**Capstone project documentation**

1. **React application:**

It is a web application built using React, which is a JavaScript library for building user interfaces. Developed and maintained by Facebook, React allows developers to create large web applications that can update and render efficiently in response to data changes.

React applications are known for their performance, modularity, and ease of maintenance.

They are widely used for developing modern web applications, including social media platforms, e-commerce sites, dashboards, and more.

1. **Node.js :**

**1. Node.js is a JavaScript-based platform for server-side and networking applications.**

2. It is a runtime environment that allows developers to run JavaScript code on the server side.

3. Node.js is commonly used for building web servers, real-time applications (like chat applications), APIs, and microservices, among other types of applications.

4. Node.js is a software platform for scalable server-side and networking applications.

5. Node.js applications are designed to maximize throughput and efficiency.

6. Node.js internally uses the Google V8 JavaScript engine to execute code.

Developer gives

1. .json file 🡪it denotes write docker file for node.js
2. .txt file 🡪 it denotes write docker file for python
3. .jar file 🡪 it denotes write docker file for java
4. .xml file 🡪 it denotes write docker file for java.

🡪 But developer does not give .jar file. We can create .jar file using build tool [maven /gradel ]

1. Maven [pom.xml]
2. Gradel [build.gradle]

|  |  |  |
| --- | --- | --- |
| File format | Docker file | Package management tool |
| package. json | Node.js | Npm [Node Package Manager] |
| requirement.txt | python | pip |
| app.jar  pom.xml | java | Maven [pom.xml]  Gradel [build.gradle] |

Suppose package.json file [ default file name] not given in github repo, it means it’s an already build application. For run that application, we need web server [ nginx or Apache ]

**difference between docker file and docker compose file**

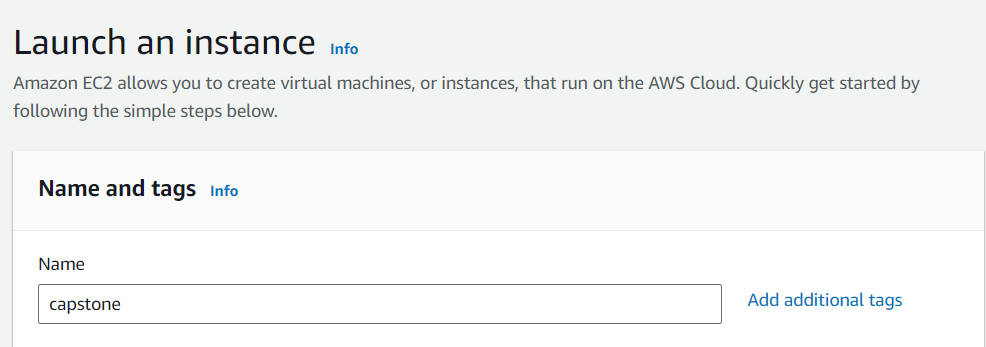
|  |  |  |
| --- | --- | --- |
|  | **Docker file** | **Docker compose file** |
| **purpose** | Defines how to build a single Docker image. | Defines how to run multi-container applications.it is used for running multiple containers. |
| **File type** | Text file | .yaml / yml file |
| **syntax** | FROM, RUN, COPY, etc | YAML syntax to define services, networks, and volumes. |
| **Content** | It Contains instructions to set up an environment inside a container | It contains service definitions and configurations for running containers |

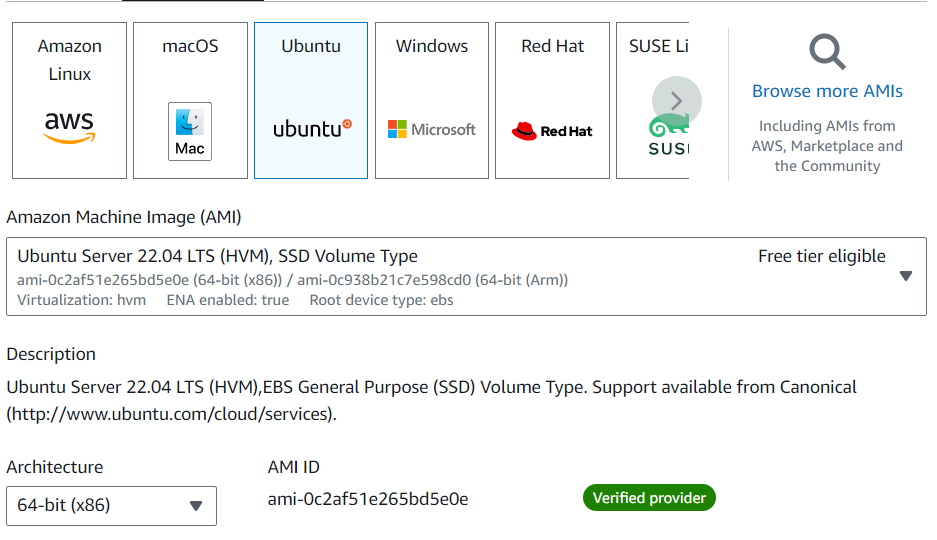
By understanding these differences, you can effectively use both Docker file and Docker Compose together to build and deploy complex Docker-based applications.

