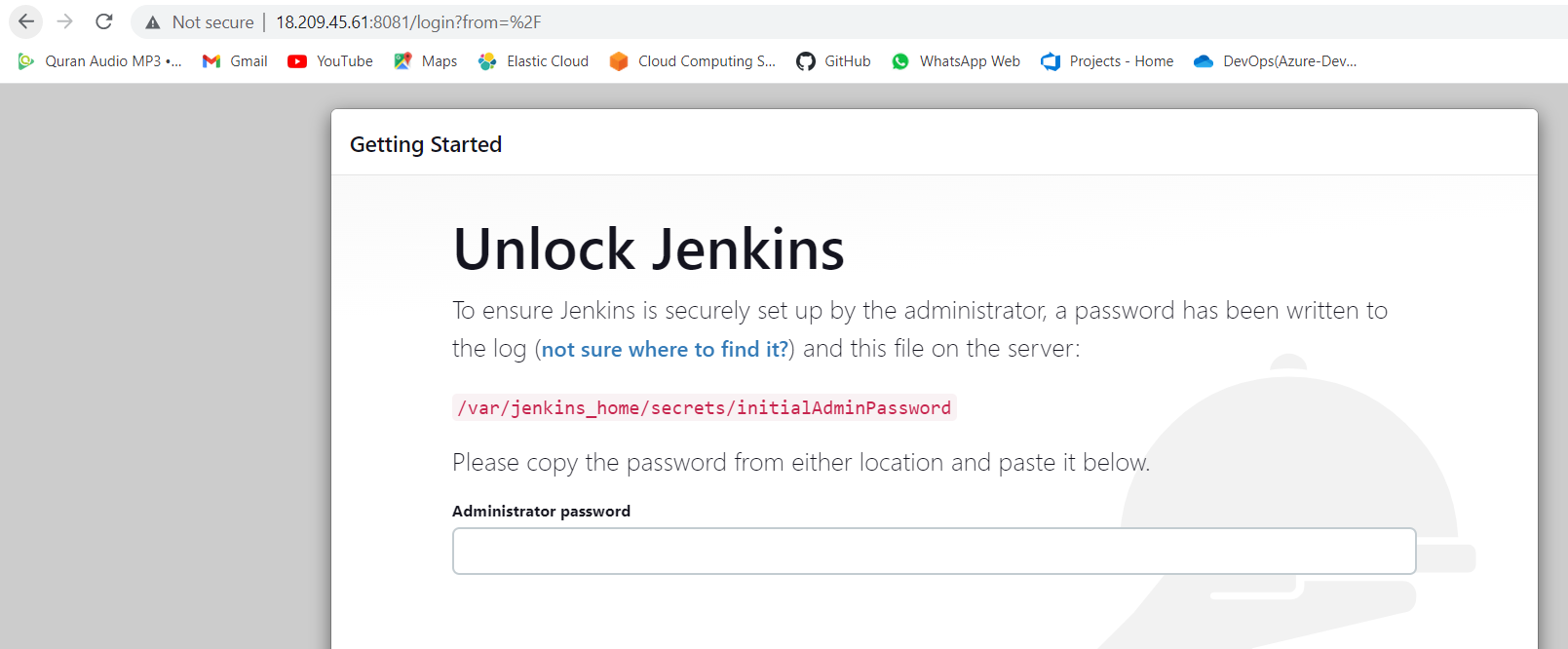
Docker run -d –name containername -p 80:80 **(imagename)**

After this go to your browser and type ipaddress:80 to see this page as below



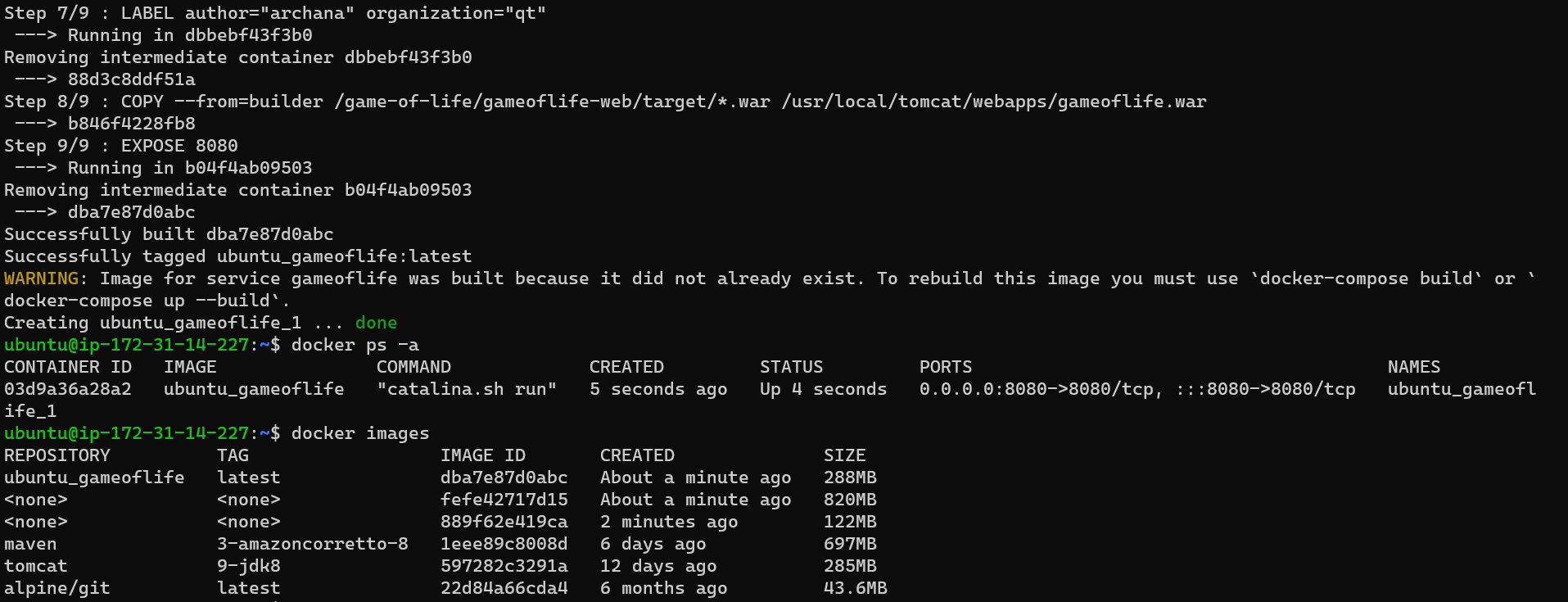
1. **Create a Jenkins image by creating an own docker file**

