

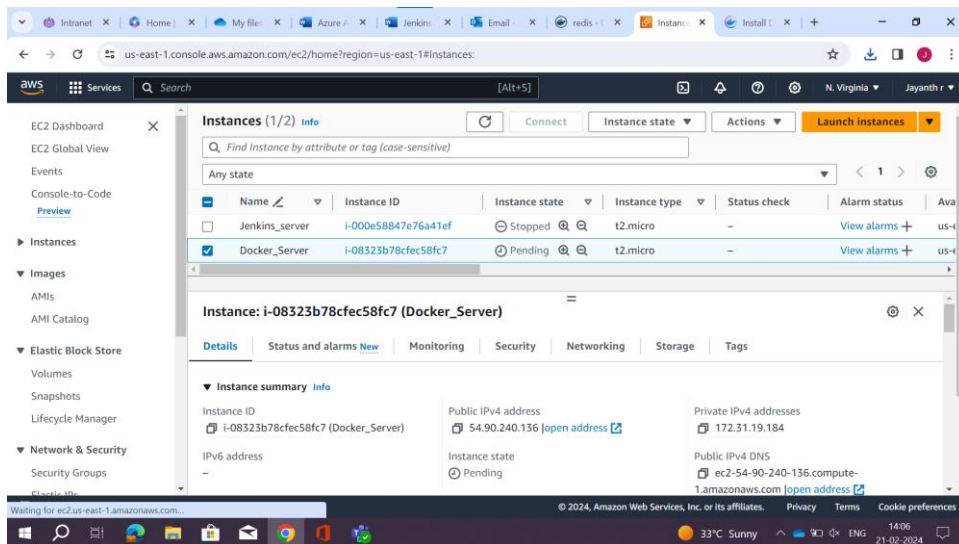
# AWS and Docker

Jayanth R

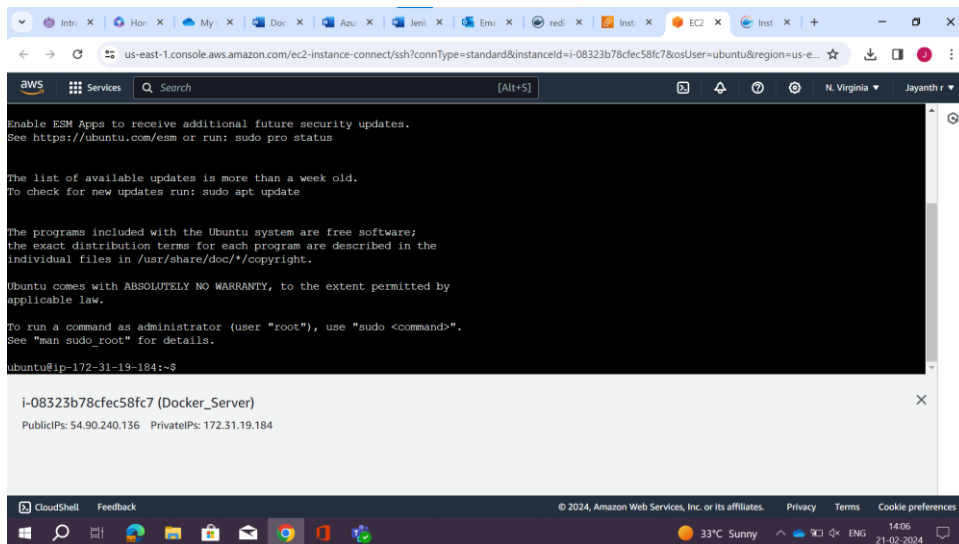
ACE11696

Here are the steps for using Docker in AWS.