**Installing the necessary software’s & services for this task:**

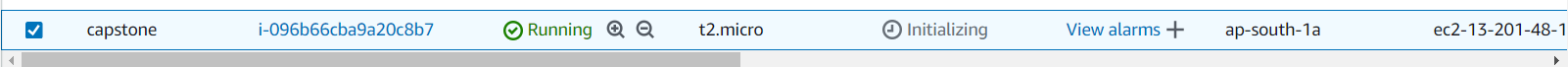
1. **Git**
2. **Docker**
3. **Docker compose**
4. **Java**
5. **Jenkins**

Before install git. we have to launch an instance. select ubuntu









Connect that instance

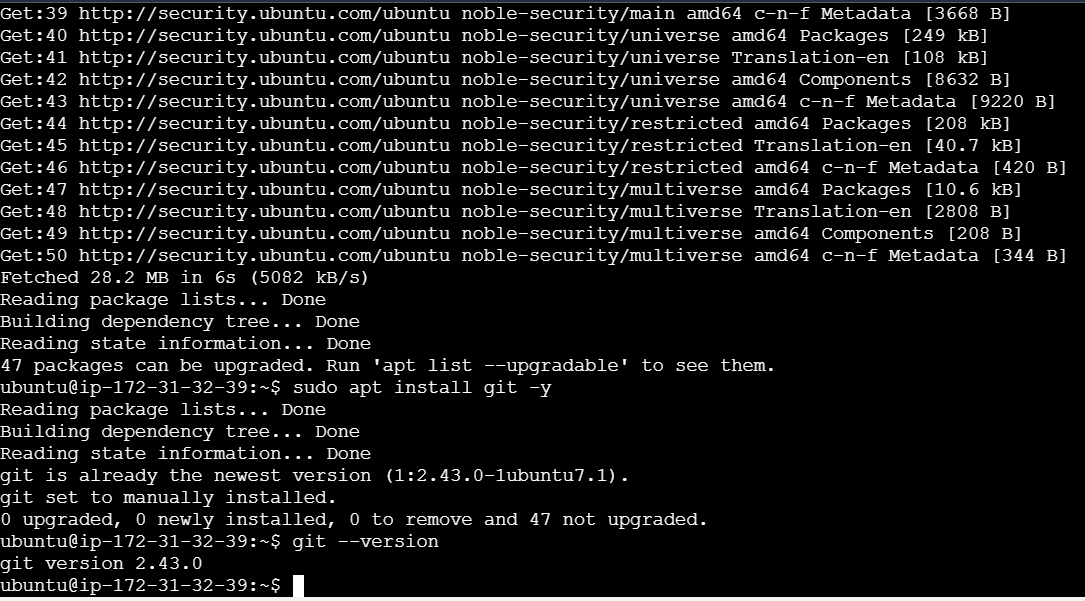
1. **Git installation:**

sudo apt update

sudo apt install git -y

git –version

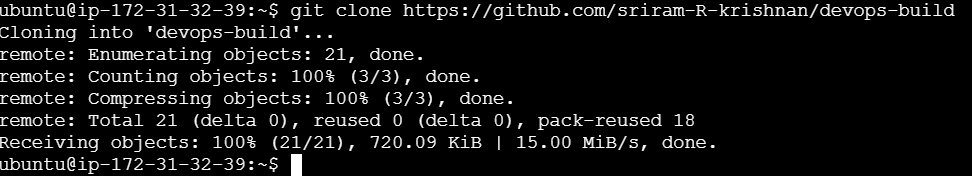
sudo apt update





Clone the given repo

git clone <https://github.com/sriram-R-krishnan/devops-build>





1. **Docker installation:**

sudo apt update

sudo apt install docker.io

sudo docker version

sudo usermod -aG docker ubuntu

sudo systemctl status docker

sudo systemctl start docker

**after that, I got some error regarding permission.so I give below commands. search the below commands in chatgpt**

sudo usermod -aG docker ubuntu

newgrp docker

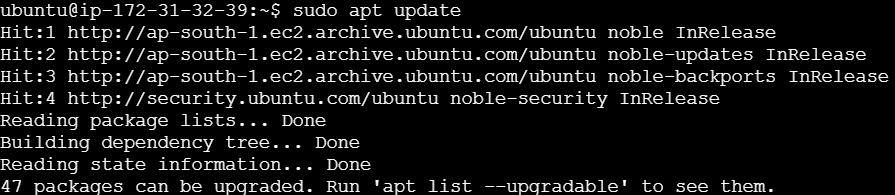
sudo systemctl start docker

sudo systemctl enable docker

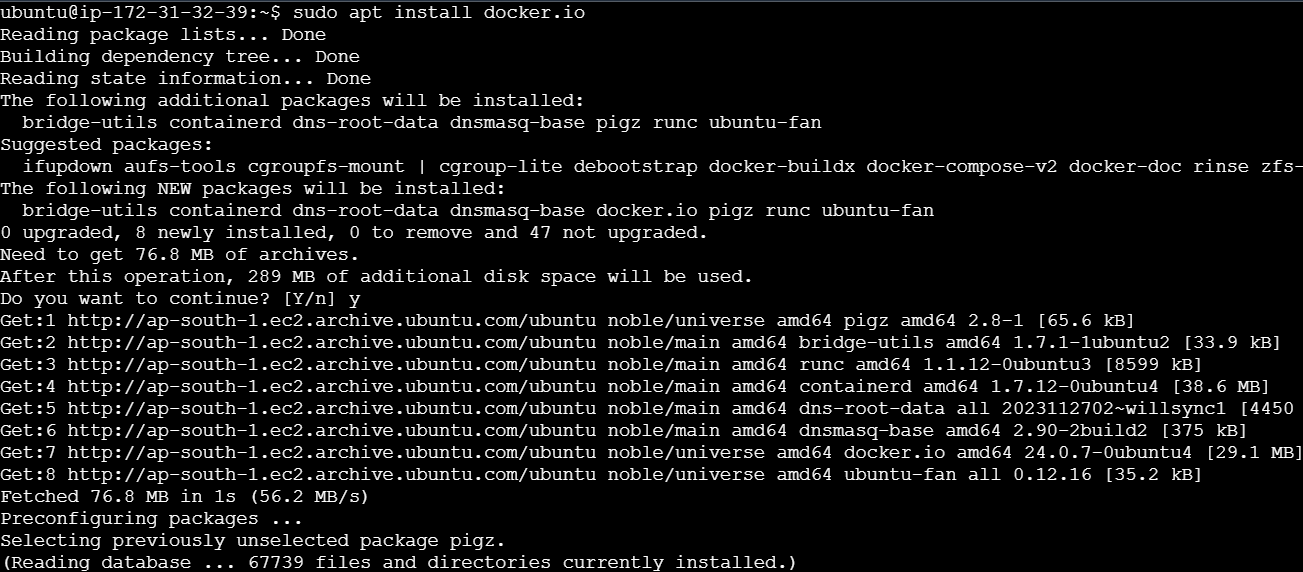
just check, docker can pull the image from docker hub. This command not needed for this task