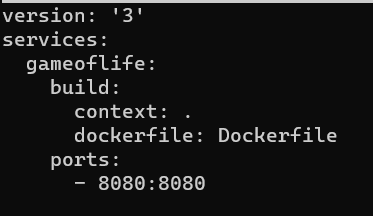


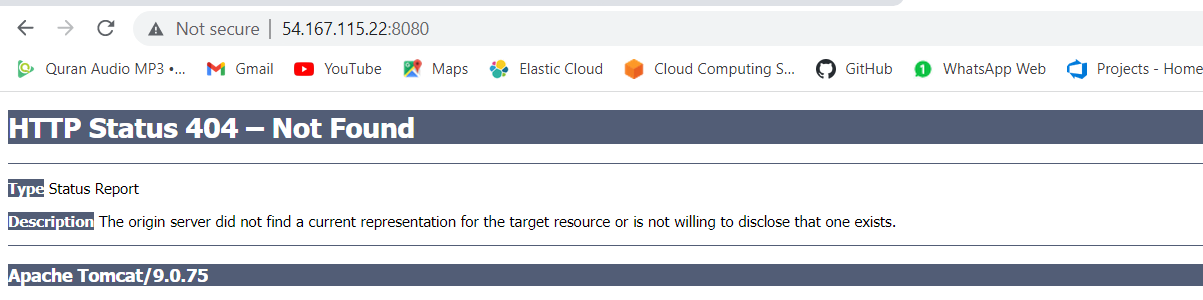


**DAY 5**

**Docker compose file for game of life**

****

****

****

**How does docker compose work?**

* You can define all of the configuration in one YAML file, and then start all the containers with one command. Rather than having all your services in one big container, Docker Compose allows you to split them up into individually manageable containers.

What is docker compose ?

Docker Compose is a tool provided by Docker that allows you to define and manage multi-container applications. It provides a simple way to define the services, networks, and volumes required for your application as a YAML file called **docker-compose.yml**

Why do we use Docker compose ?

Docker Compose is used for several reasons:

**Simplifying multi-container applications**: Docker Compose simplifies the management of multi-container applications by allowing you to define and manage multiple services.

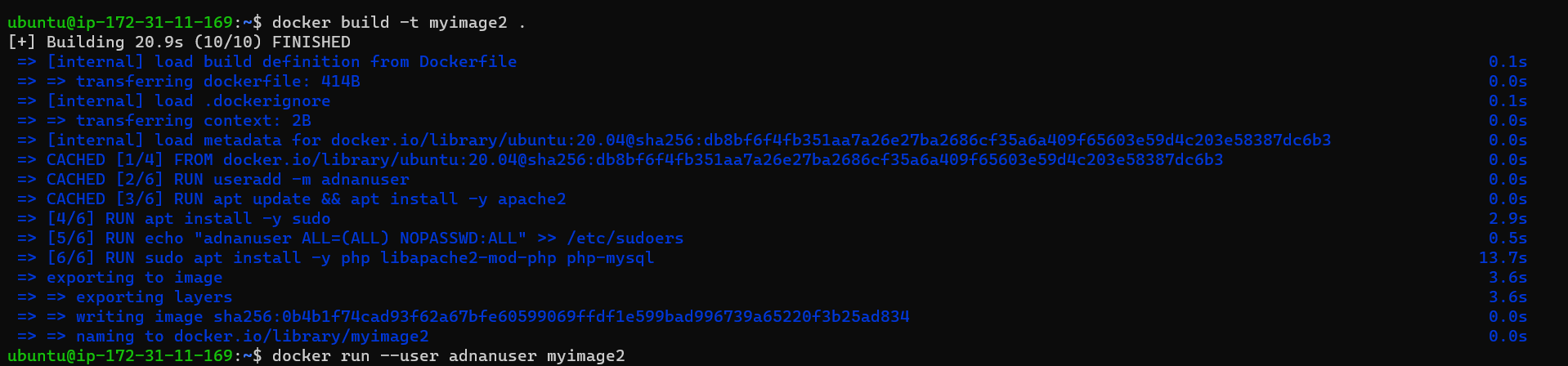
**Standardization and reproducibility**: Docker Compose enables you to define your application's infrastructure and dependencies in a declarative manner using a YAML file.

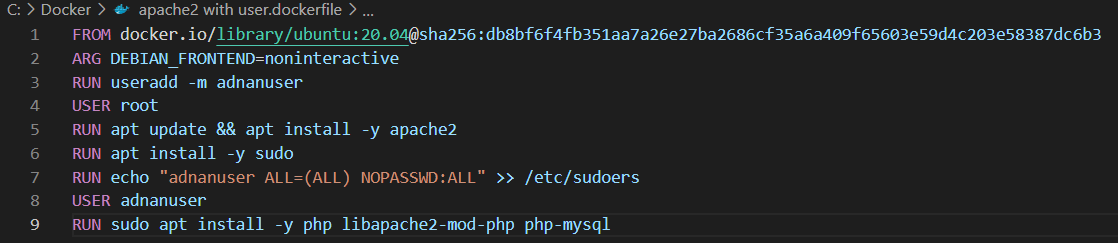
**Isolation and encapsulation**: Docker Compose helps achieve isolation between different components of your application by running them in separate containers.

**Efficient development workflows**: Docker Compose facilitates efficient development workflows by providing a convenient way to set up and tear down development environments.