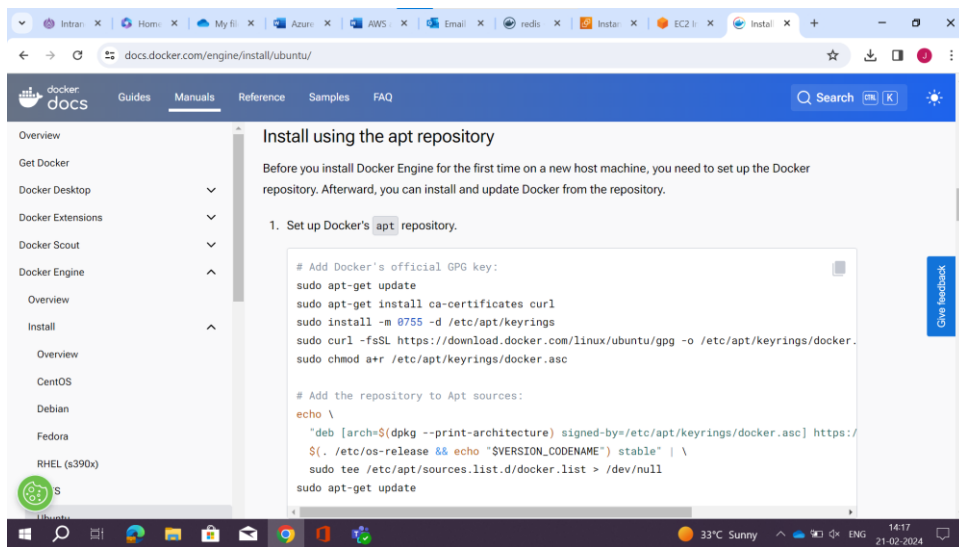
Step 1: I'm creating an Instance in AWS EC2 as "Docker\_Server".



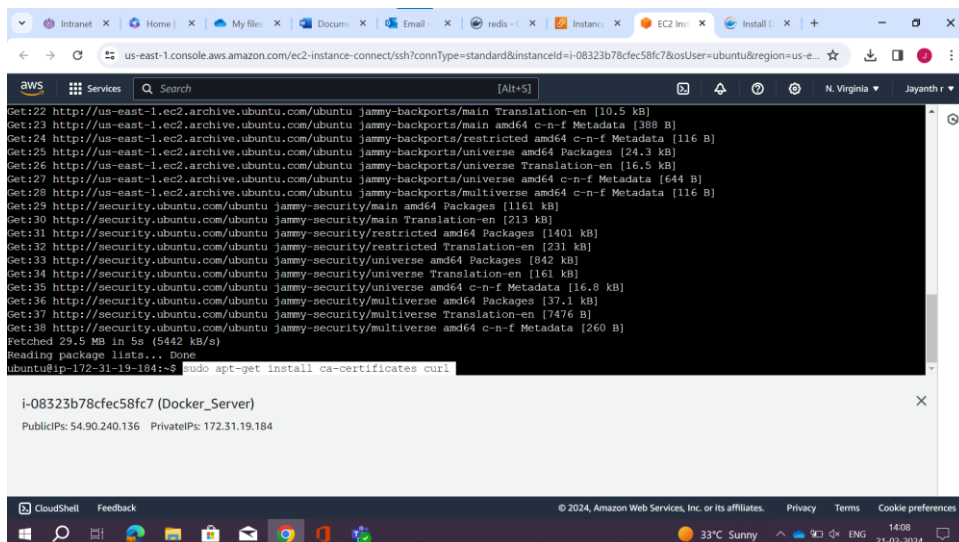
Step 2: Connet to the Instance "Docker\_Server" using EC2 Connect.



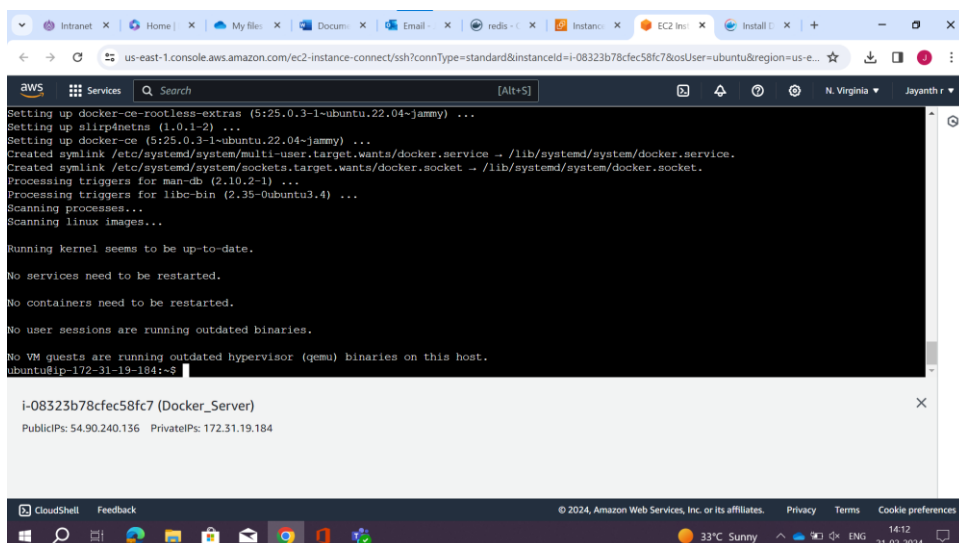
### Step 3: Now I'm Installing Docker using APT Repository.



