# ***How to deploy Medial Billing App on docker***

**App overview:**

Medical Billing Application - Docker Deployment

Our Medical Billing Application is designed to efficiently manage and retrieve medical billing records. To ensure seamless deployment, we have containerized the application using Docker.

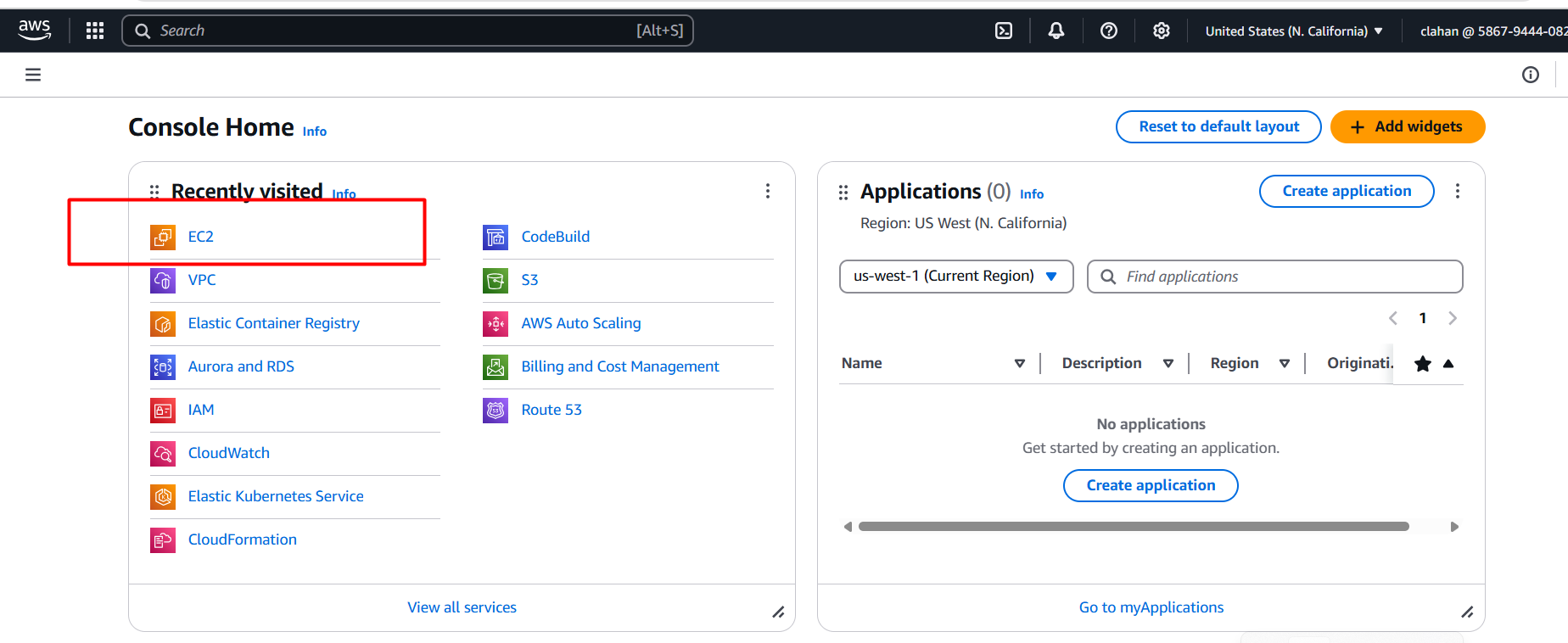
Deployment Overview

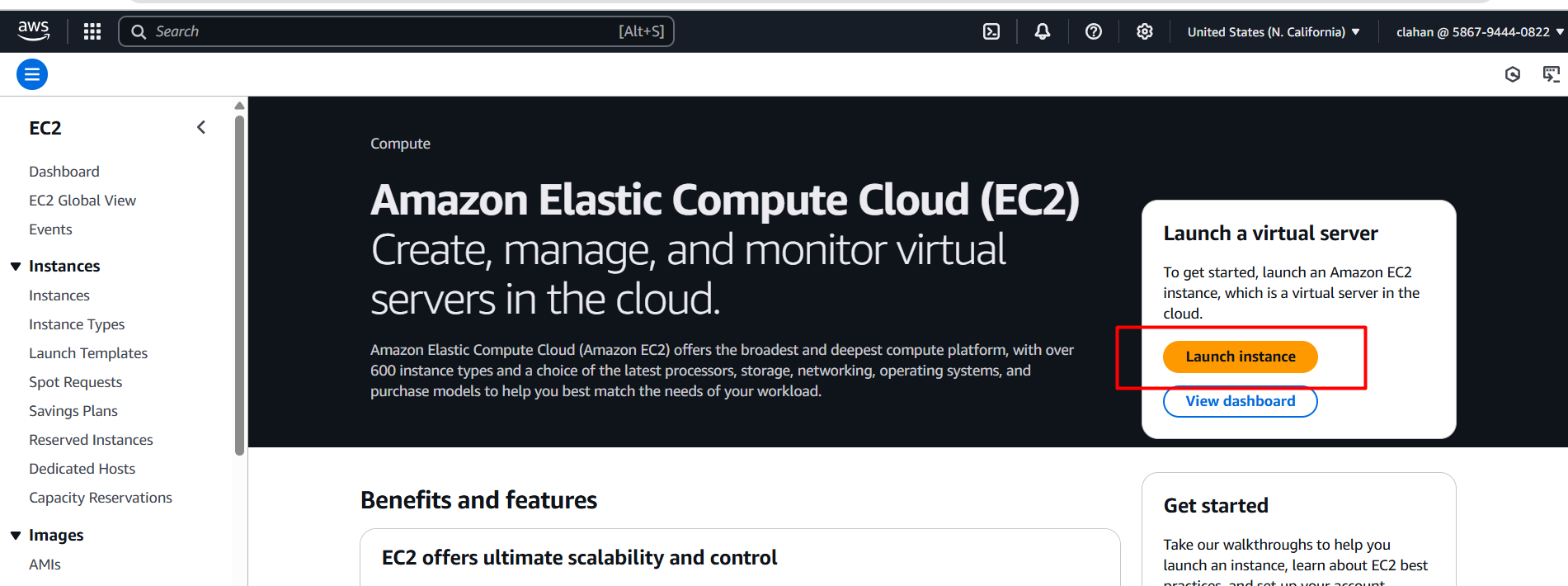
We have implemented a multi-container setup using Docker Compose, which includes:

* Frontend: The user interface for interacting with medical billing data.
* MongoDB Database: A NoSQL database for securely storing medical billing records.

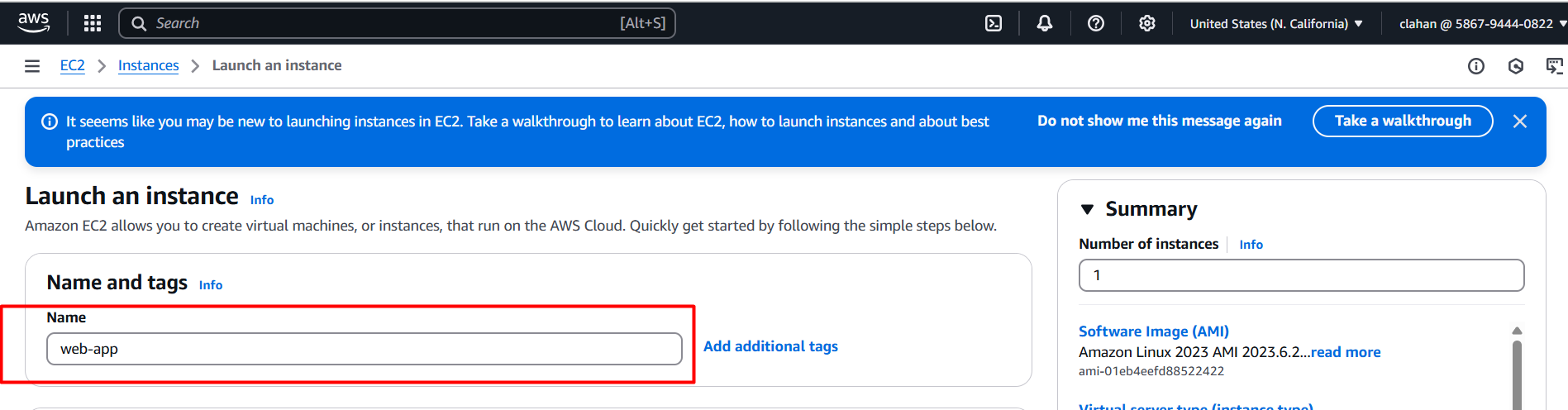
Dockerization Strategy

* We have written a Dockerfile to containerize the application.
* A Docker Compose file has been created to manage multiple containers efficiently.
* The application will be deployed in Docker containers, ensuring scalability and ease of management.
* Firstly, we need to launch one ec2 instance below are the steps 👇
* Loign to aws account and go to ec2 service dashboard

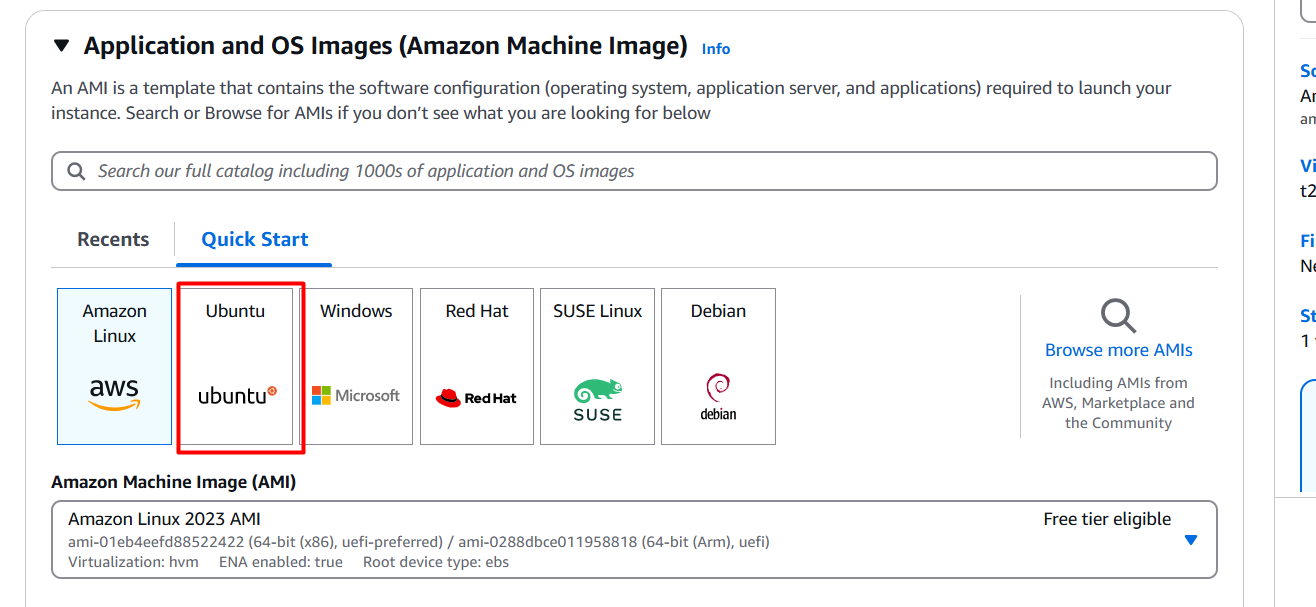




* Give a name for your ec2 instance as below image



* Choose operating system ubuntu (in this case I am choosing ubuntu based on your requitement you can change the OS)



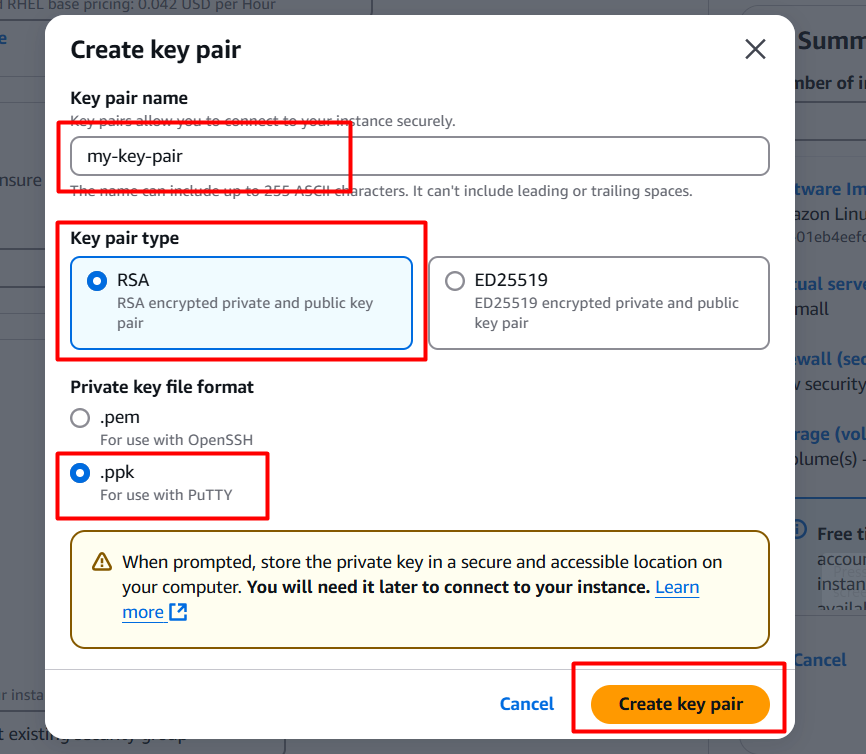
* Next choose instance type here I am choosing **t2.medium** for deploying simple static web app (but based on app requirement instance type will be changed)



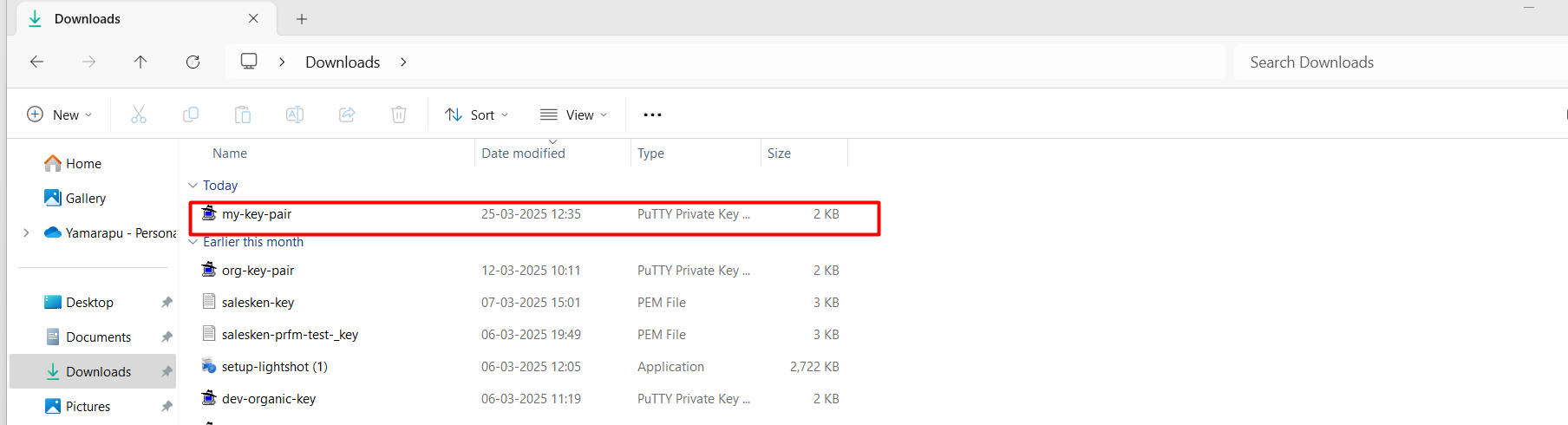
* Next, we need to key pair for connecting ec2 instance.
* How to create key pair follow below steps 👇
* Click on create new key pair