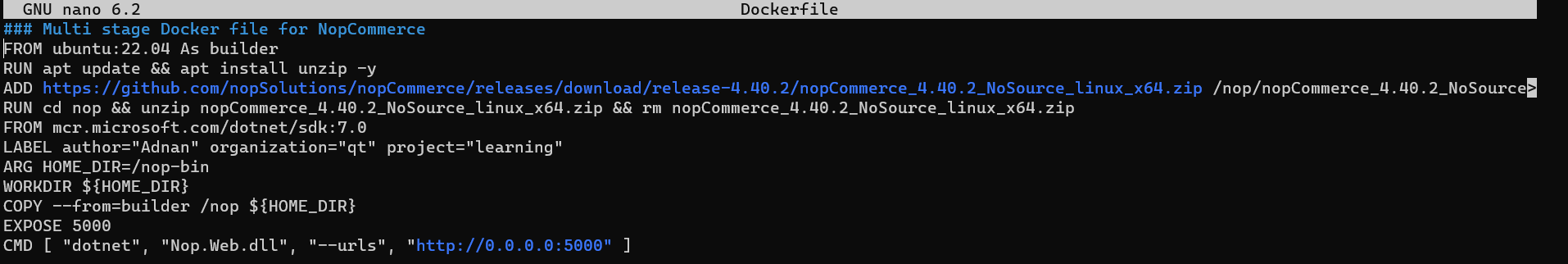
**Service orchestration and scalability**: Docker Compose allows you to define and manage the relationships between services in your application.

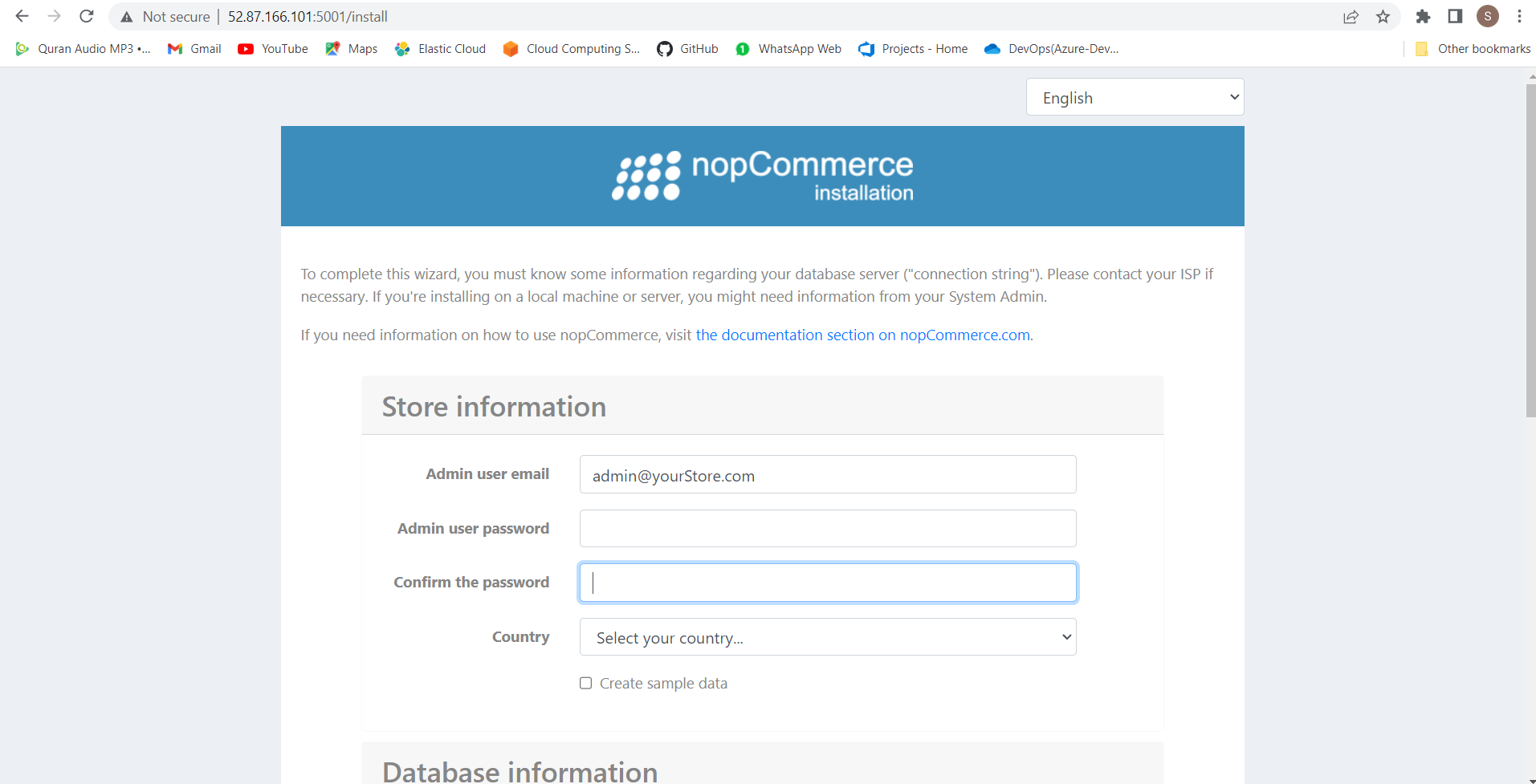
**Running containers with specific User ?**