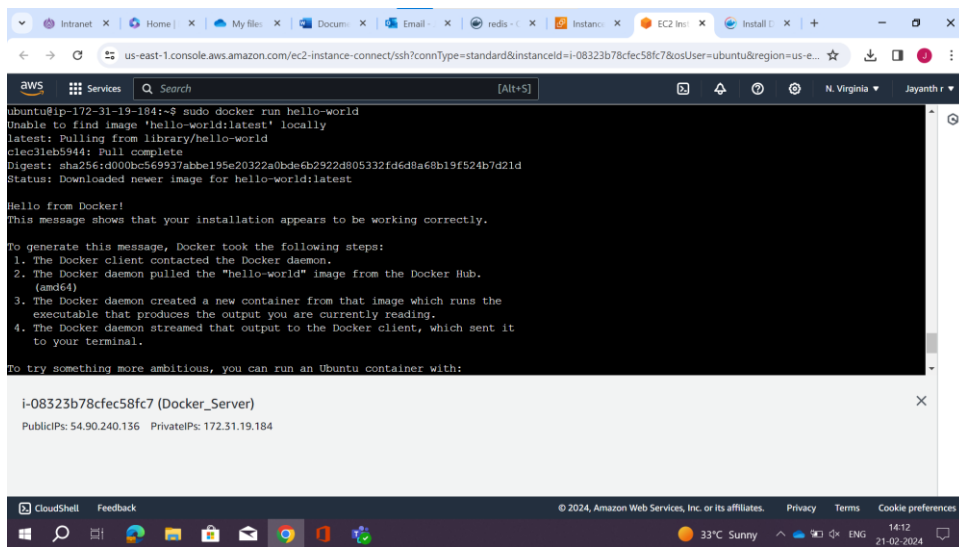
### Step 4: I'm running all the commands to set up APT repository.



### Step 5: Now i have successfully installed Docker in my EC2 Server.



Step 6: I'm running Hello World using command “docker run hello world”.



The screenshot shows a terminal window in the AWS CloudShell environment. The user has executed the command `sudo docker run hello-world`. The output indicates that the Docker client contacted the daemon, which then pulled the 'hello-world' image from the Docker Hub. The daemon created a new container from this image, and the container executed the 'hello-world' program, which printed a message: 'Hello from Docker! This message shows that your installation appears to be working correctly.' Below the message, a list of four steps explains the process. The terminal also shows the public and private IP addresses of the EC2 instance.

```
aws
Services
Search
[Alt+S]
N. Virginia
Jayanth r

ubuntu@ip-172-31-19-184:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d000bc569937abbel195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
Status: Downloaded newer image for hello-world:latest

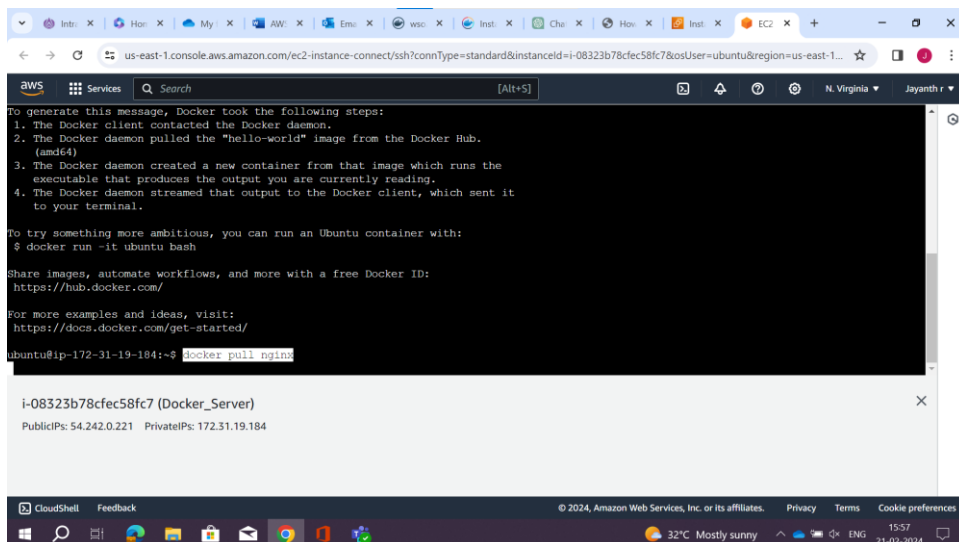
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

i-08323b78cfec58fc7 (Docker_Server)
PublicIPs: 54.90.240.136 PrivateIPs: 172.31.19.184
```