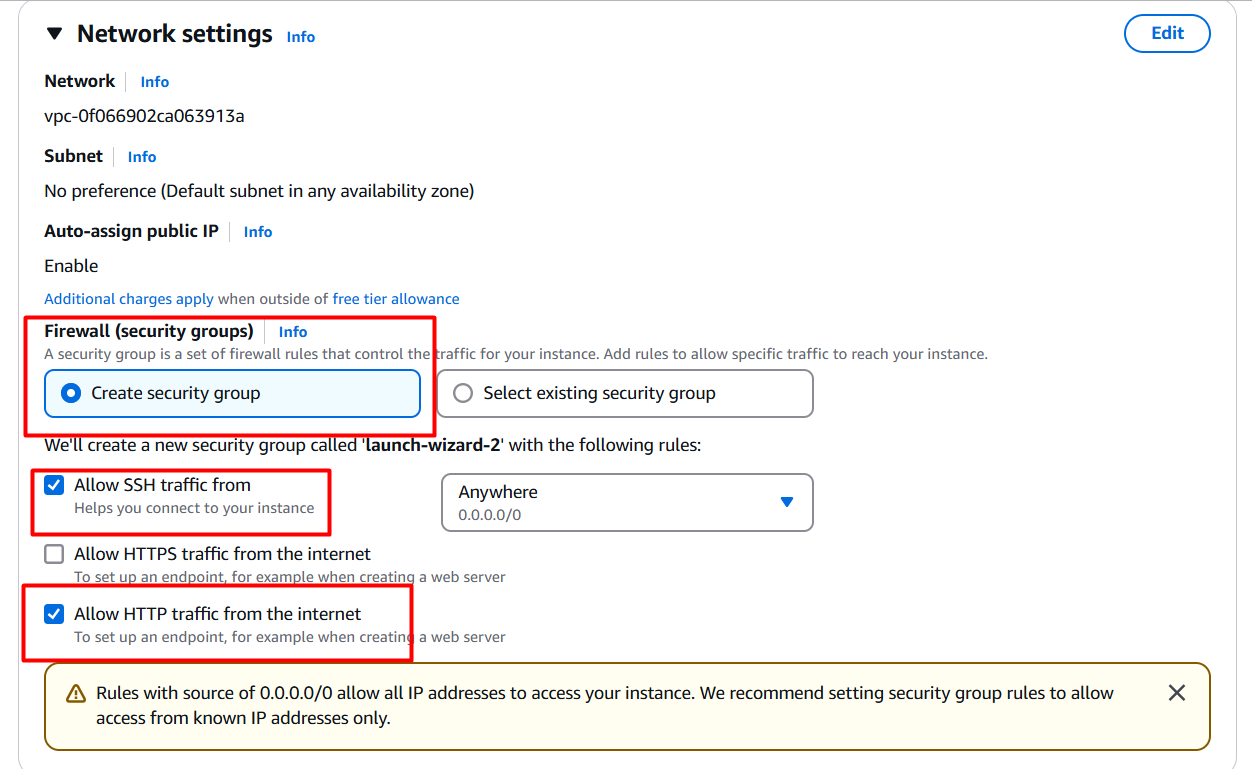
* Provide a name for key pair, select RSA, choose .ppk and click on create key pair 👇



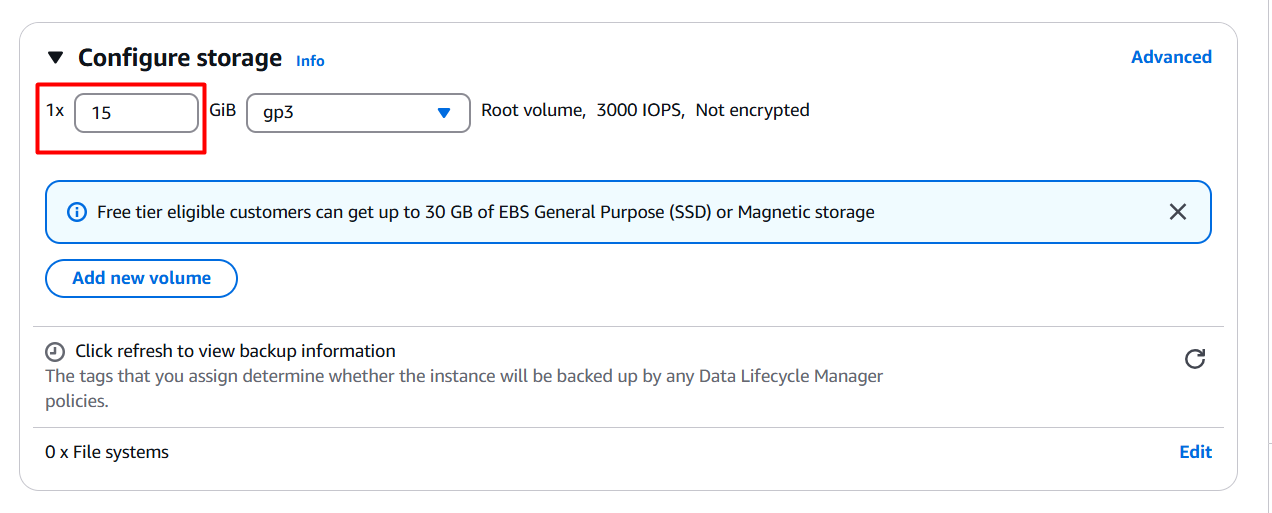
* It will be downloaded into your local server and go to file manager 🡪 downloads there you can see your new downloaded key pair



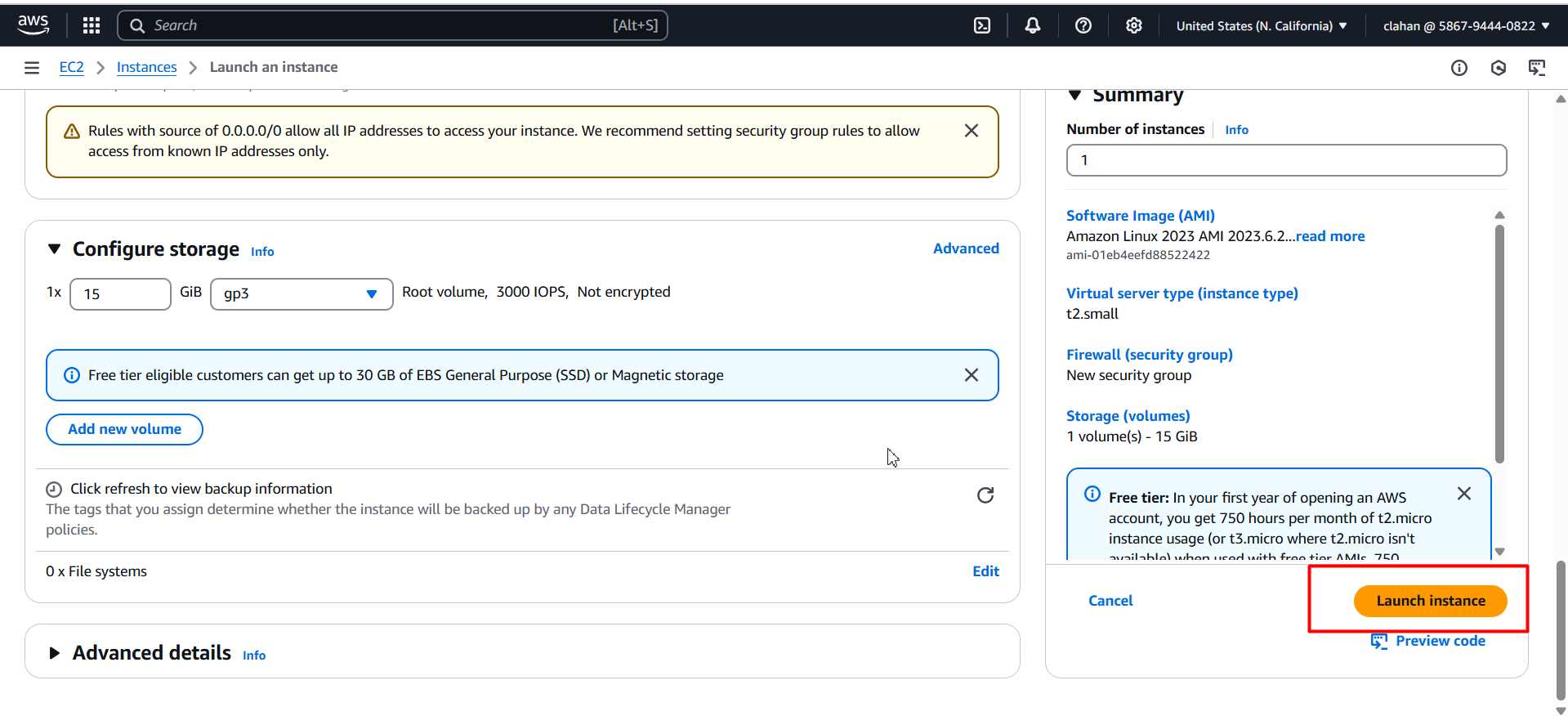
* Next in networking section we need to create one security group for enable port numbers 👇

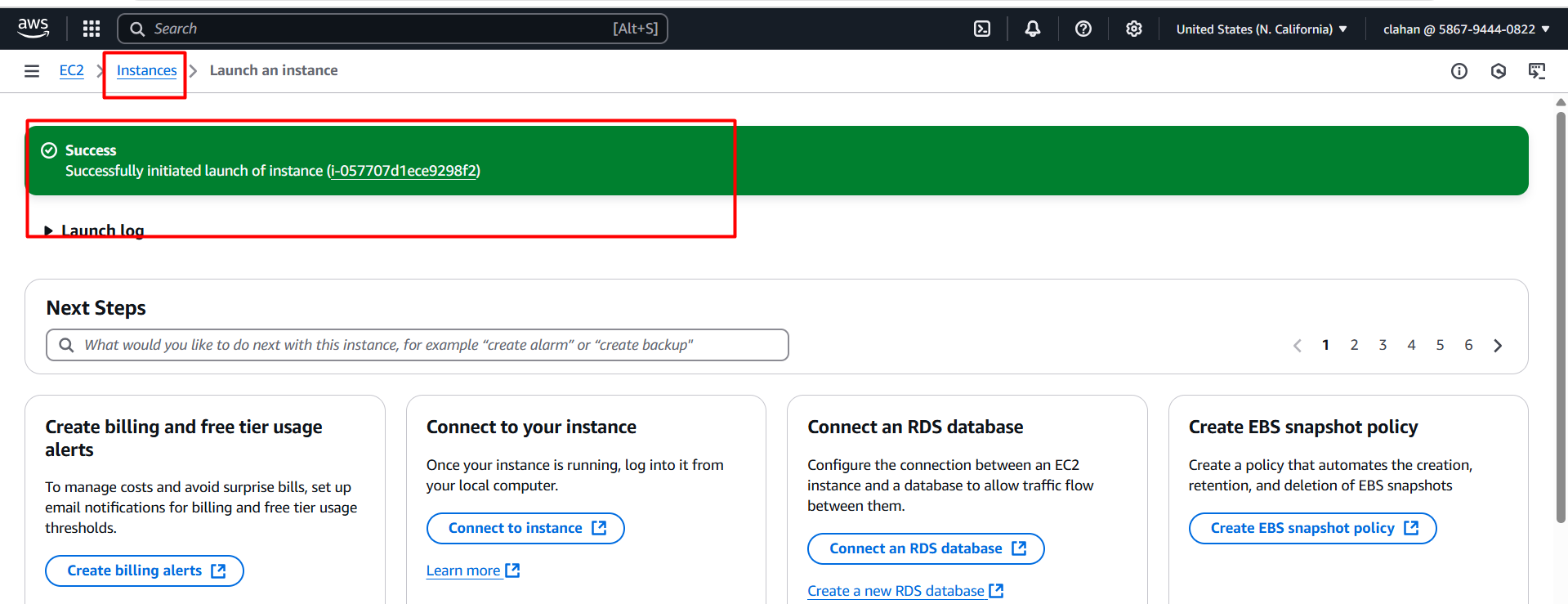


* Next we need to configure storage based upon app requirement , here I am choosing **20gb** for static web app 👇