**Create nopCommerce and MYSQL server and try to make them work by configuring**

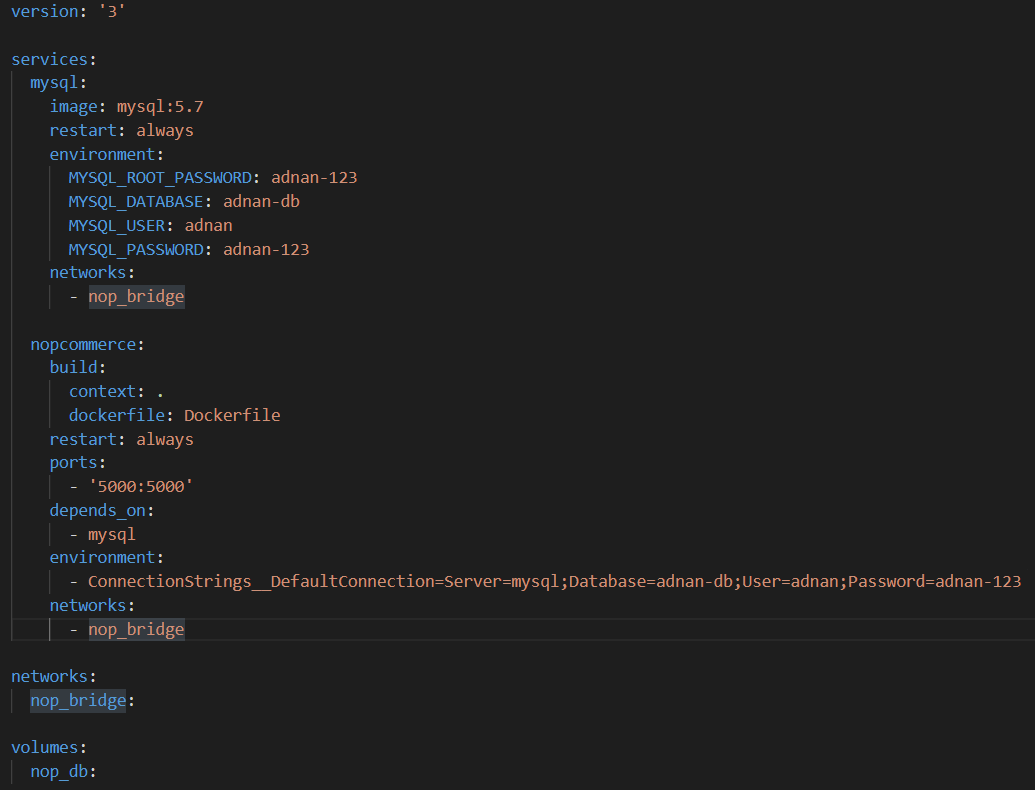
****

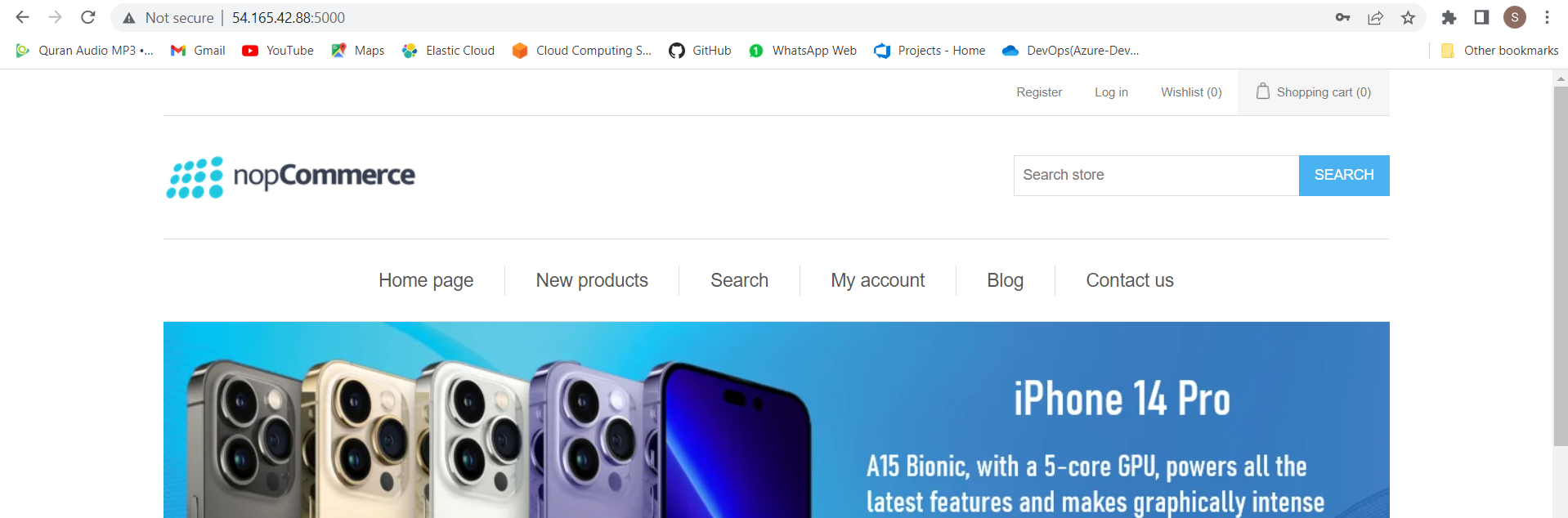
****

For configuration we need docker-compose file to build both nop and MYsql

So we can run both containers with single command

**Docker-Compose file for Nop and MYsql**

****



**To configure This task**

First install docker in your machine with this commands

sudo apt update

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

sudo usermod -aG docker ubuntu

after installing check whether if it is downloaded or not **{docker info}**

Install docker-compose **{sudo apt install docker-compose -y}**

Then create a directory **{mkdir anyname}**

Enter in that directory by **{cd}** command

add a Dockerfile **{nano Dockerfile}**

After this now we have to add a compose file **{nano docker-compose.yaml}**

Then run both docker-compose.yaml and Dockerfile with only one single command **{docker-compose up -d}**

After completion of building check container status

**{docker ps -a}**

Now final step is go to browser and your instance ip **{ipaddress:5000}**

And configure with mysql details

And u will nopcommerce page

**NOTE**: we need t2.medium ubuntu 20.04 for this task it won’t work on t2.micro

1. **Multi stage Docker file and push images to azure/aws registries and docker compose file for following applications**

**nopCommerce**

**For this Task:**

First install docker in your machine with this commands

sudo apt update

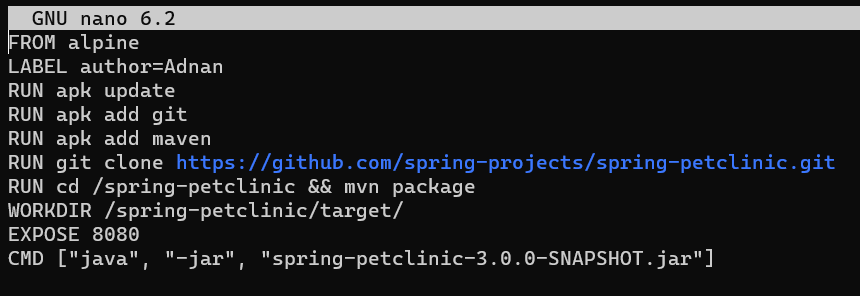
curl -fsSL https://get.docker.com -o get-docker.sh