sudo docker run hello-world

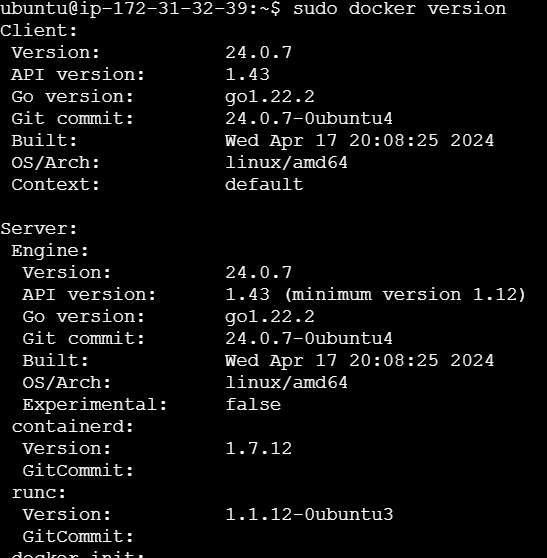
sudo apt update



sudo apt install docker.io



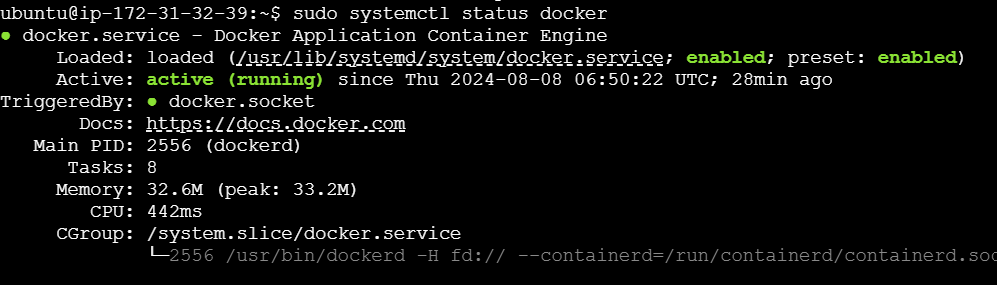
sudo docker version



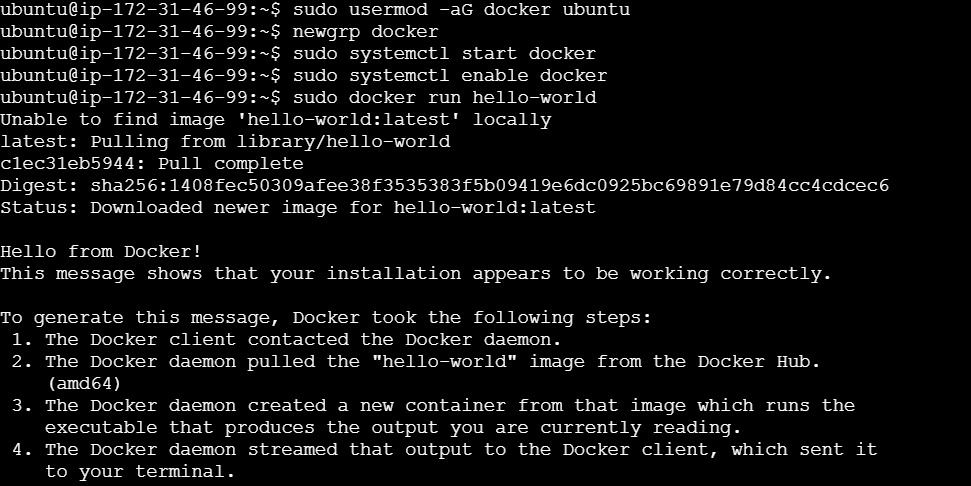
sudo usermod -aG docker ubuntu

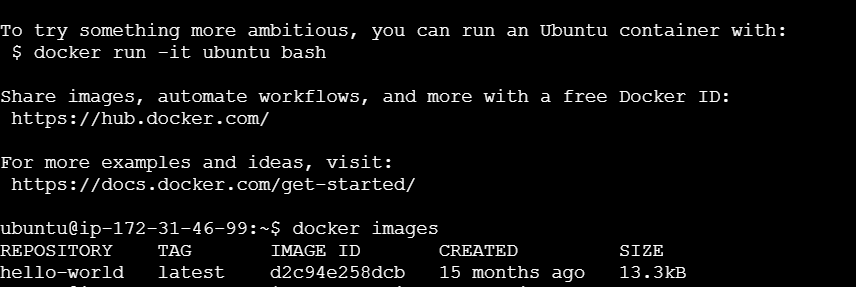


sudo systemctl status docker



sudo systemctl start docker





**docker file:**

* In our task given application was already built.
* So, we did not need to write node.js docker file. Just write docker file for nginx server.
* For running the application, we need web server. So, write docker file for nginx web server.
* Because in that server we have to run an already built application.
* **Docker file written in the folder devops-build [in given github this folder available].**
* **so, get into the folder [ cd devops-build]**