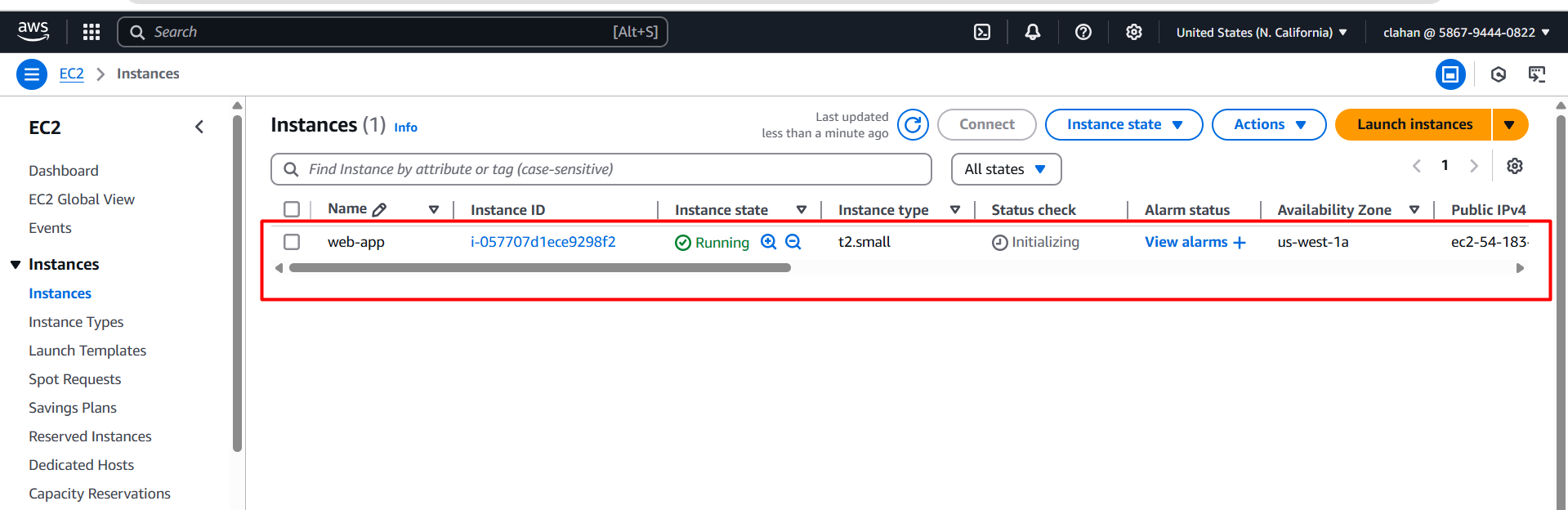


* Finally, we need to launch the ec2 instance 👇

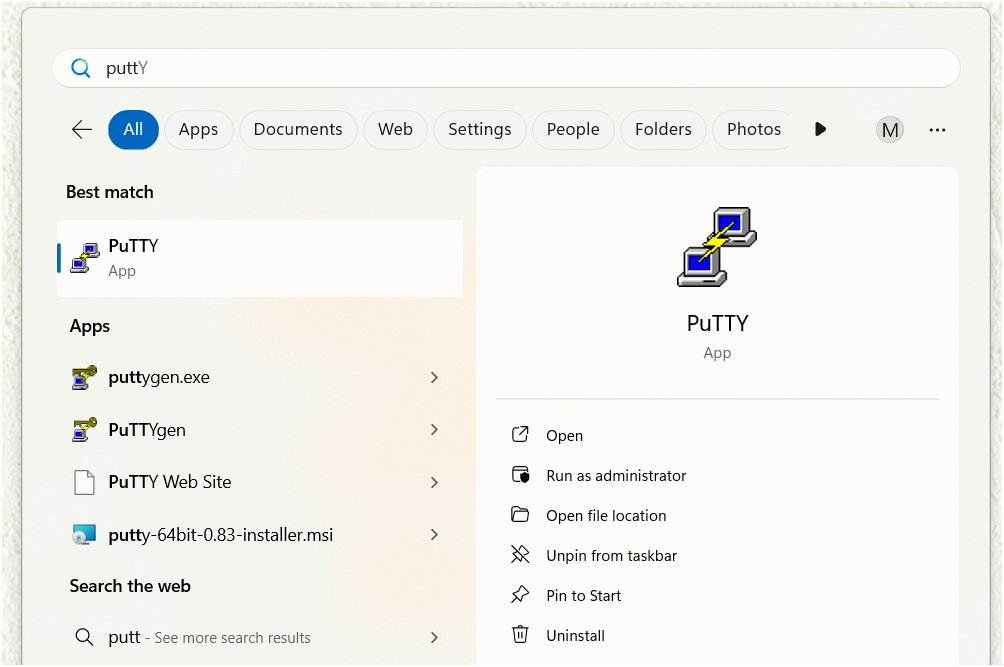




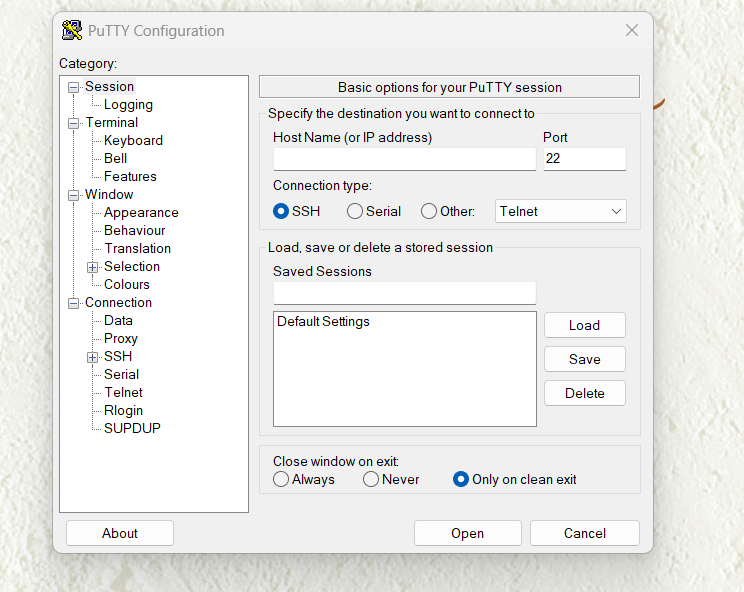
* Click on instances you can see your new created instance ☝️



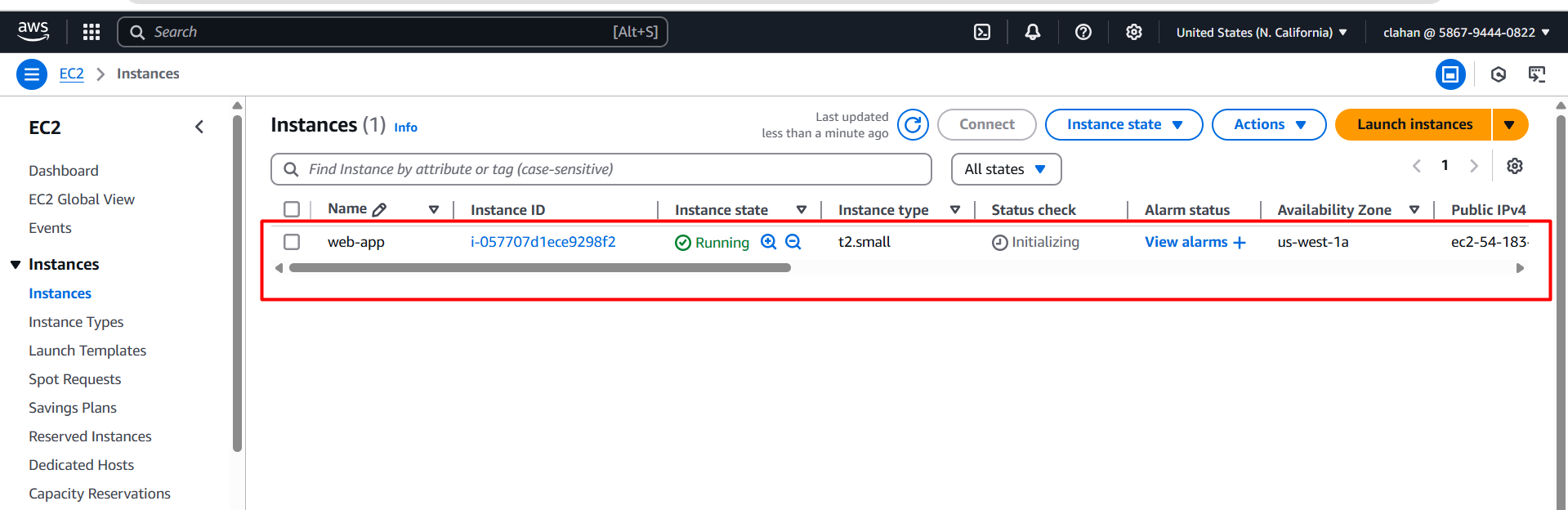
* open putty server on your local machine 👇



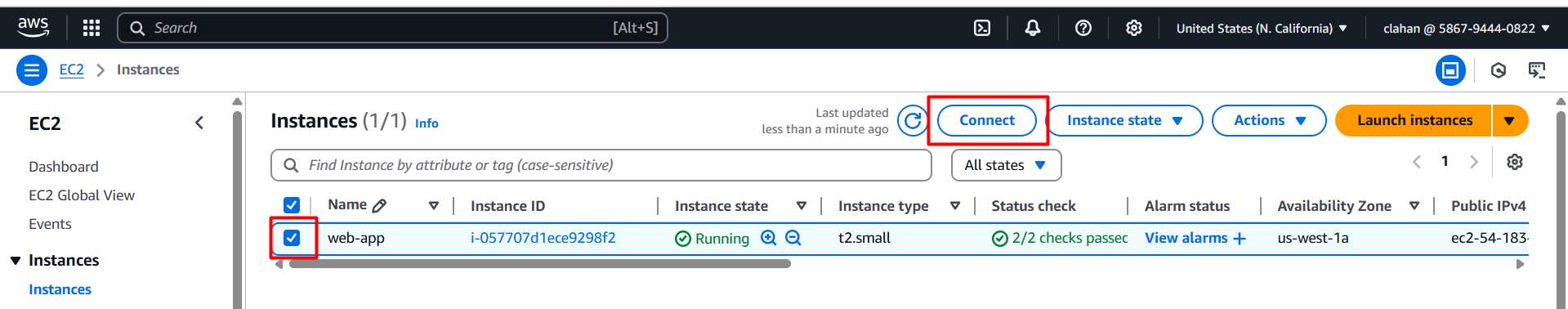
* It will open like this 👇

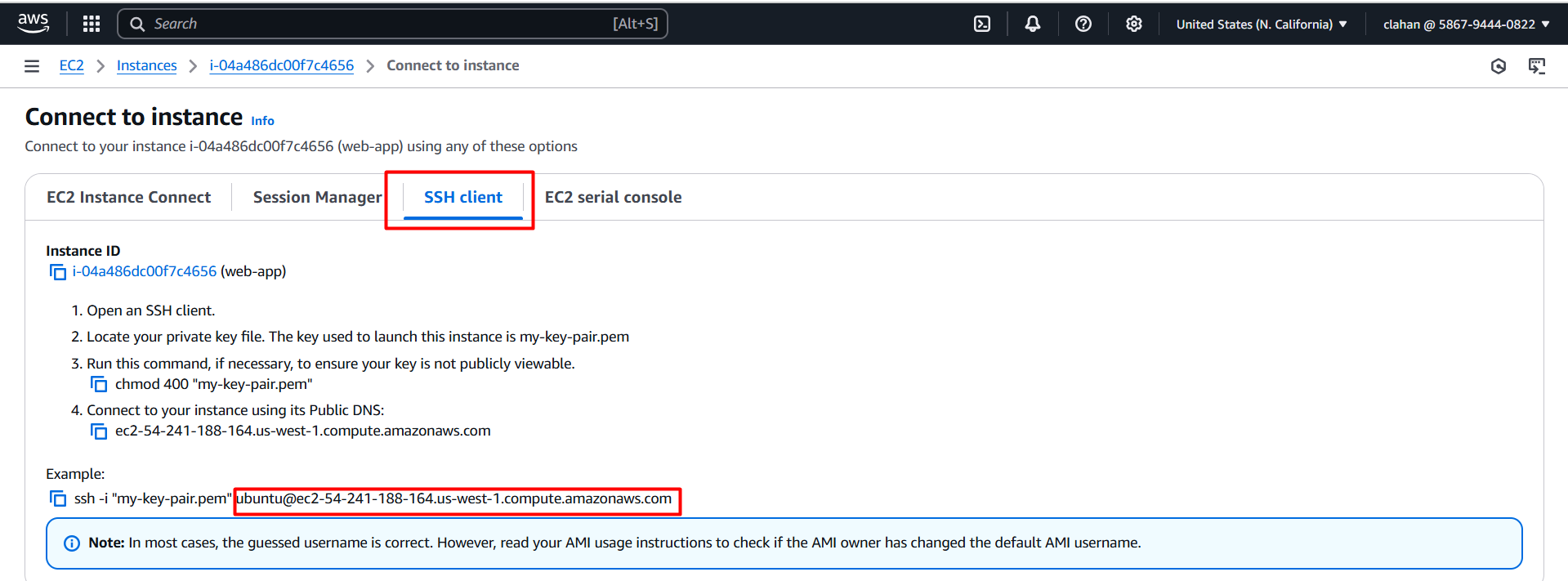


* Now go to your aws console and go to ec2 instance 👇

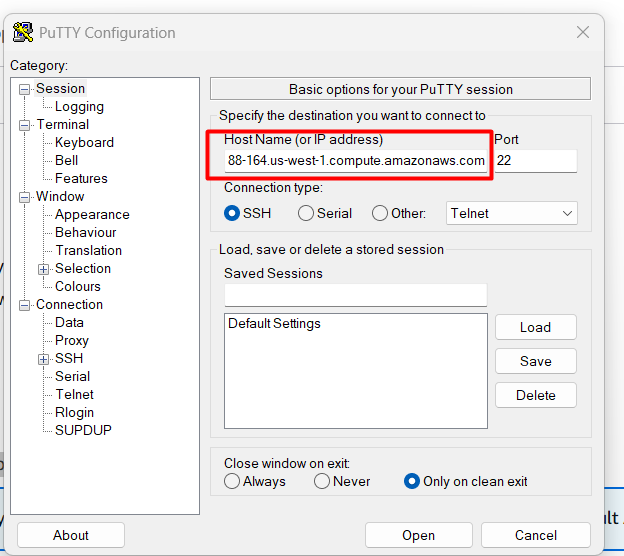


* Choose your instance and click on connect

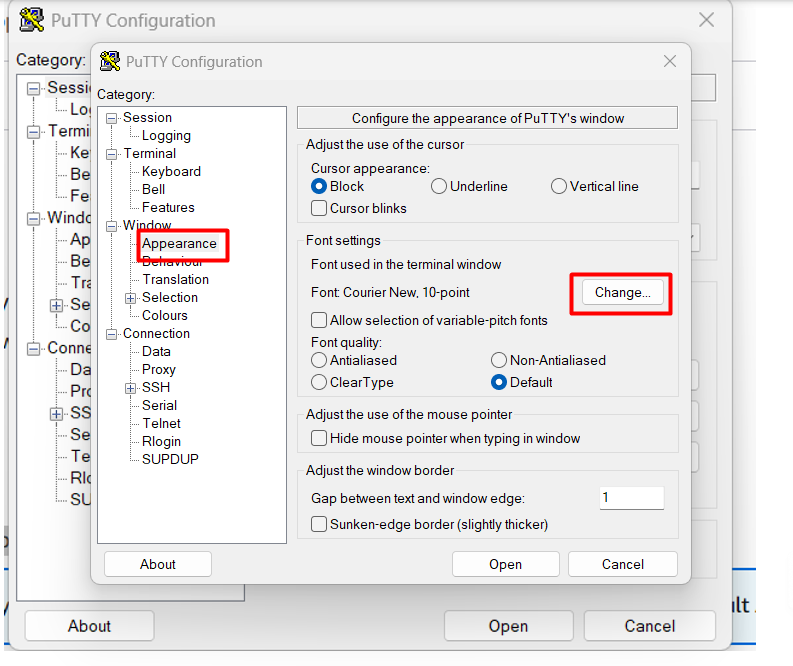


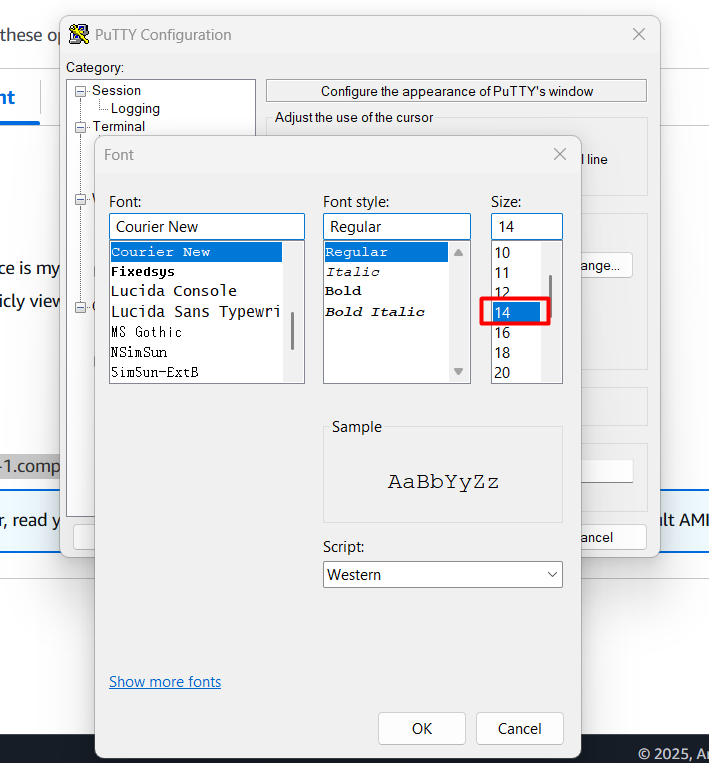


* Go to putty server enter copied host name as mentioned above image (second marked box)