Step 7: Now here im using one image called “NGINX” from docker and pulling it.



The screenshot shows the same AWS CloudShell terminal window. The user has entered the command `docker pull nginx`. The terminal also displays the same 'Hello from Docker!' message and the list of steps as in Step 6. The public and private IP addresses of the EC2 instance are also visible.

```
aws
Services
Search
[Alt+S]
N. Virginia
Jayanth r

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

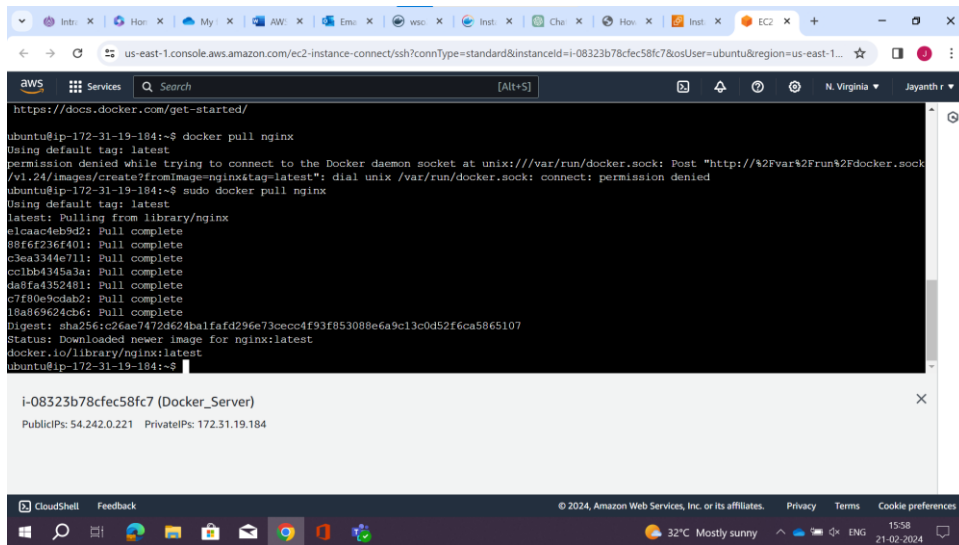
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

ubuntu@ip-172-31-19-184:~$ docker pull nginx

i-08323b78cfec58fc7 (Docker_Server)
PublicIPs: 54.242.0.221 PrivateIPs: 172.31.19.184
```

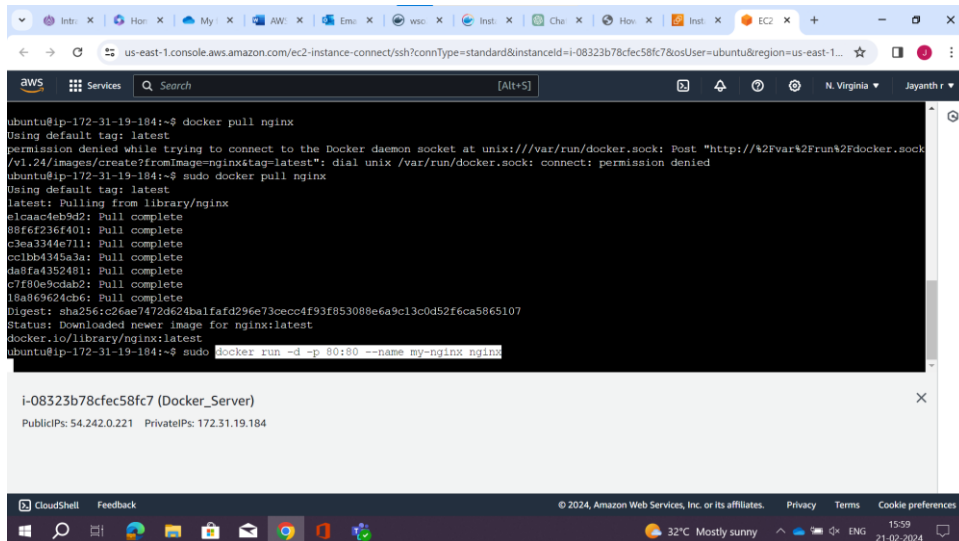
Step 8: You can see “NGINX” image is downloaded.