**Explanation of the Dockerfile:**

1. **FROM nginx:latest**:
   * This line specifies the base image to use for the Docker image. In this case, it pulls the latest version of the official Nginx image from Docker Hub.
2. **WORKDIR /usr/share/nginx/html**:
   * This sets the working directory inside the container to /usr/share/nginx/html. This is the default directory where Nginx serves static files (like HTML, CSS, and JavaScript).
3. **COPY build/ .**:
   * This copies the contents of the build/ directory [ this dir in the given repo]on your host machine into the current working directory in the container (/usr/share/nginx/html). This typically contains the static files that your web application will serve.
4. **EXPOSE 80**:
   * This line informs Docker that the container will listen on port 80 at runtime. Port 80 is the default HTTP port, so this is where Nginx will serve the web application.
5. **CMD ["nginx", "-g", "daemon off;"]**:
   * This command runs Nginx in the foreground (instead of in the background as a daemon), which is necessary for Docker containers so that the container doesn't exit immediately after starting. The command starts Nginx and keeps it running.

**vi dockerfile**

FROM nginx:latest

WORKDIR /usr/share/nginx/html

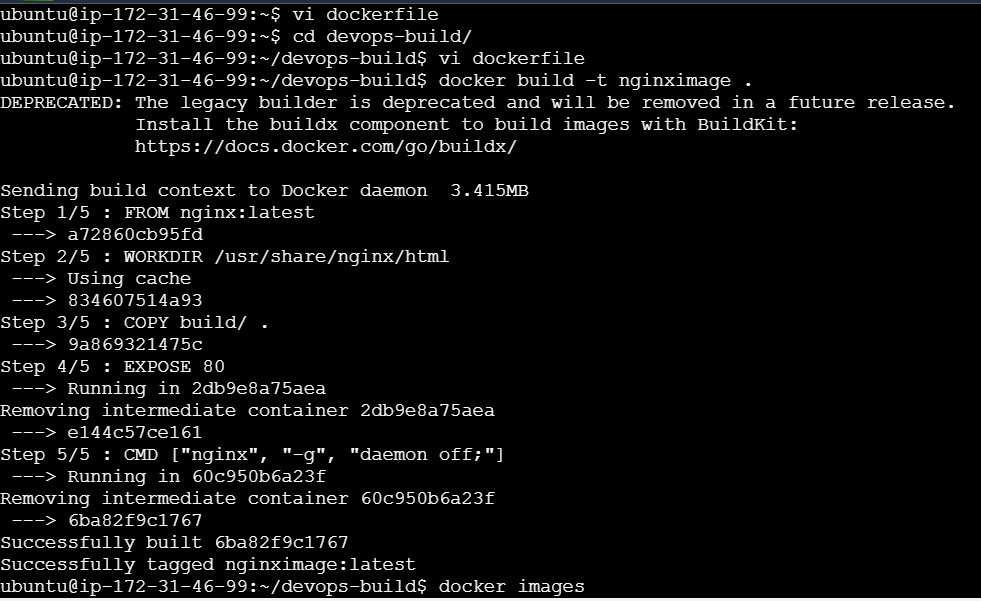
COPY build/ .

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

**Convert docker file into docker image:**

**docker build -t nginximage . [nginximage -we give that image name for our choice]. This command used for build docker image**