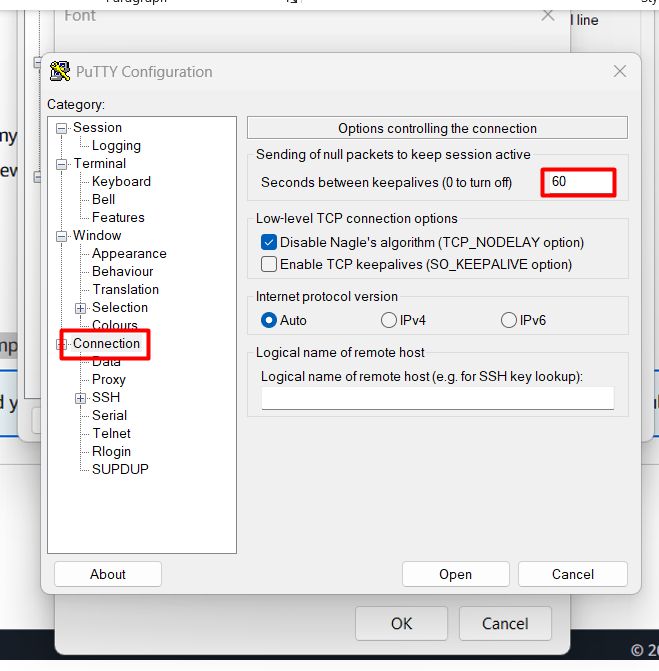


* Click on appearance and change for changing font size

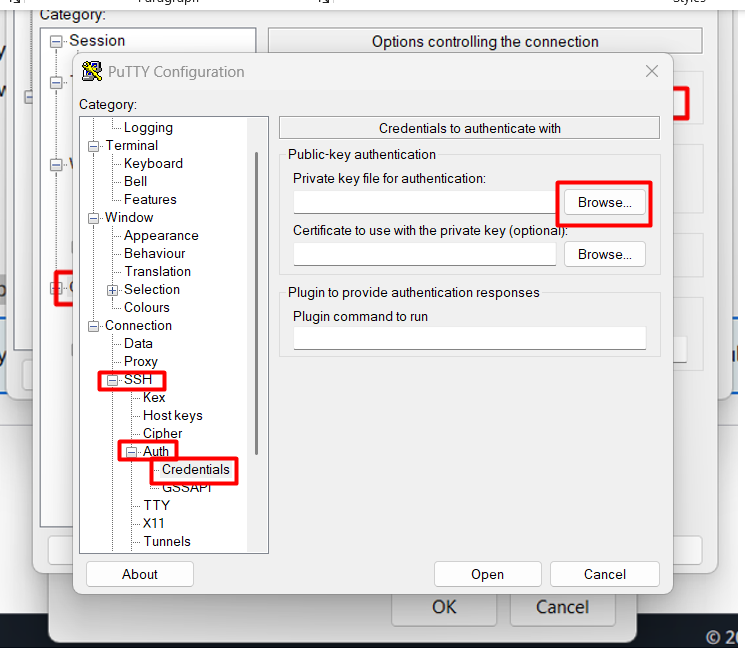




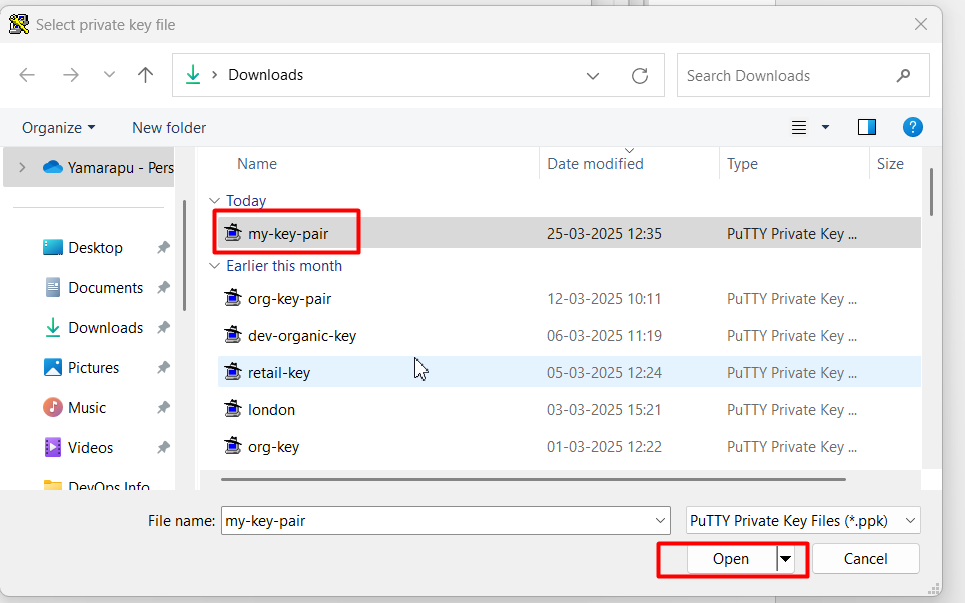
* Next click on connection and give 60 sec for timed out



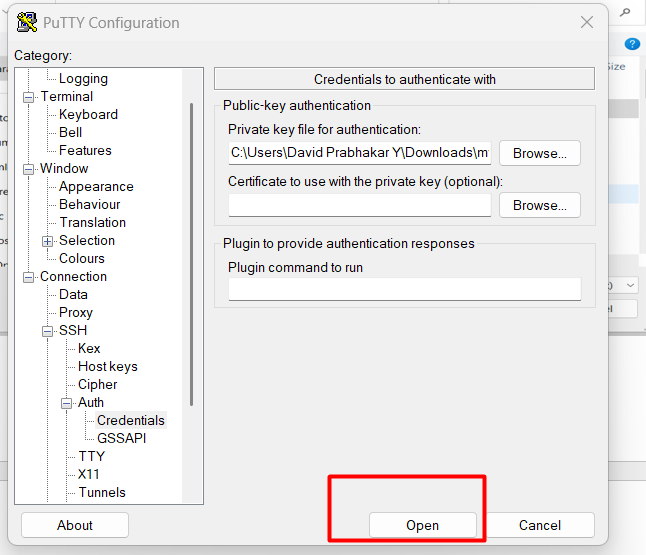
* Click on ssh 🡪 auth 🡪 credentials 🡪 browse



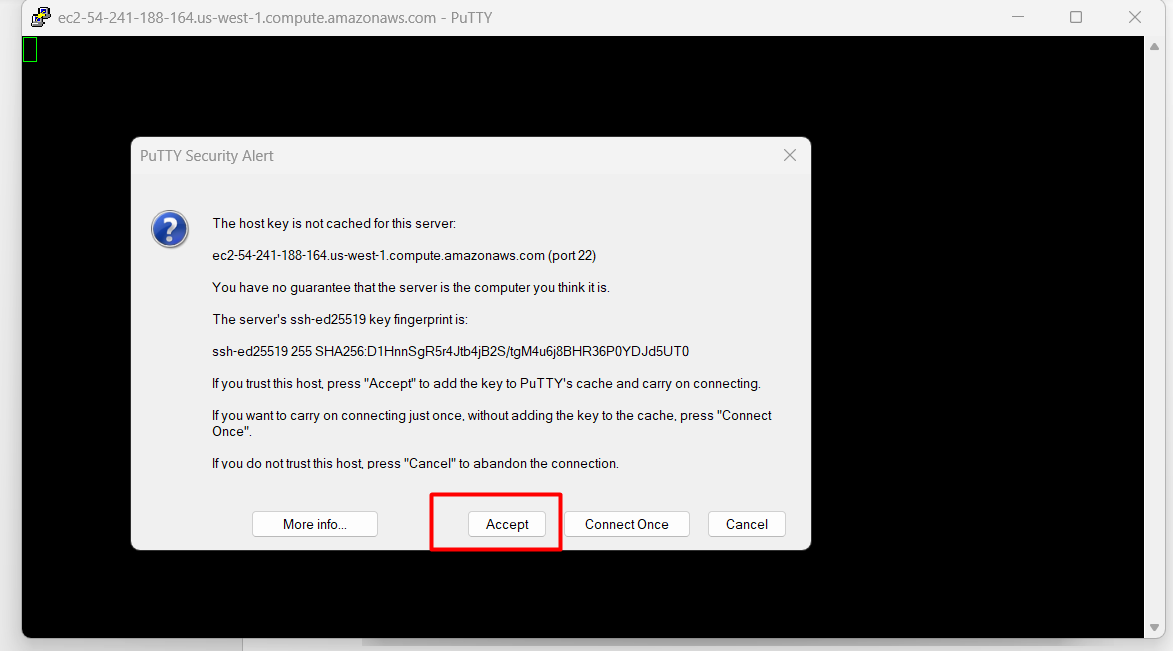
* After it will redirect into your file manager there choose your downloaded key pair