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output shows the command `docker pull nginx` being executed. It starts with a permission denied error, then successfully pulls the `nginx:latest` image. The output lists several layers being pulled and their SHA256 hashes, followed by the digest and status: `Status: Downloaded newer image for nginx:latest`. The terminal prompt is `ubuntu@ip-172-31-19-184:~$`. Below the terminal, a metadata box for the instance `i-08323b78cfec58fc7 (Docker_Server)` is visible, showing public and private IP addresses. The bottom of the screen shows the AWS CloudShell toolbar and a Windows taskbar with the date 21-02-2024.

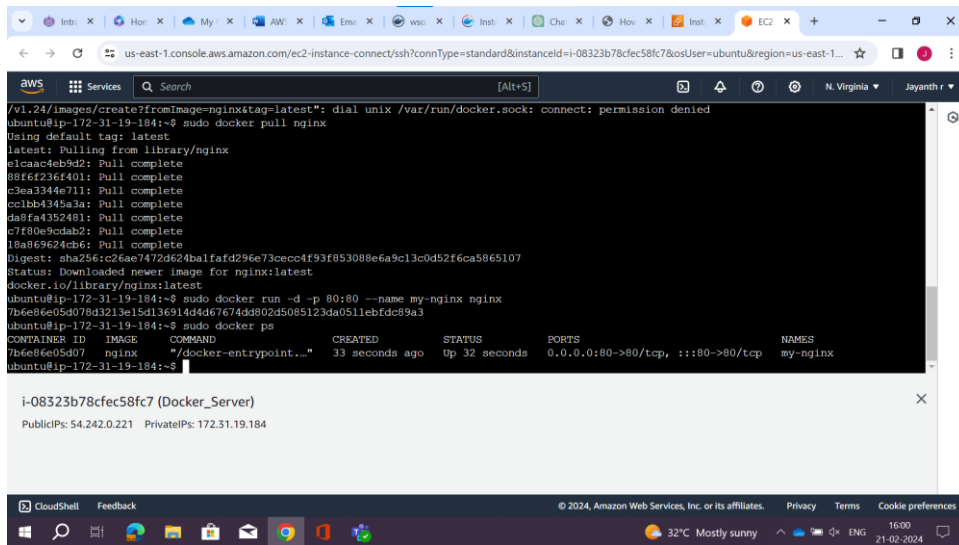
```
ubuntu@ip-172-31-19-184:~$ docker pull nginx
Using default tag: latest
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/create?fromImage=nginx&tag=latest": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-19-184:~$ sudo docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
e1caac4eb9d2: Pull complete
88f6f236f401: Pull complete
c3ea3344e711: Pull complete
cc1bb4345a3a: Pull complete
da9fa4352481: Pull complete
c7f80e9cdab2: Pull complete
18a869624cb6: Pull complete
Digest: sha256:c26ae7472d624balfafd296e73cecc4f93f853088e6a9c13c0d52f6ca5865107
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
ubuntu@ip-172-31-19-184:~$
```

Step 9: now im starting to run my image in background as a name “my-nginx”.



This screenshot is similar to the previous one, showing the same terminal output for `docker pull nginx`. However, the terminal prompt is now `ubuntu@ip-172-31-19-184:~$` and the next command `docker run -d -p 80:80 --name my-nginx nginx` is visible in the input line, ready to be executed. The rest of the interface, including the instance metadata box and the bottom toolbar, remains the same.