****



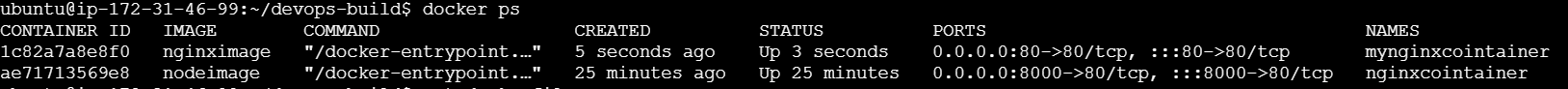
**docker images**

****

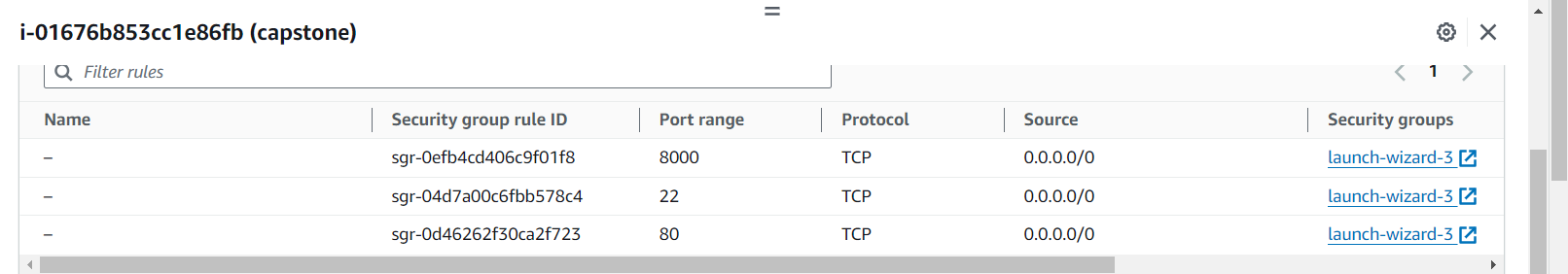
**docker run -d –name mynignxcointainer -p 80:80 nginximage**