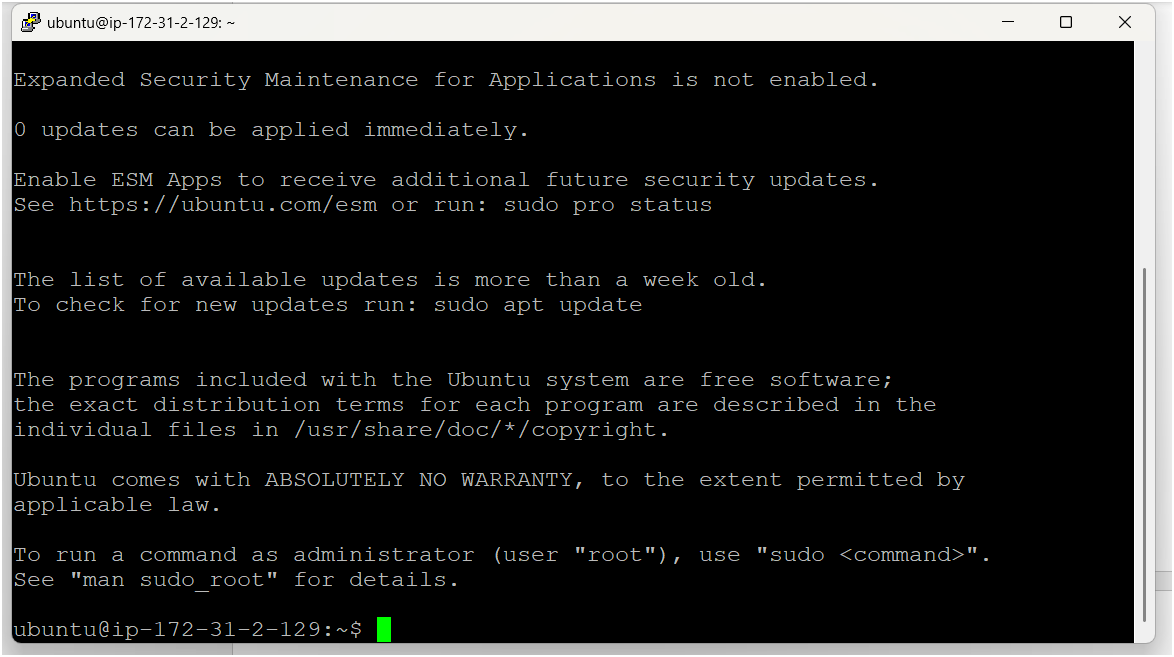
* Next again click on open



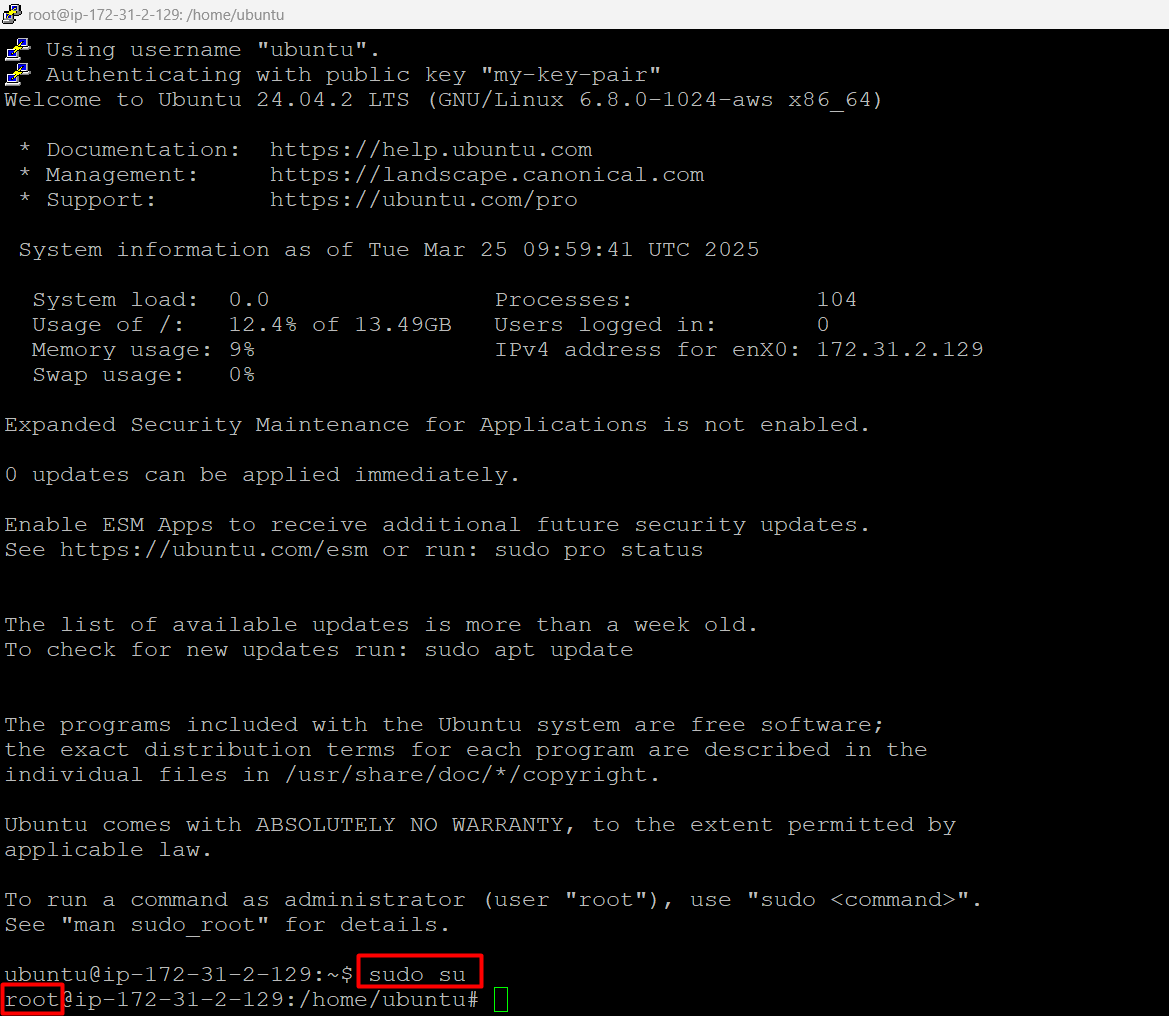
* Click on acceptance



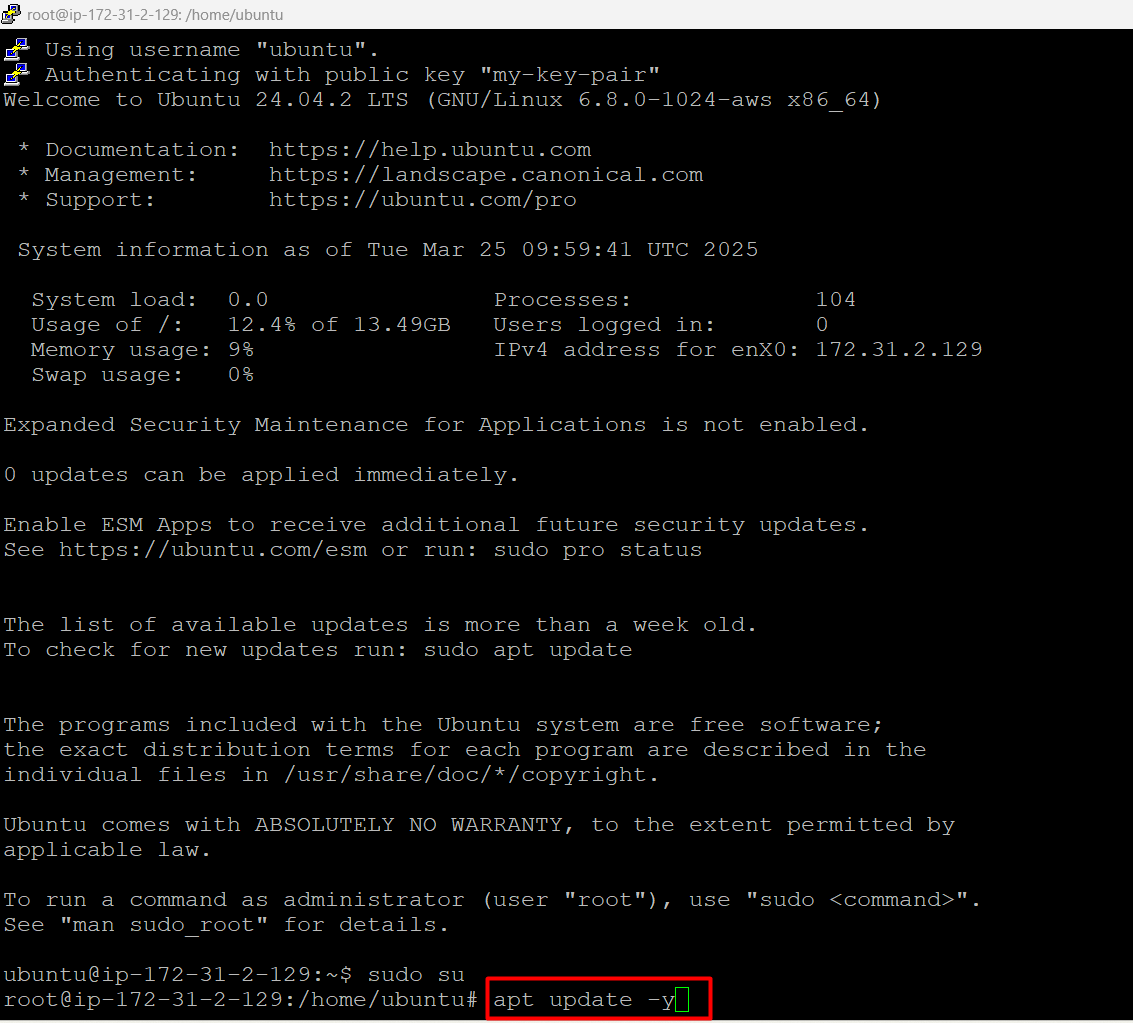
* Now you will connect to your server



* After logged to server firstly you need to become as a root user



* Then update your server by using apt update -y

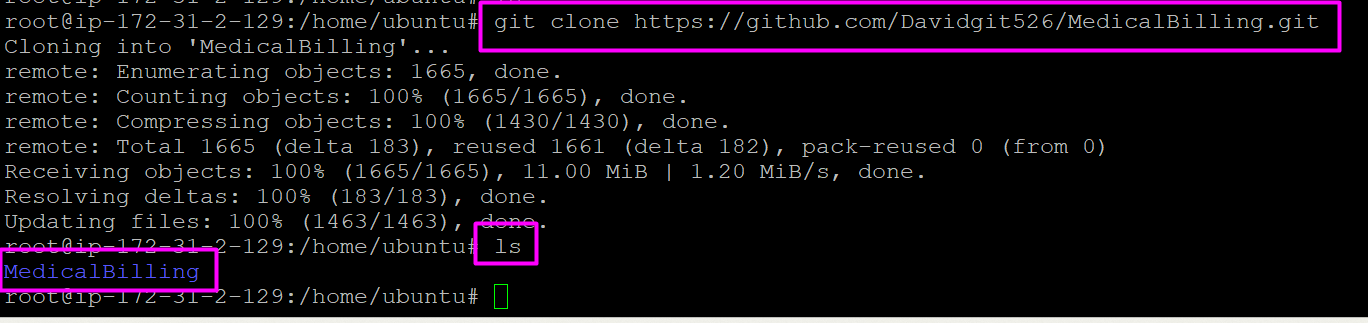


* Now we have to clone the repository into our server

<https://github.com/Davidgit526/MedicalBilling.git>

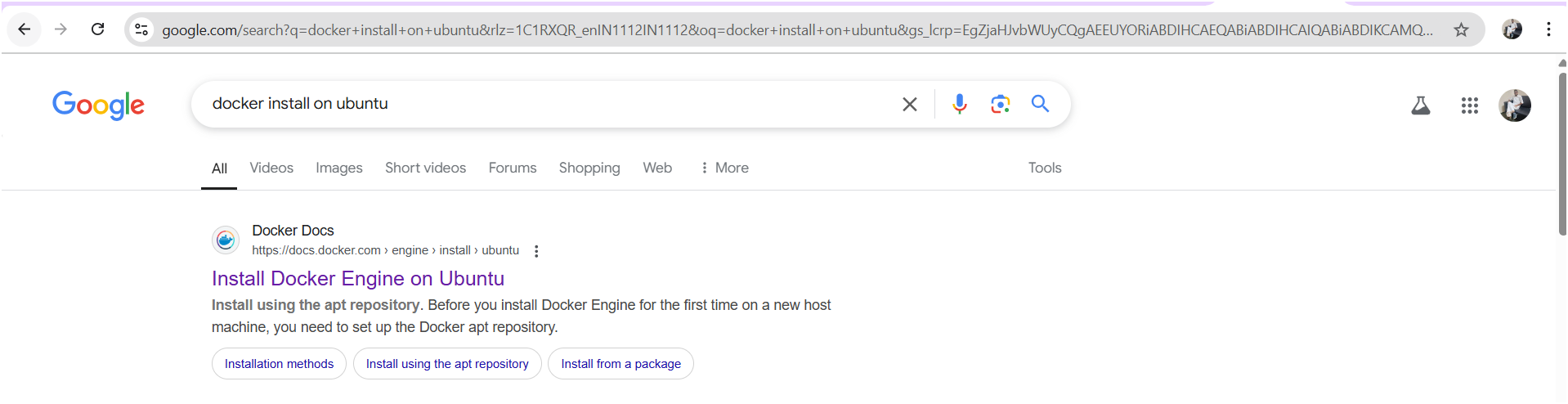
using below command

git clone <https://github.com/Davidgit526/MedicalBilling.git>

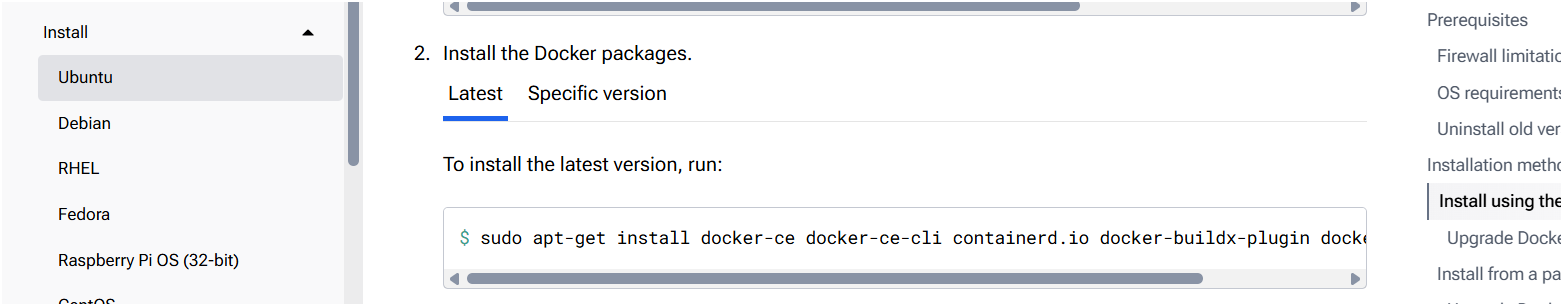




* Now install docker on ubuntu os/instance using below commands







**Set up Docker's apt repository.**

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

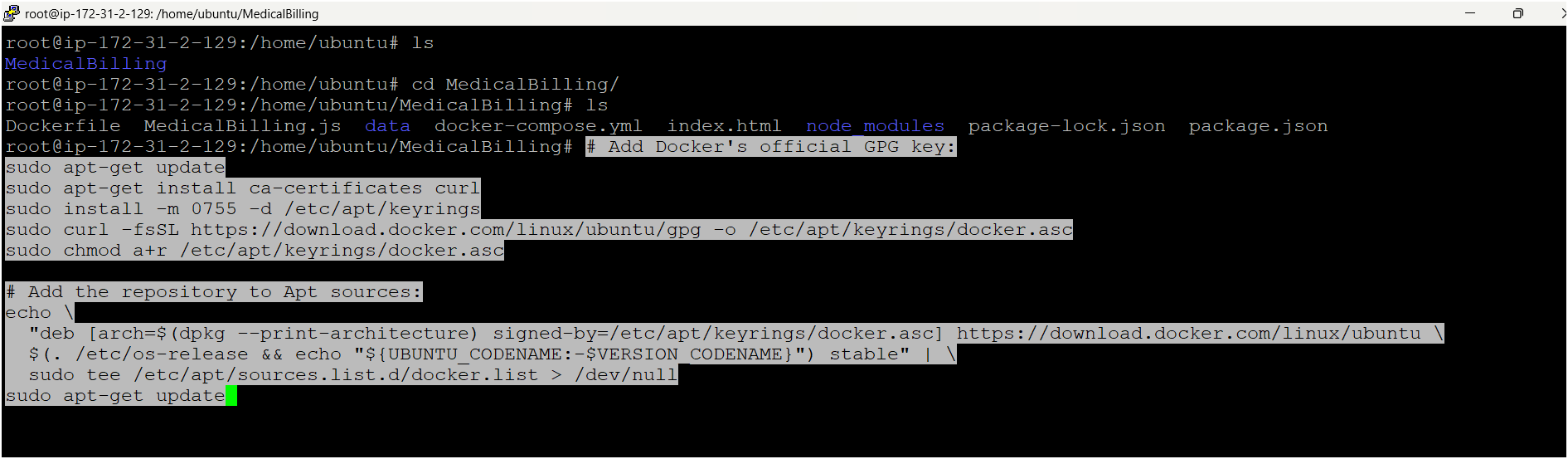
echo \

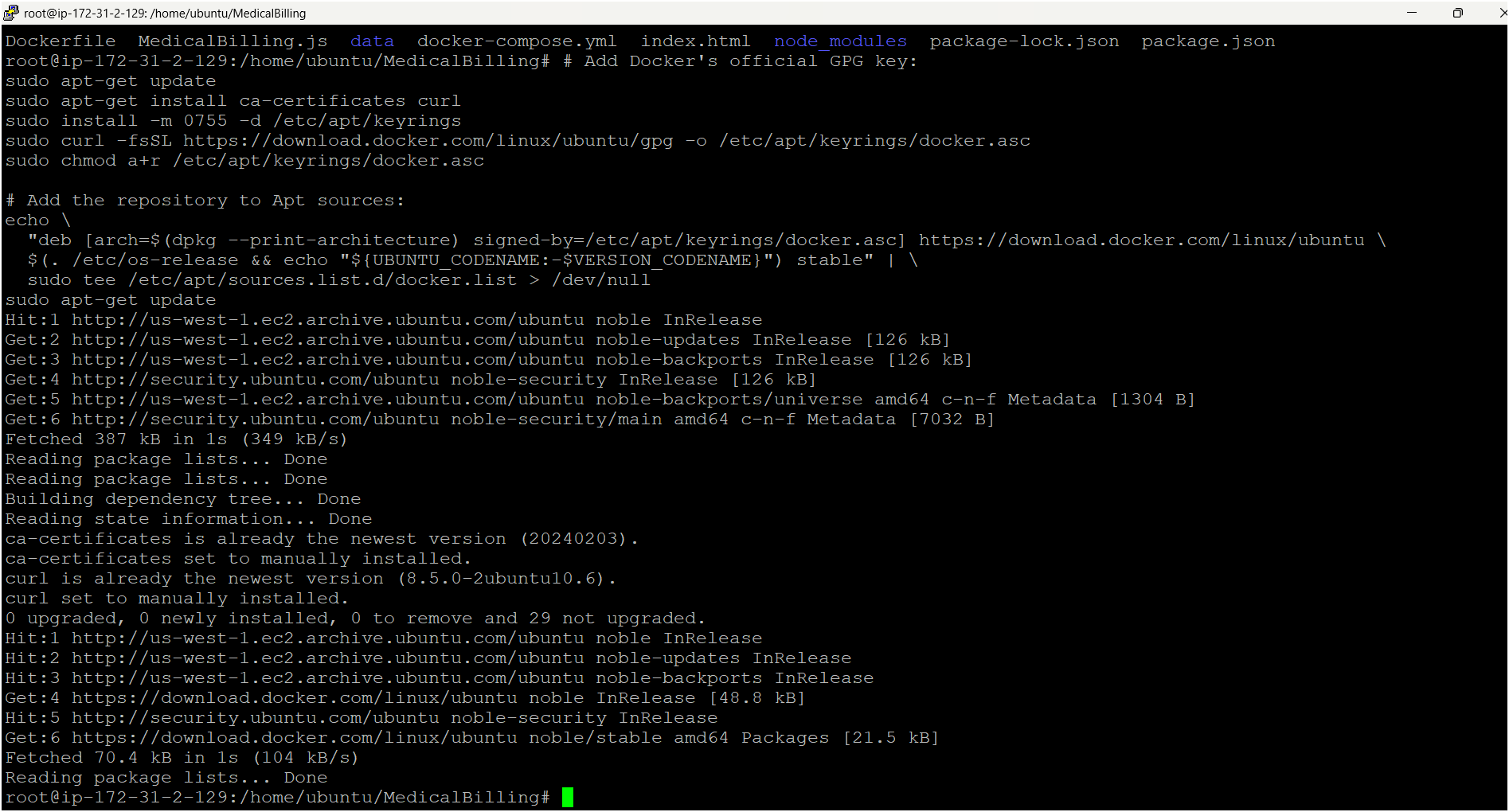
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