Step 10: Now you can see the image running successfully using command “docker ps”



```
/v1.24/images/create?fromImage=nginx&tag=latest": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-19-184:~$ sudo docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
e1caac4eb9d2: Pull complete
88f6f236f401: Pull complete
c3ea3344e711: Pull complete
cd1bb4345a3a: Pull complete
4a8fa4352481: Pull complete
c7f80e9cdab2: Pull complete
18a869624cb6: Pull complete
Digest: sha256:c26ae7472d624balfard296e73cecc4f93f853088e6a9c13c0d52f6ca5865107
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
ubuntu@ip-172-31-19-184:~$ sudo docker run -d -p 80:80 --name my-nginx nginx
7b6e86e05d078d3213e15d136914dd67674d3d802d5085123da0511ebfdc89a3
ubuntu@ip-172-31-19-184:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
7b6e86e05d07   nginx     "/docker-entrypoint..." 33 seconds ago Up 32 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp   my-nginx
ubuntu@ip-172-31-19-184:~$
```

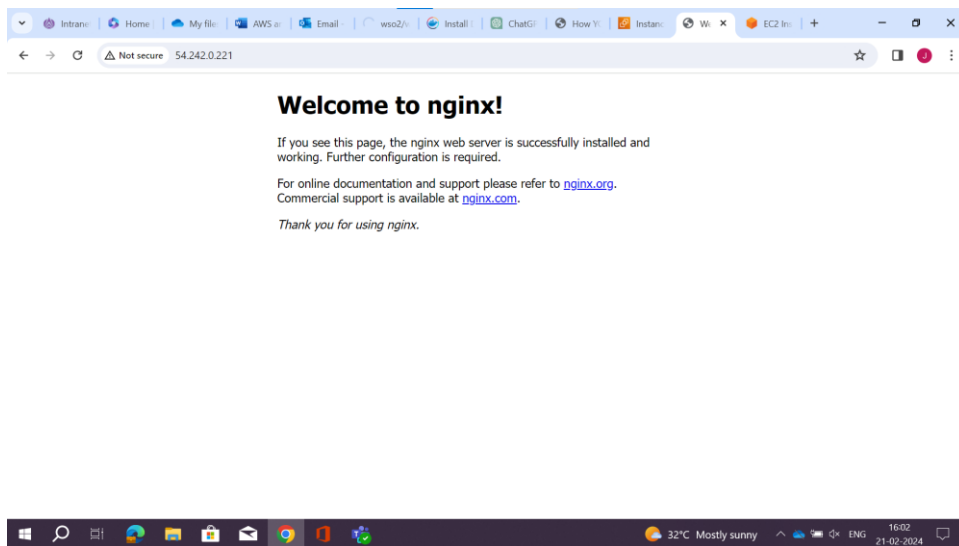
i-08323b78cfec58fc7 (Docker\_Server)

PublicIPs: 54.242.0.221 PrivateIPs: 172.31.19.184

Step 11: Now you can go to the port 80 and check your nginx server is running.

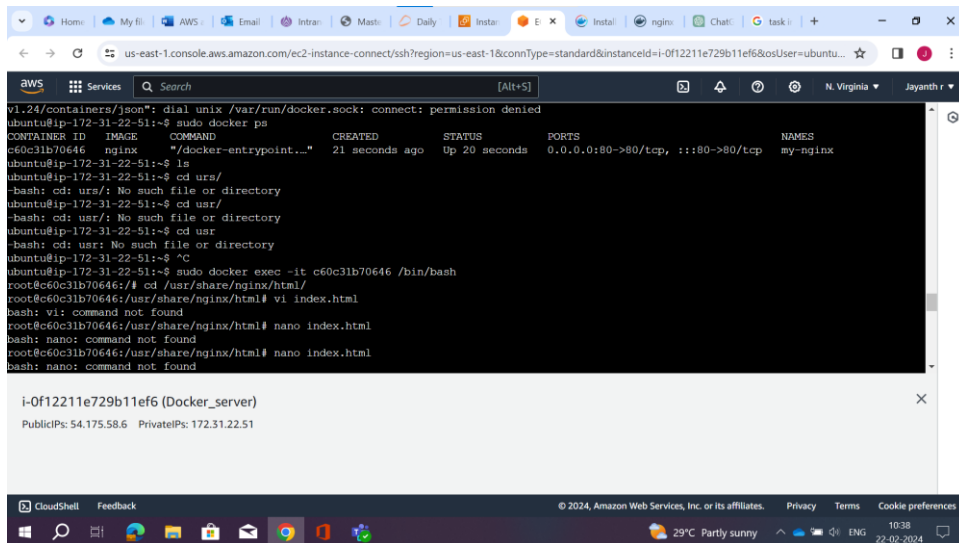
docker pull nginx

docker run --name my-nginx-container -d -p 80:80 nginx



Step 12: Now let us customize our nginx server which is running.