****

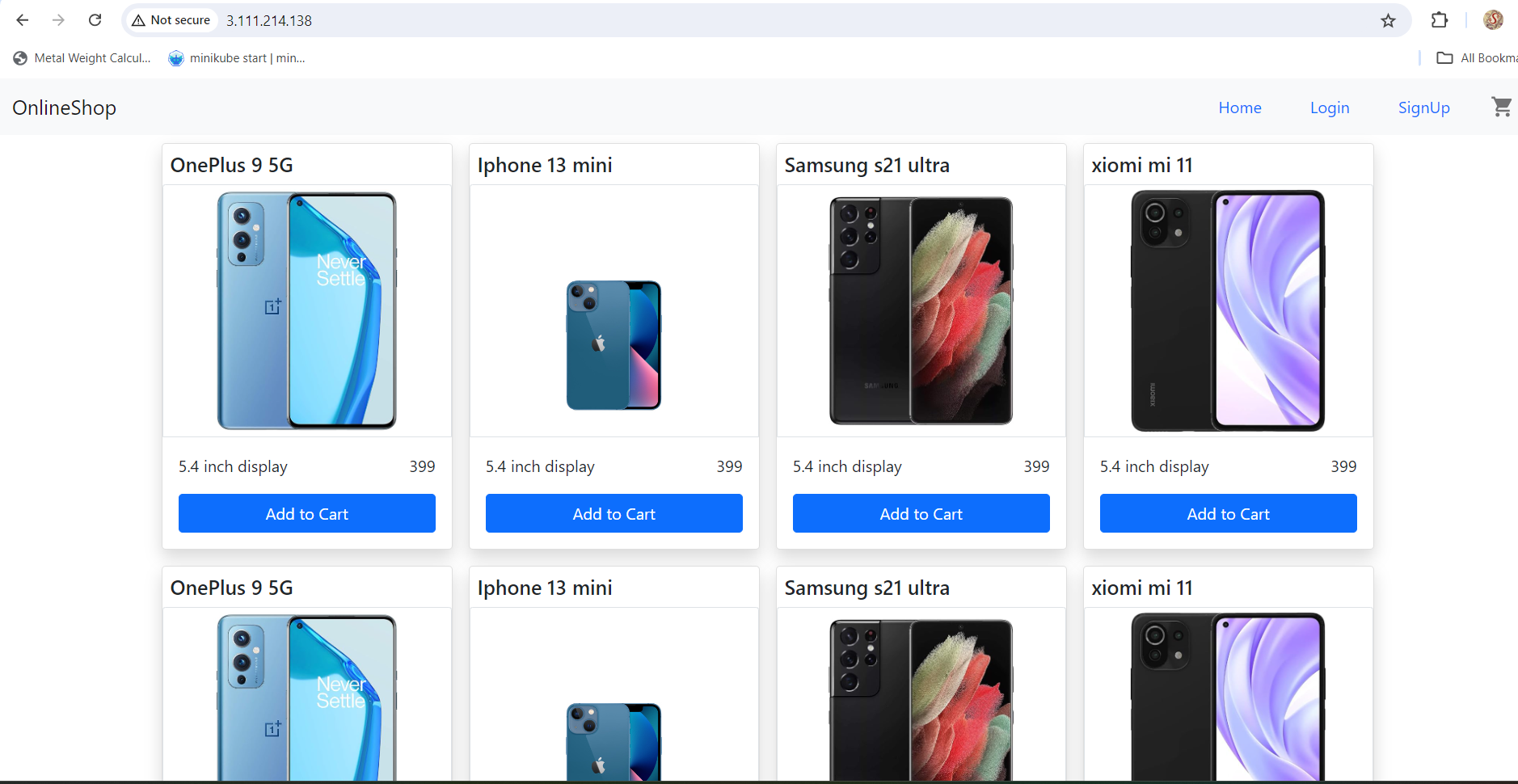
**docker ps**

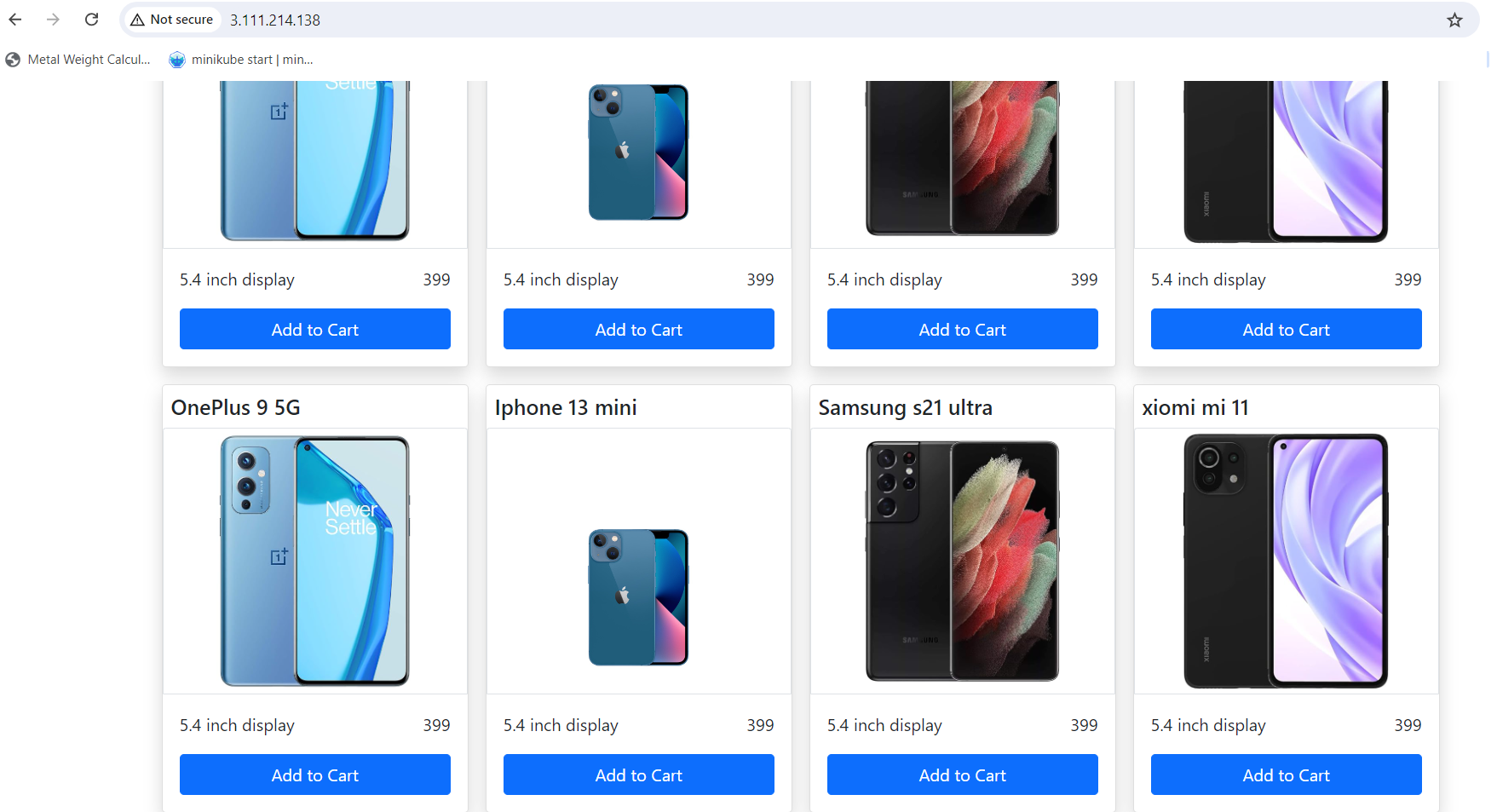
****

**Open the port no :80 in ec2. Copy and paste the ip address**

****

**Output page of given application:**

****

****

1. **Docker compose installation:**

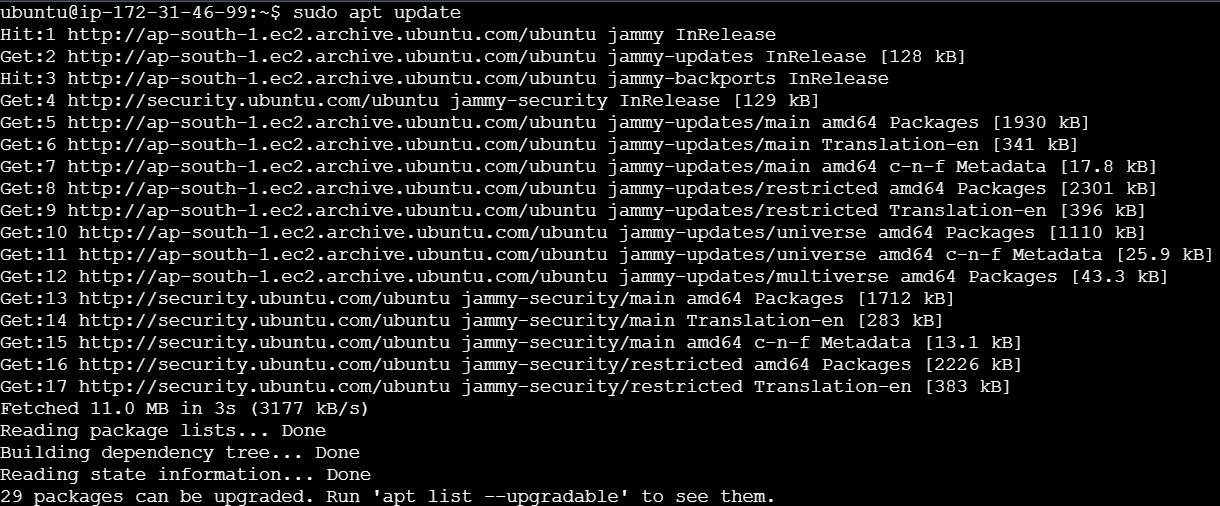
sudo apt update

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