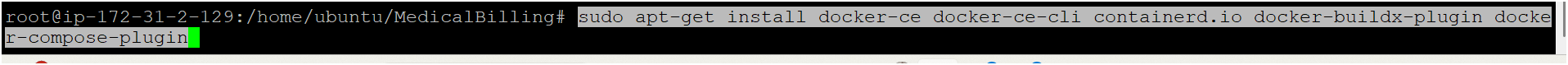
sudo apt-get update

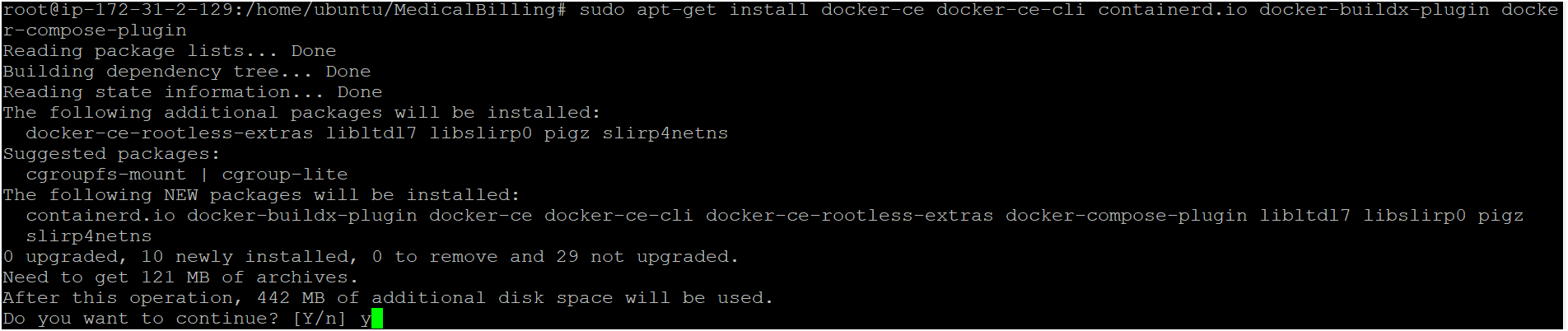


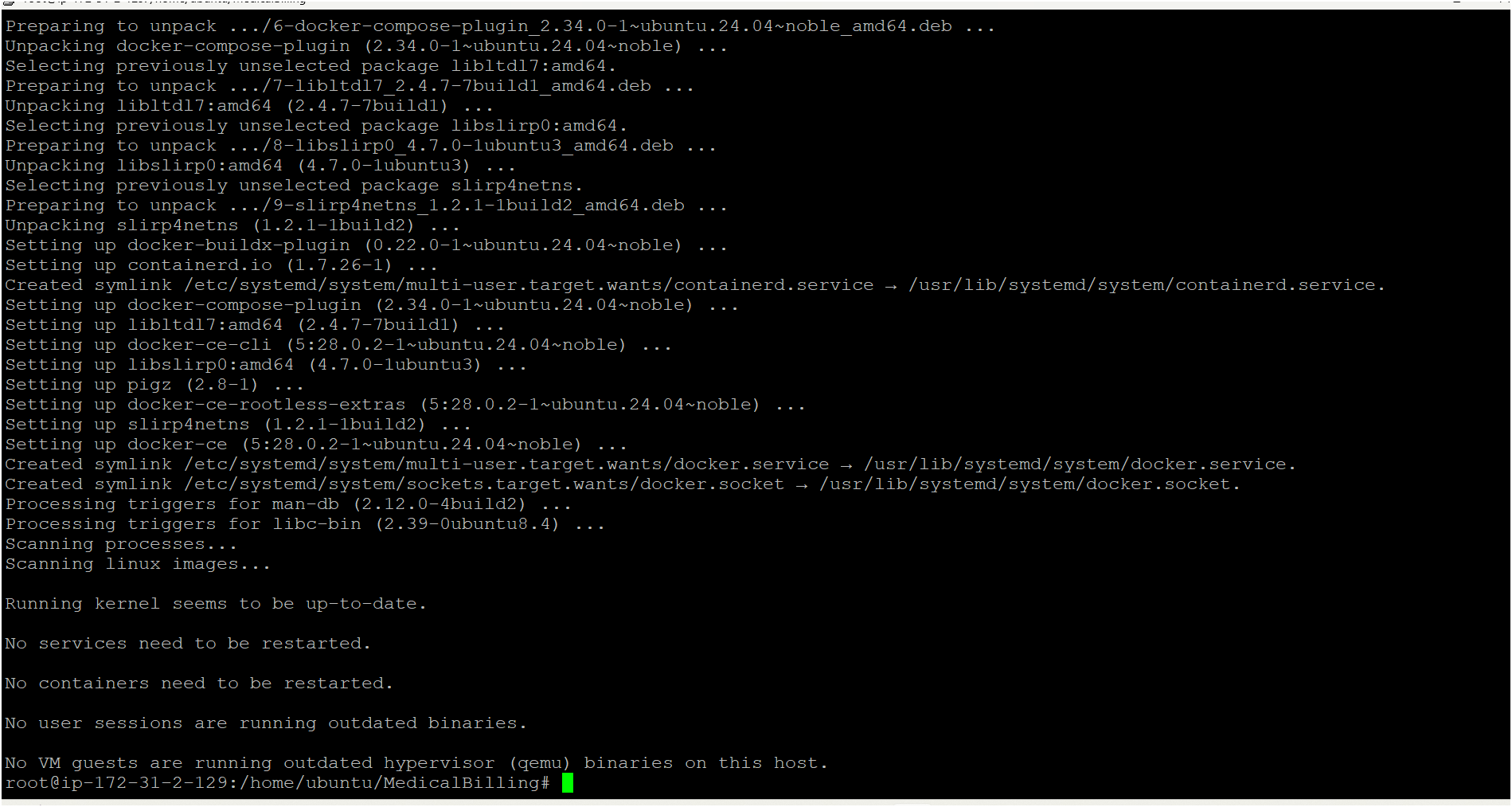


**Install the Docker packages.**

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

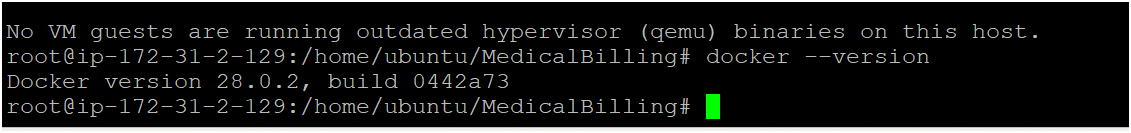






* Now check docker version whether its installed or not using below command

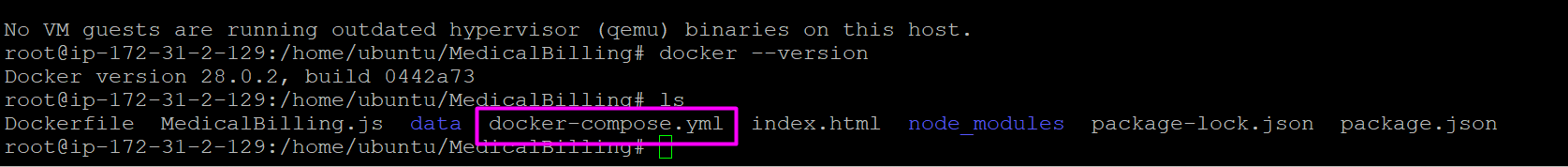
docker –verion



* If version is not visible manually restart docker using

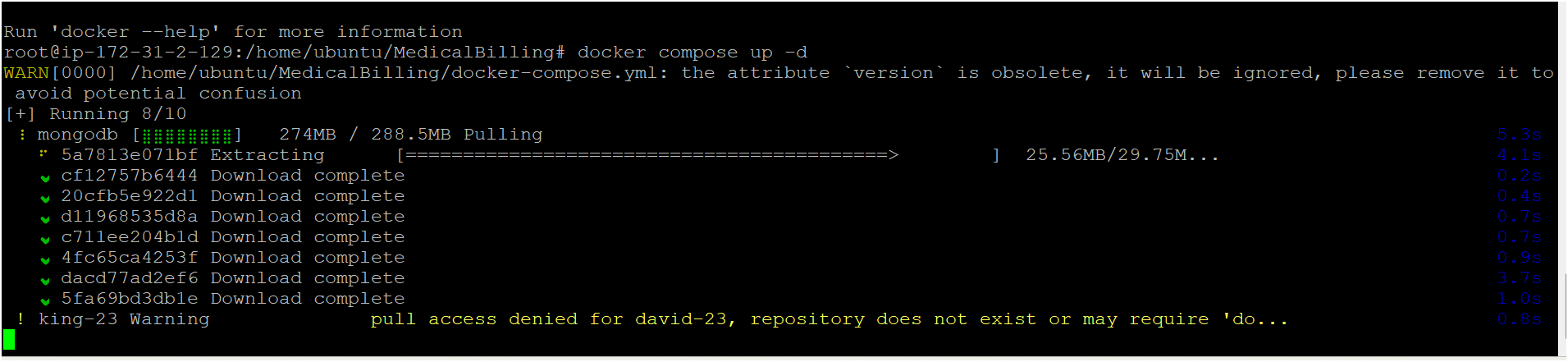
service docker restart

* Now deploy the app by running docker compose file 👇

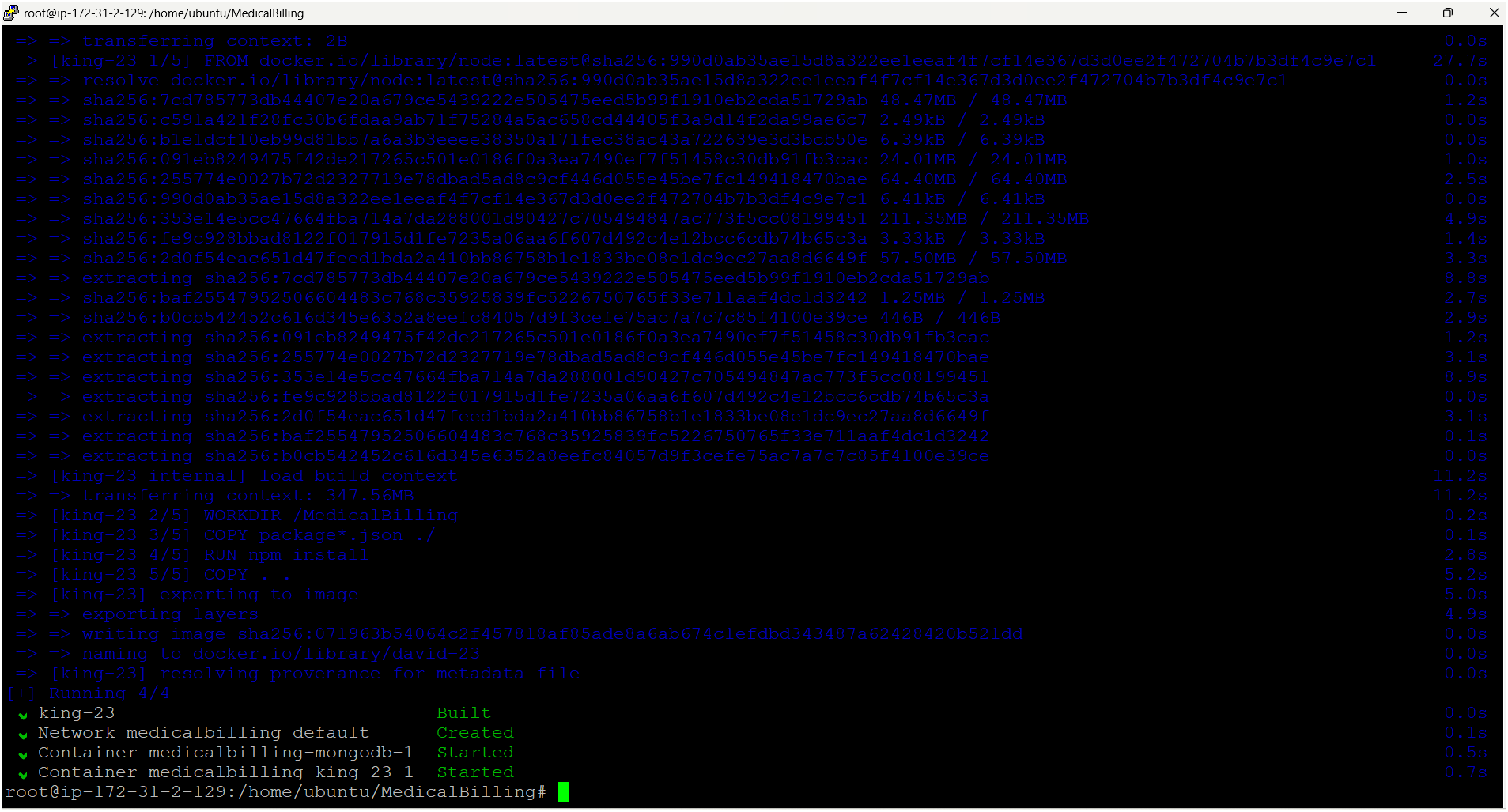


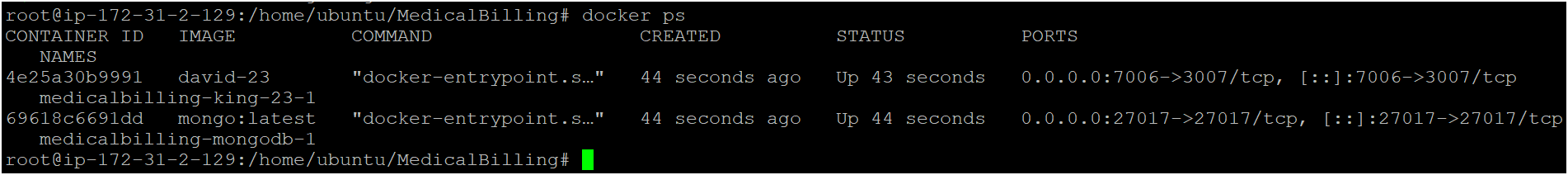
use this command for running compose yaml file