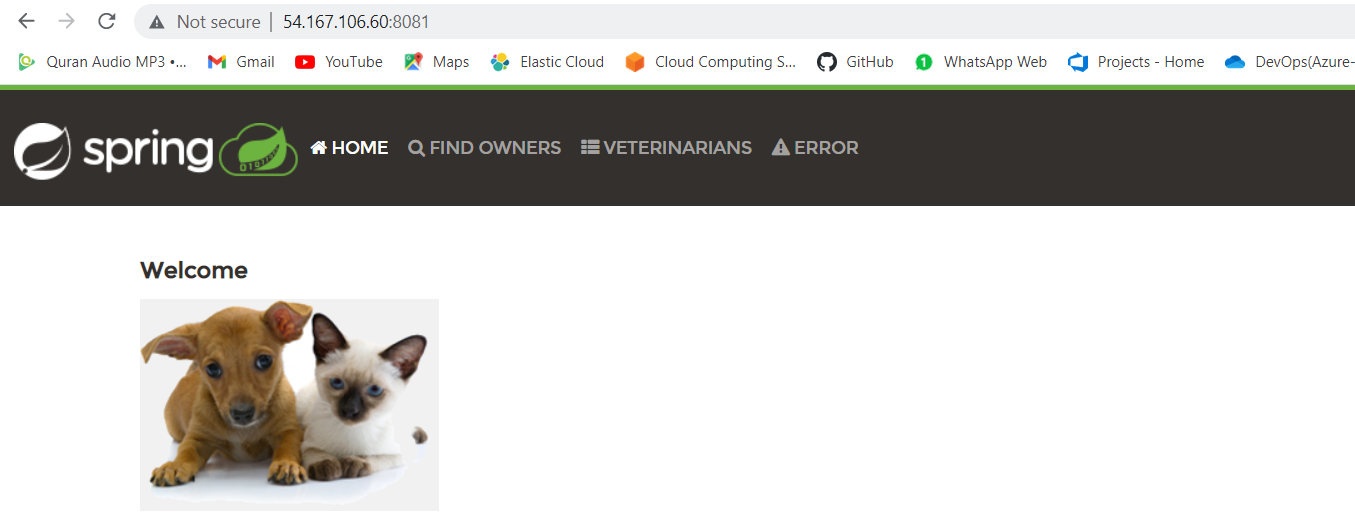
sudo sh get-docker.sh

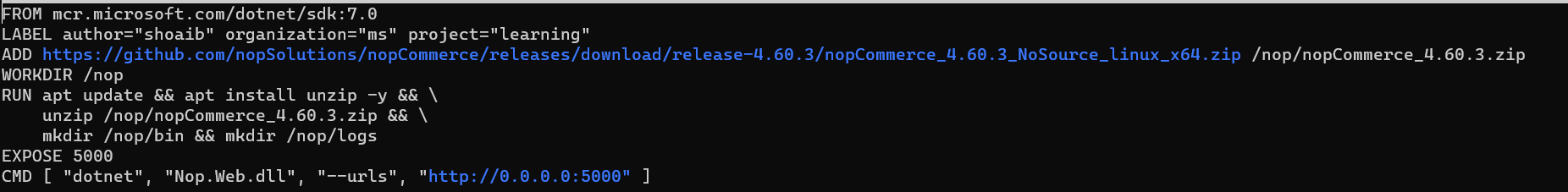
Then create a directory {mkdir multi}

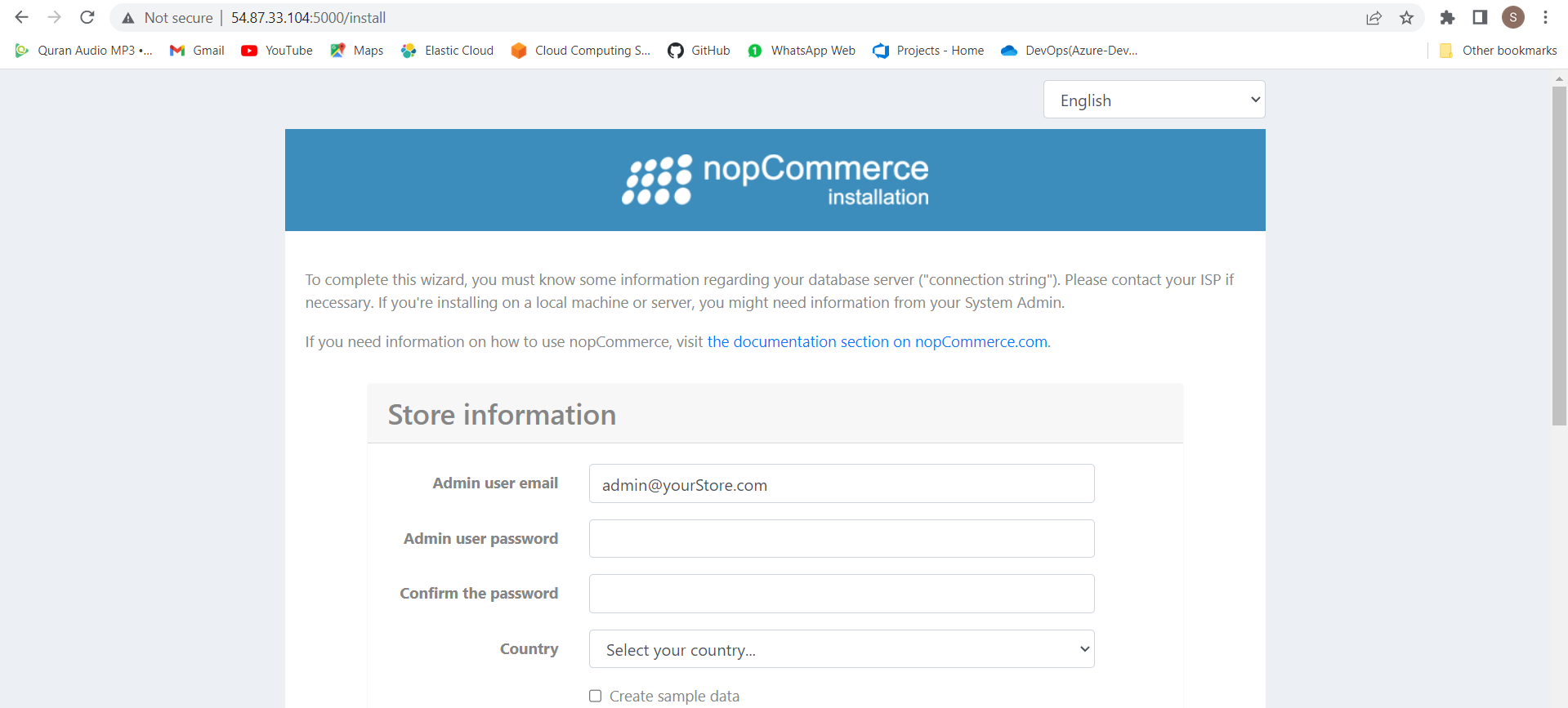
Enter into directory and make files

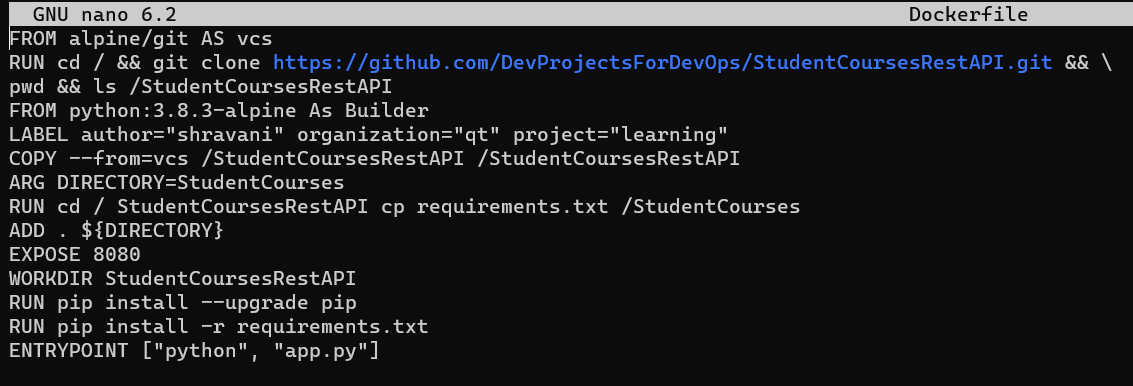
1. Spring-petclinic

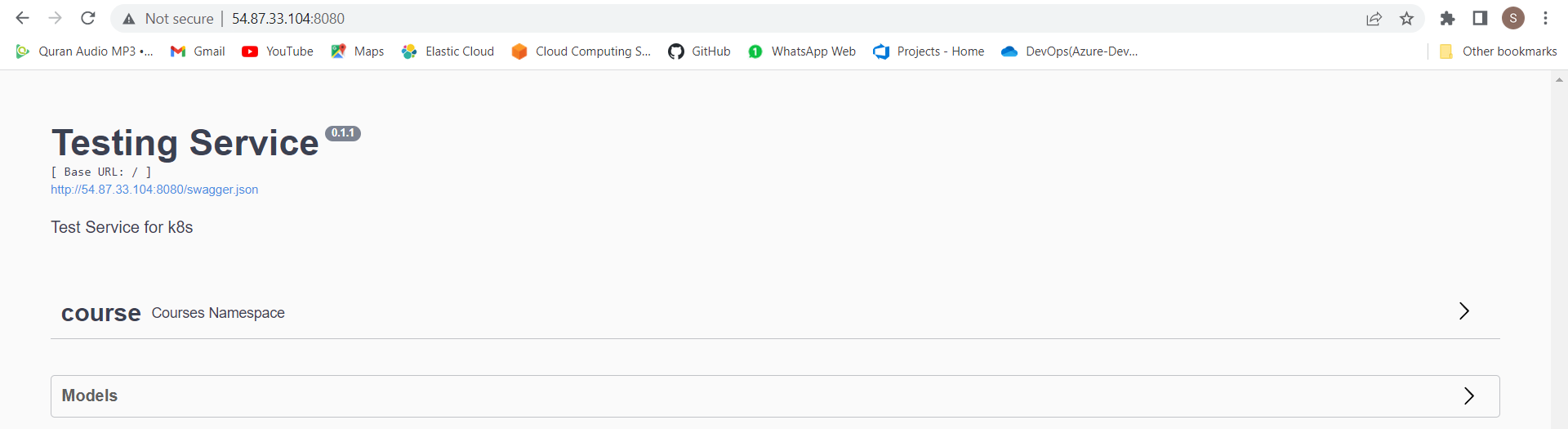