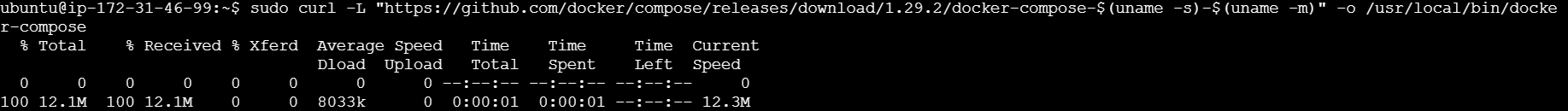
sudo chmod +x /usr/local/bin/docker-compose

docker-compose –version

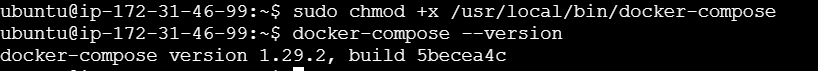
sudo apt update



sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose



sudo chmod +x /usr/local/bin/docker-compose

****

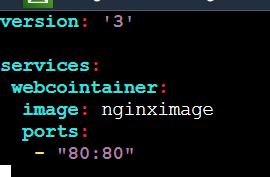
docker-compose –version

**Docker compose file for node js:**

**getinto folder devops-build. Then write this**

****

**vi docker-compose.yaml**

****

version: '3'

services:

webcointainer:

image: nginximage [ already we created docker image. give that image name]

ports:

* “80:80”

**Give permission for this file:**

**Chmod 777 docker-compose.yaml**

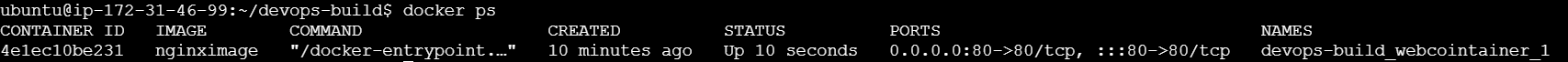
****

**Execute the docker-compose.yaml :**

**docker-compose up -d**

****

**docker ps**

****