docker compose up -d

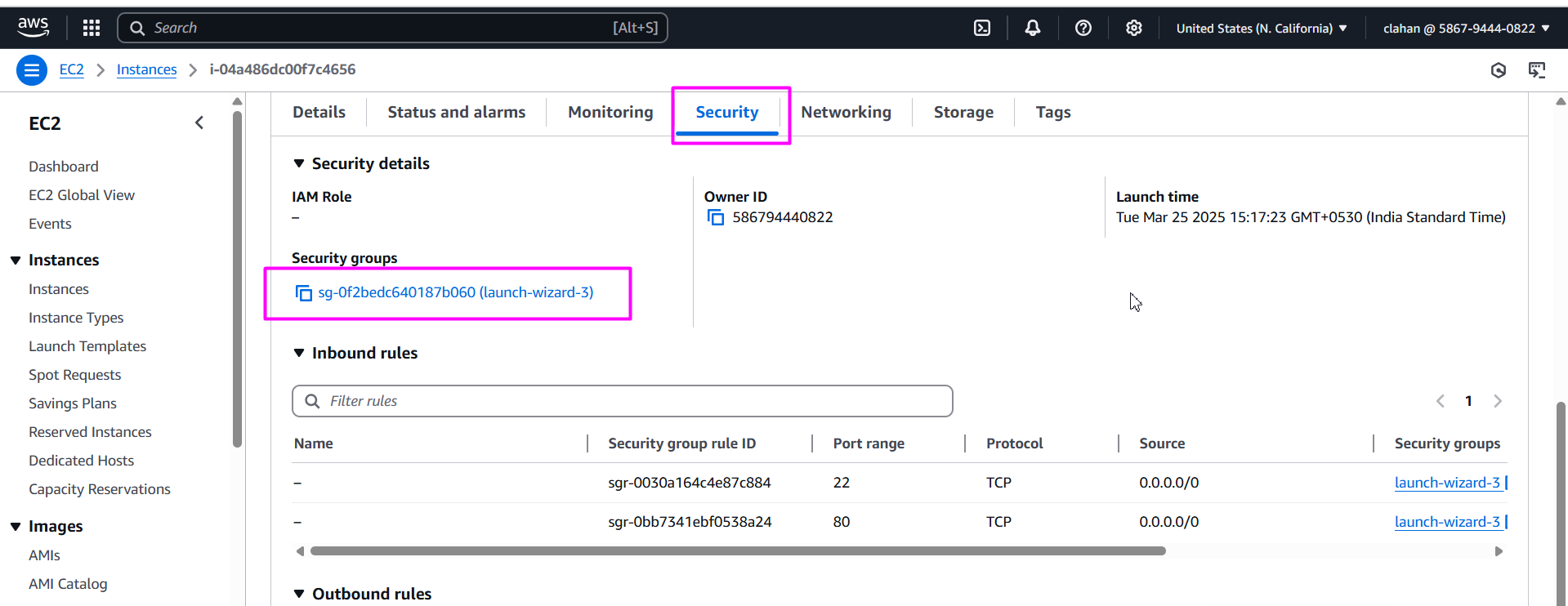


* Containers are created 👇

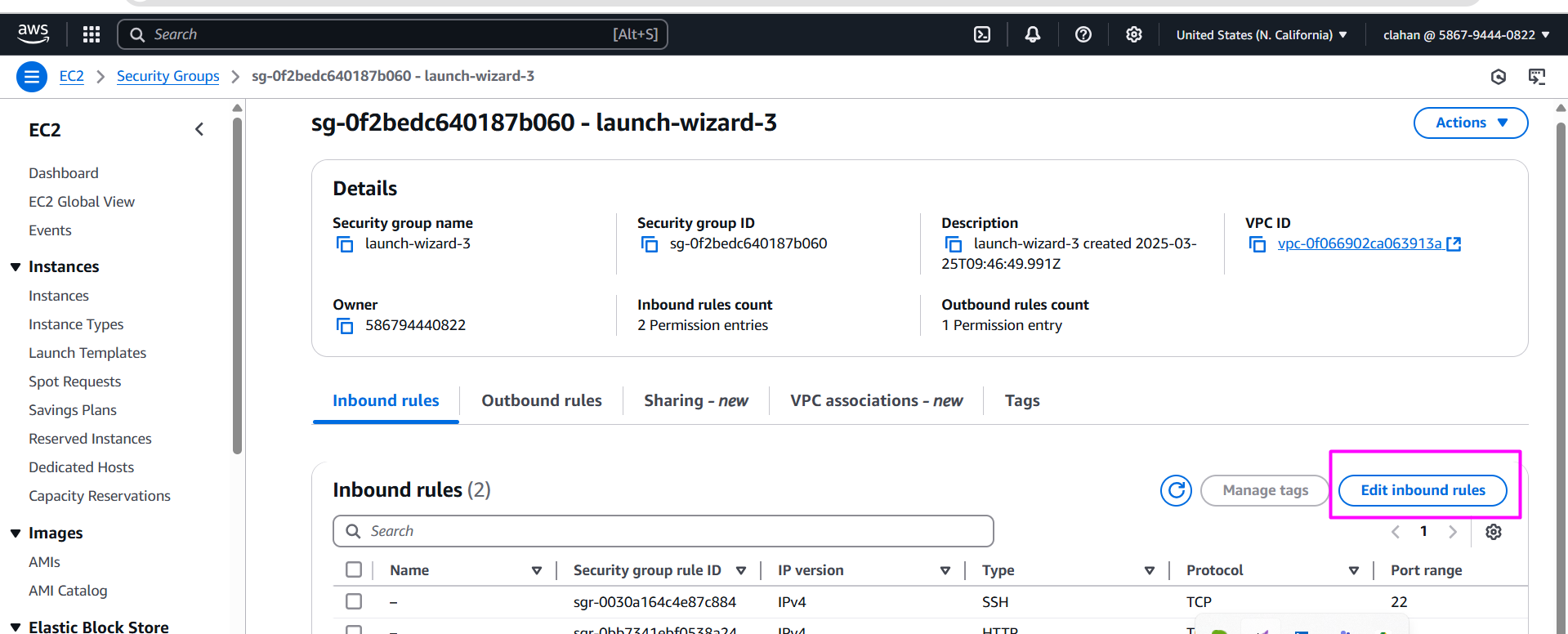




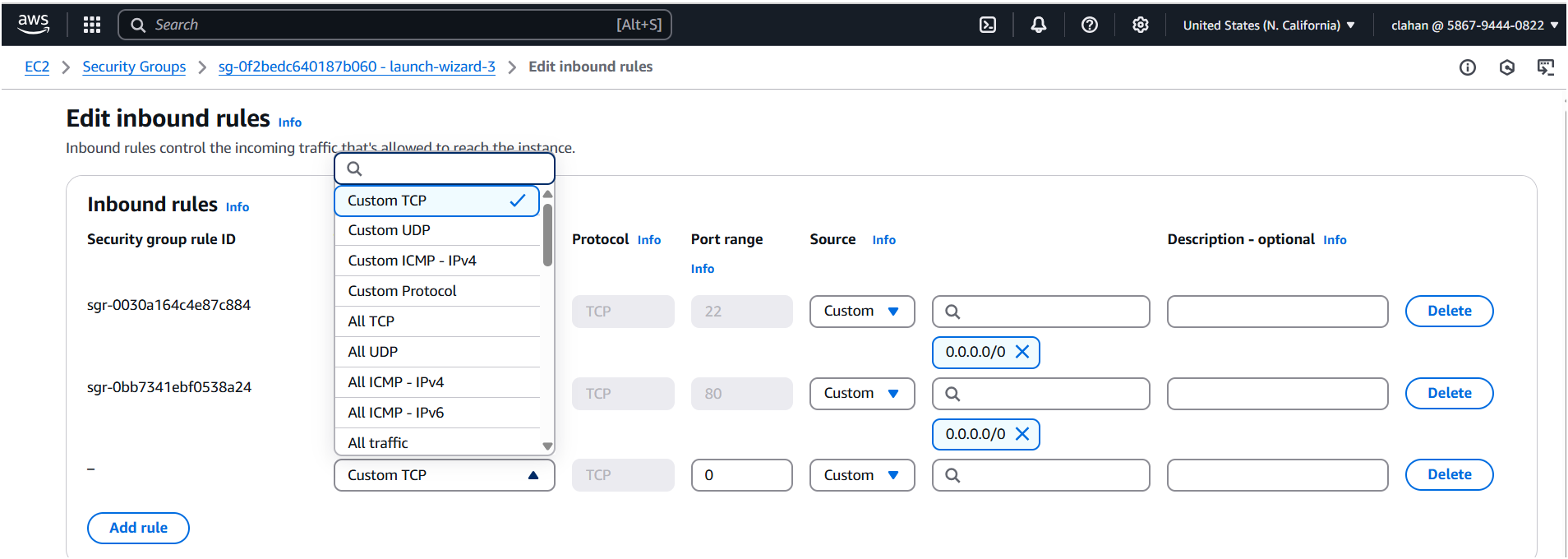
* Two containers are running successfully before accessing the app we need to open all tcp or app port number on ec2 instance security group