Enter into the executing container using command “sudo docker exec -it c60c31b70646 /bin/bash”.



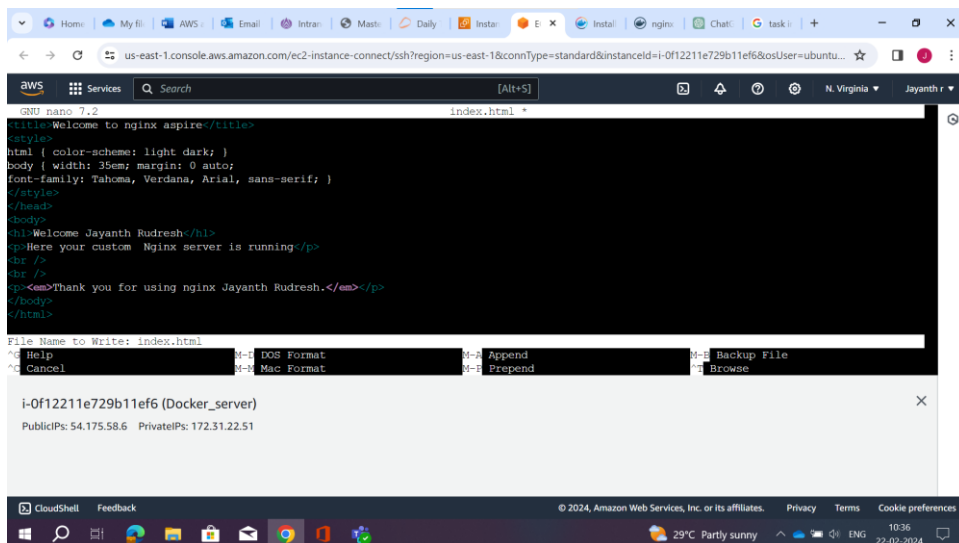
The screenshot shows the AWS CloudShell interface. At the top, there's a navigation bar with 'Home', 'My files', 'AWS', 'Email', 'Intranet', 'Masthead', 'Daily', 'Insta', and a search bar. Below this, the terminal window displays the following commands and output:

```
v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-22-51:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
c60c31b70646   nginx    "/docker-entrypoint..." 21 seconds ago Up 20 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp my-nginx
ubuntu@ip-172-31-22-51:~$ ls
ubuntu@ip-172-31-22-51:~$ cd usr/
-bash: cd: usr/: No such file or directory
ubuntu@ip-172-31-22-51:~$ cd usr/
-bash: cd: usr/: No such file or directory
ubuntu@ip-172-31-22-51:~$ cd usr
-bash: cd: usr: No such file or directory
ubuntu@ip-172-31-22-51:~$ ^C
ubuntu@ip-172-31-22-51:~$ sudo docker exec -it c60c31b70646 /bin/bash
root@c60c31b70646:/# cd /usr/share/nginx/html/
root@c60c31b70646:/usr/share/nginx/html# vi index.html
bash: vi: command not found
root@c60c31b70646:/usr/share/nginx/html# nano index.html
bash: nano: command not found
root@c60c31b70646:/usr/share/nginx/html# nano index.html
bash: nano: command not found
```

Below the terminal, a box shows the instance details: i-0f12211e729b11ef6 (Docker\_server), PublicIPs: 54.175.58.6, PrivateIPs: 172.31.22.51.