1. NopCommerce
2. ****



1. StudentCourseRegister



In spc folder Attach spc dockerfile

In Nop folder attach nopcommerce dockerfile

In Src folder attach src dockerfile

Then come back from folders

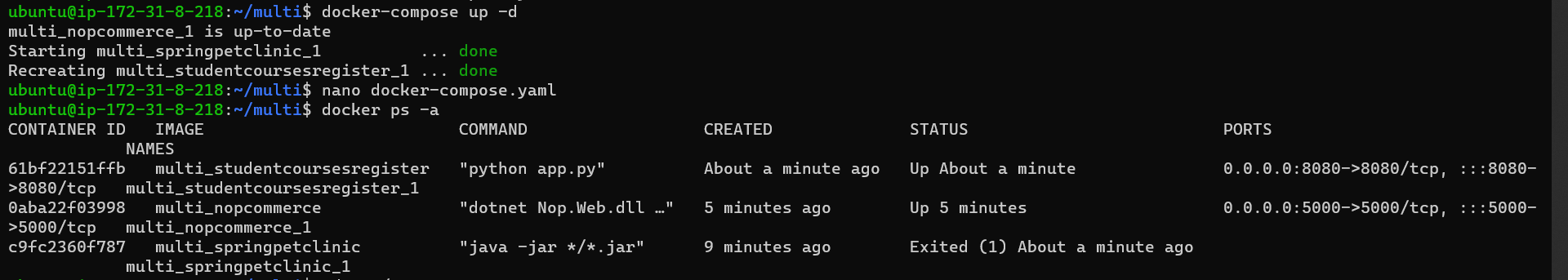
In multi directory make a docker-composefile.yaml