Step 13: Now customize index.html file by navigating to the index.html file.

cd /usr/share/nginx/html/ nano index.html

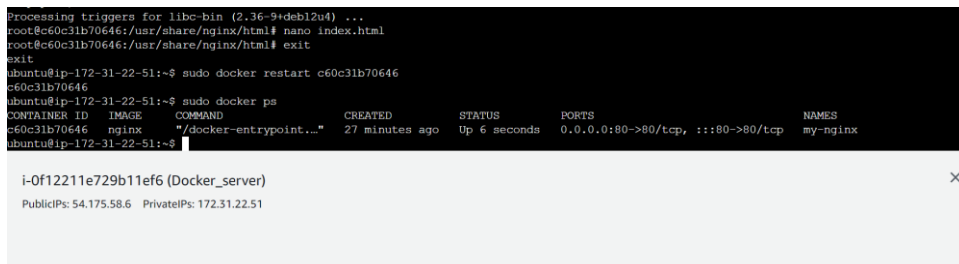


The screenshot shows the AWS CloudShell interface. The terminal window displays the following commands and output:

```
GNU nano 2.2 index.html
<title>Welcome to nginx aspire</title>
<style>
html { color:scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
<head>
<body>
<h1>Welcome Jayanth Rudresh</h1>
<p>Here your custom Nginx server is running</p>
<hr />
<p><em>Thank you for using nginx Jayanth Rudresh.</em></p>
</body>
</html>
File Name to Write: index.html
[?] Help [^] Cancel [^] DOS Format [^] Mac Format [^] Append [^] Backup File
[?] [^] Prepend [^] Browse
```

Below the terminal, a box shows the instance details: i-0f12211e729b11ef6 (Docker\_server), PublicIPs: 54.175.58.6, PrivateIPs: 172.31.22.51.

Step 14: Now restart the running container.

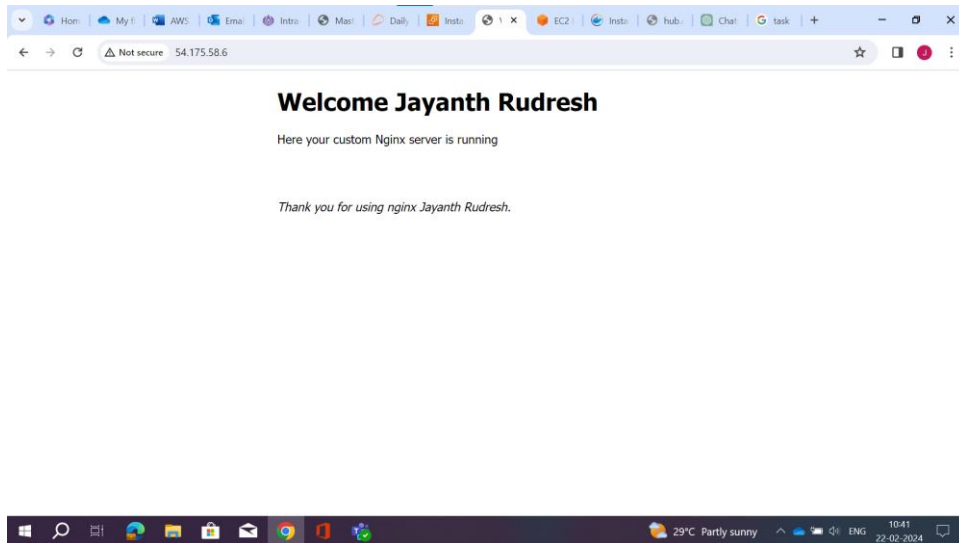


The screenshot shows the AWS CloudShell interface. The terminal window displays the following commands and output:

```
Processing triggers for libe-bin (2.36-9+deb12u4) ...
root@c60c31b70646:/usr/share/nginx/html# nano index.html
root@c60c31b70646:/usr/share/nginx/html# exit
exit
ubuntu@ip-172-31-22-51:~$ sudo docker restart c60c31b70646
c60c31b70646
ubuntu@ip-172-31-22-51:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
c60c31b70646   nginx    "/docker-entrypoint..." 27 minutes ago Up 6 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp my-nginx
ubuntu@ip-172-31-22-51:~$
```

Below the terminal, a box shows the instance details: i-0f12211e729b11ef6 (Docker\_server), PublicIPs: 54.175.58.6, PrivateIPs: 172.31.22.51.

Step 14: Now you can see the customized html file running in server.



Step 13: Now delete the container and image.

“Docker rmi” for image.

“Docker rm” for container.