And run all three dockerfiles with one single command

**{docker-compose up -d}**

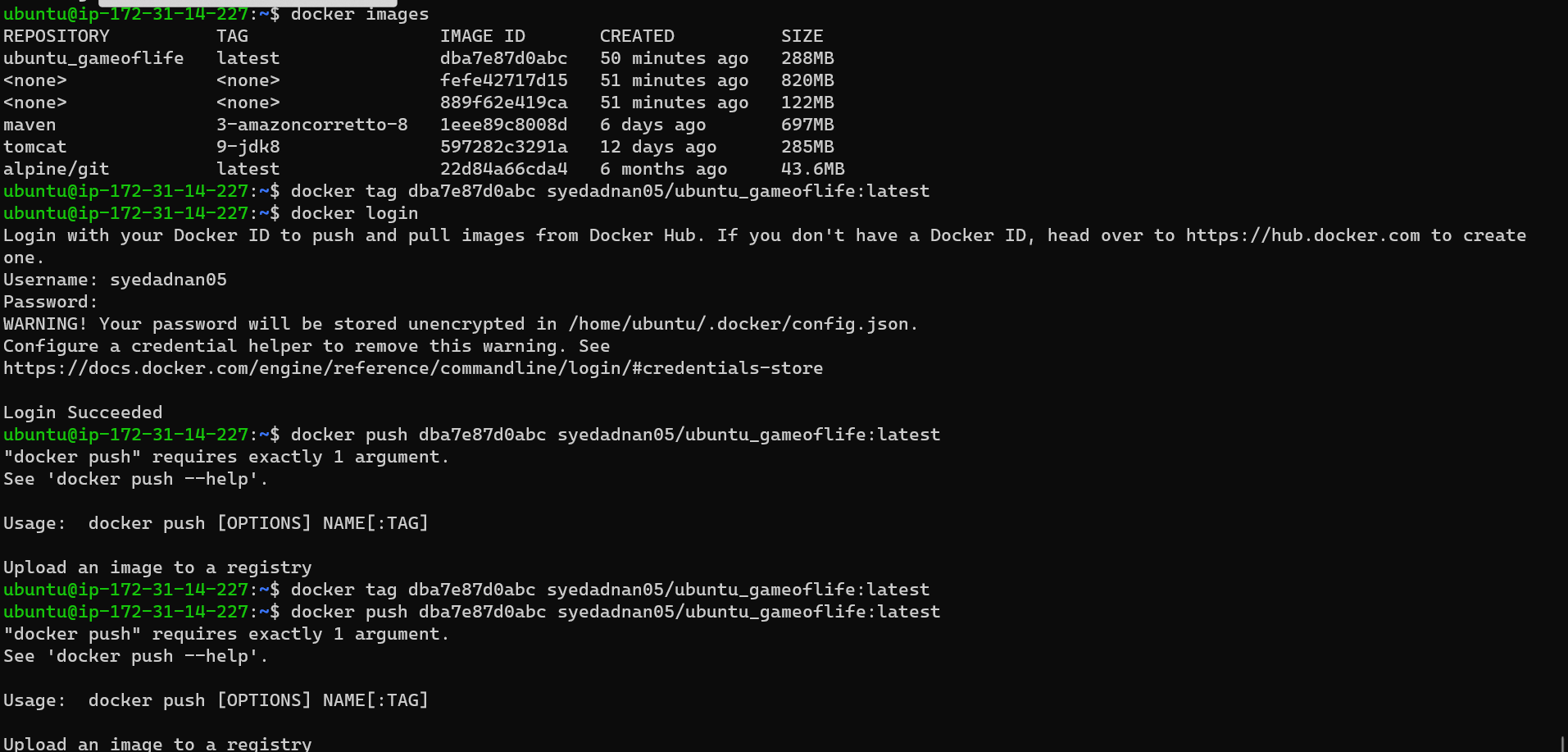
After building of 3 images

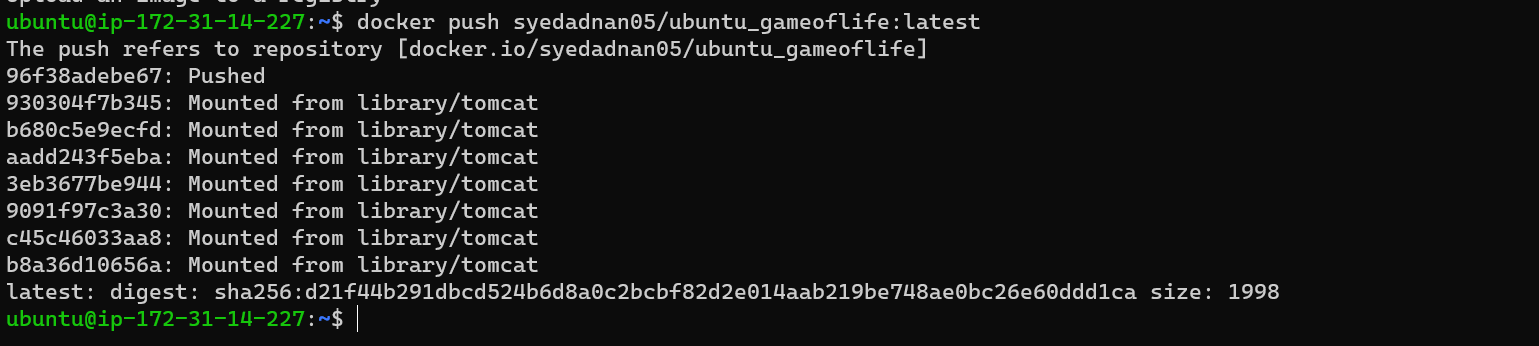
Check their containers **{docker ps -a}**



**PUSH IMAGES TO DOCKER HUB**

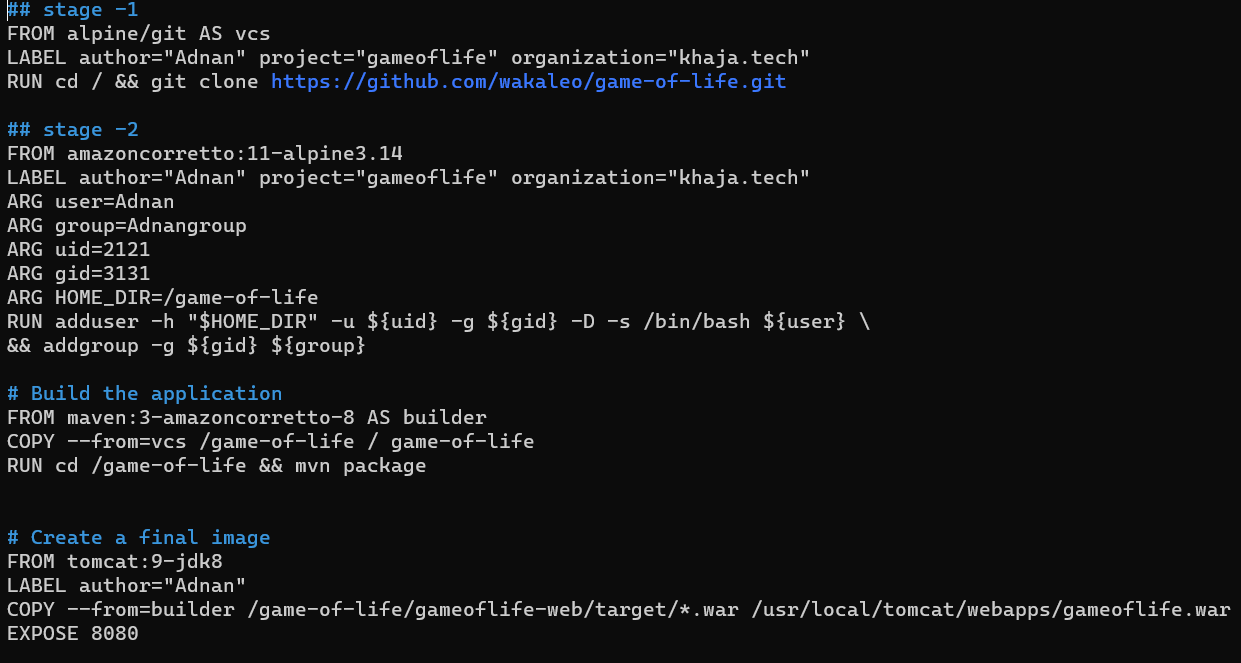
* 1. **Install docker in your machine**
  2. **Write a Docker file and build an image**
  3. **Then tag image {docker tag imageid syedadnan05/ubuntu\_gameoflife:latest}**
  4. **Then push your image {docker push syedadnan05/ubuntu\_gameoflife:latest}**

****

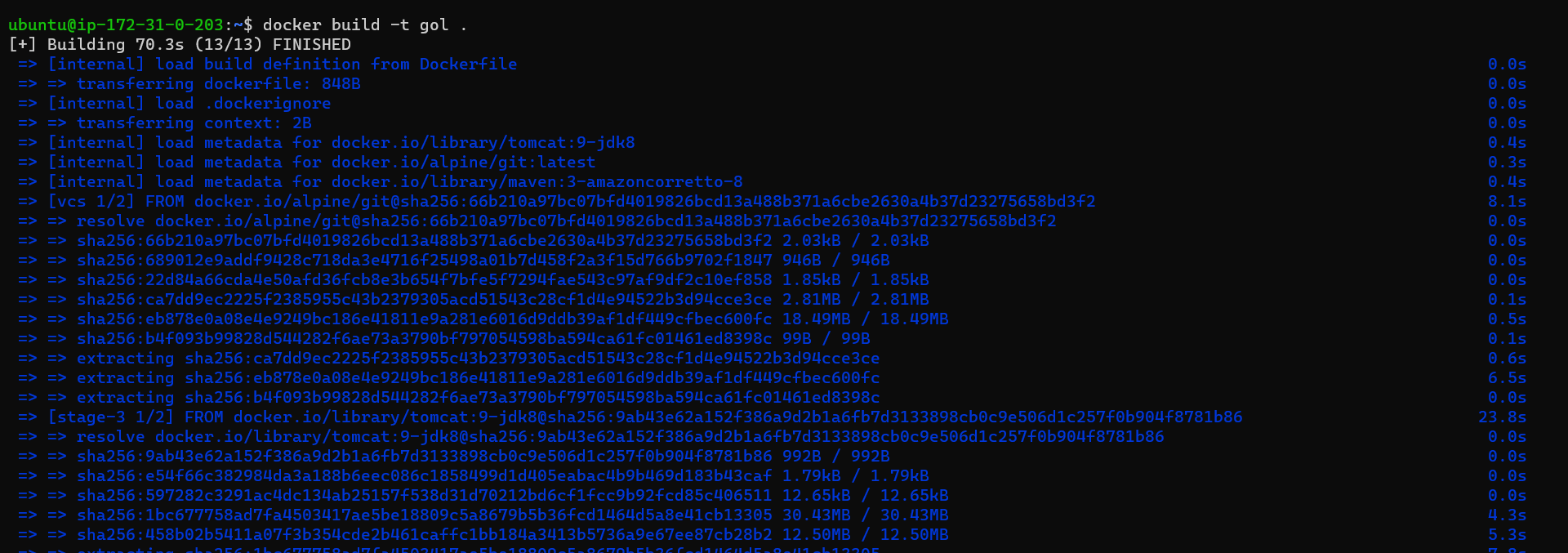
****

**Running container with a specific user**

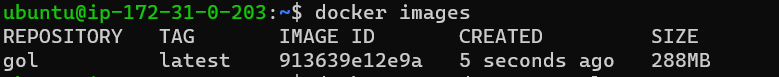
* 1. First Install dockerin your machine



* 1. And build the image



* 1. Check whether image is build or not



* 1. After this create a container with a specific user

{docker run -d --name gol-cont --user 2121 -p 8080:8080 gol}