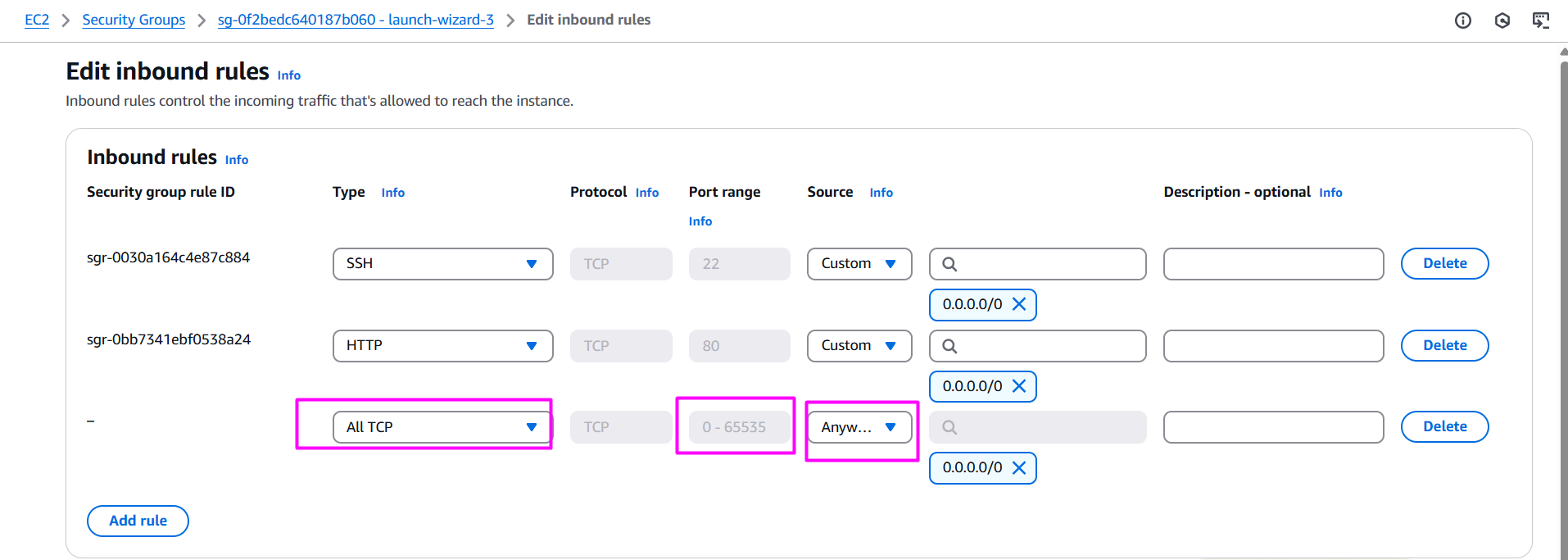


* Click on edit inbound rules

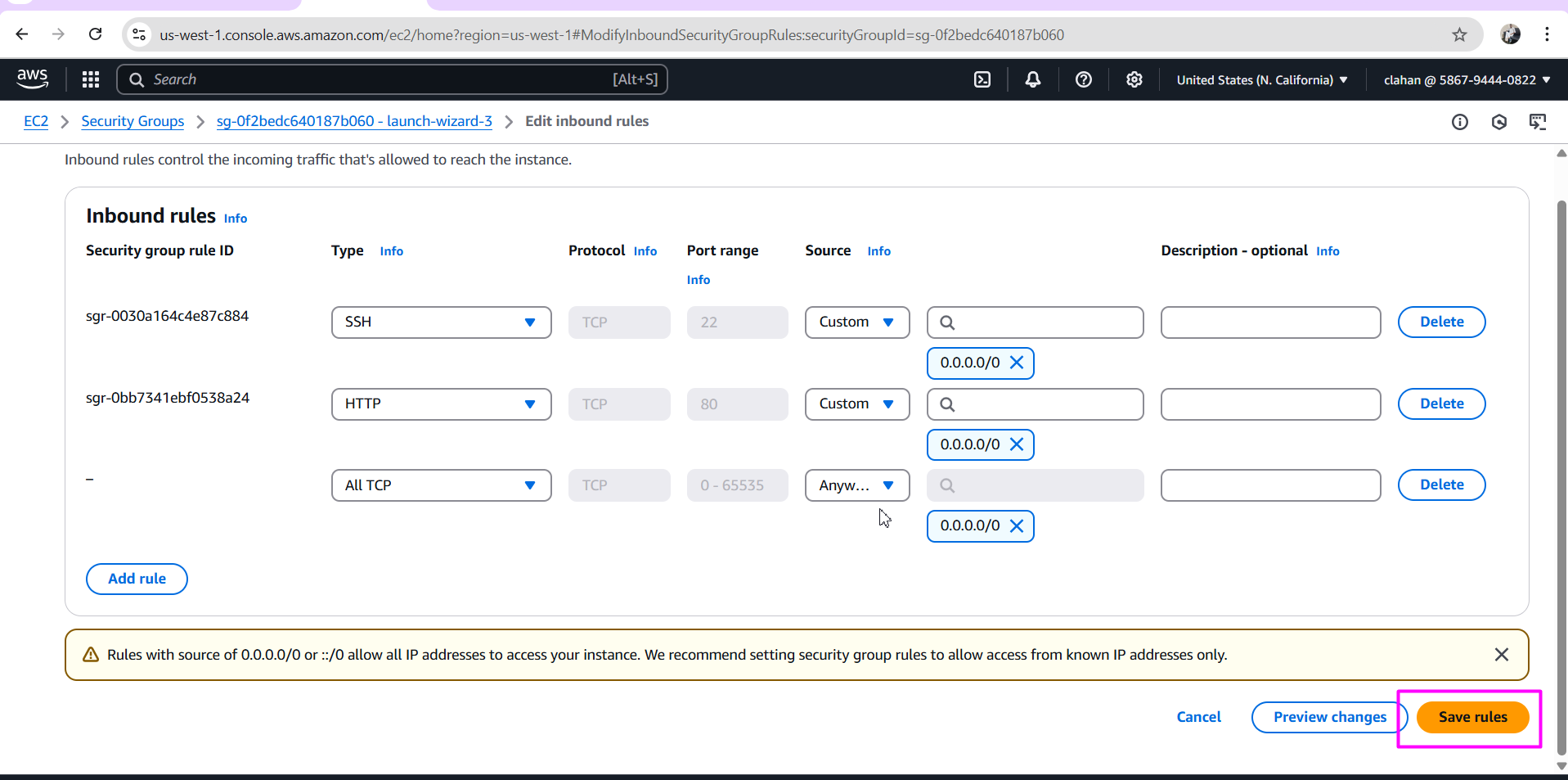


* Select all tcp if you want to open all port numbers or go with custom tcp if you want to open only individual port number and select anywhereipv2

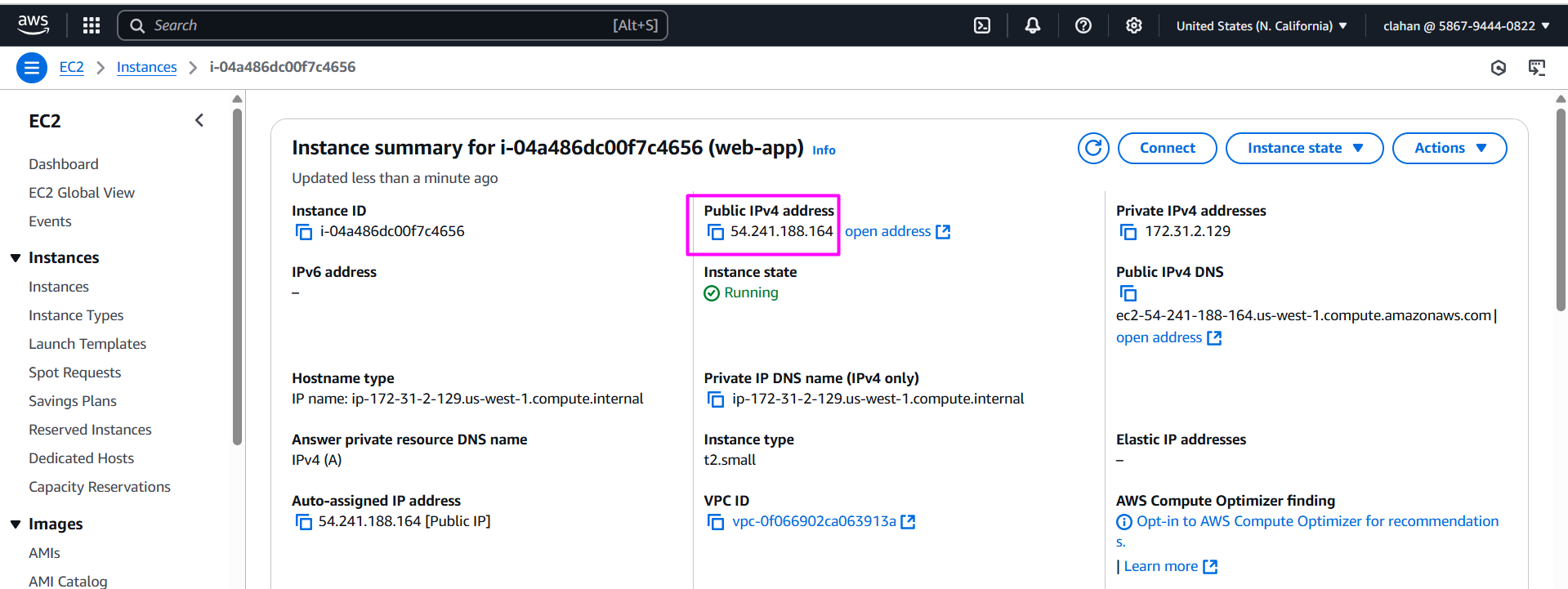




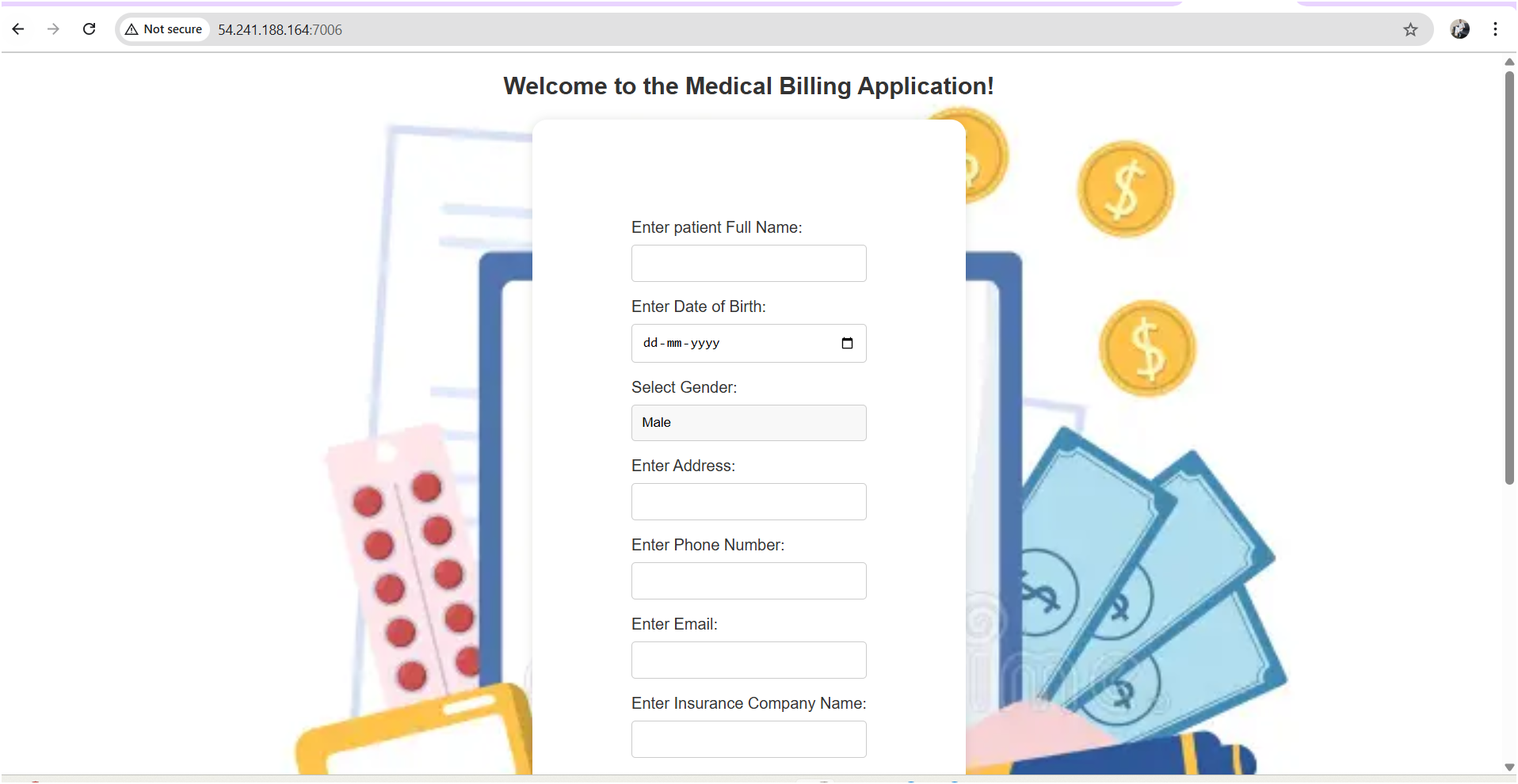
* Save your rule



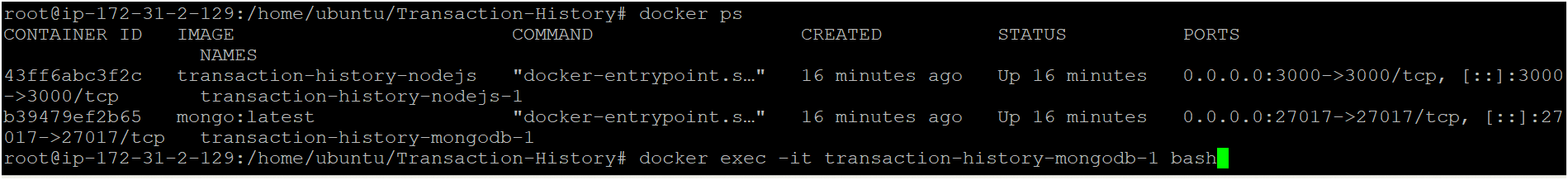
* Now access your app by using your instance public ip add with app port no ( before that you must enable port number on security group)

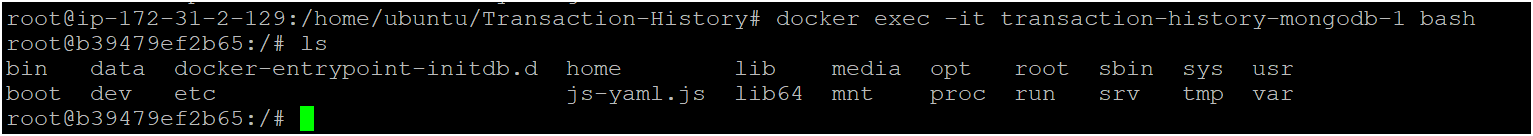


* Open browser paste ip address as shown the image 👇

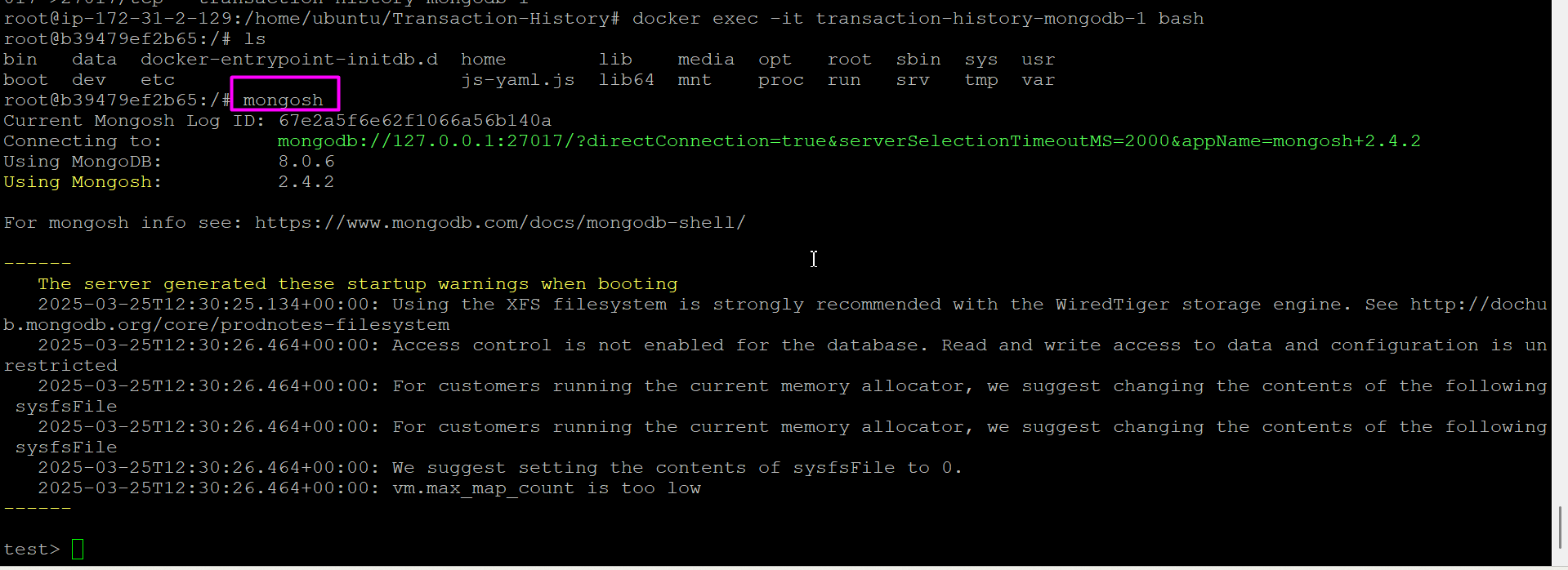


* Now how to check your data is stored in mongodb or not , follow the below steps. (note: use actual names )

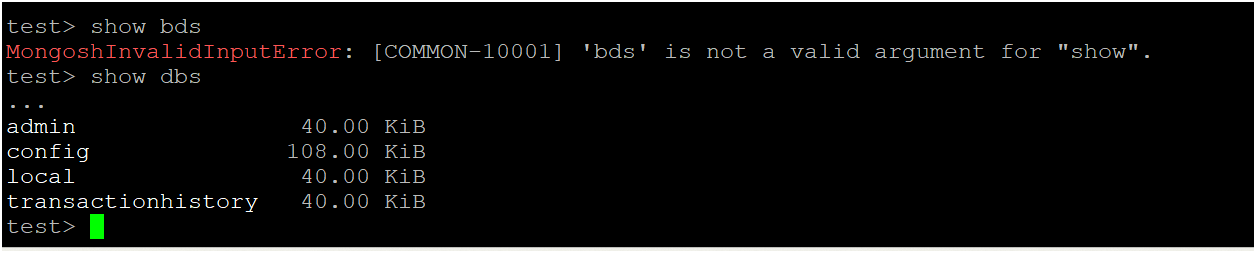




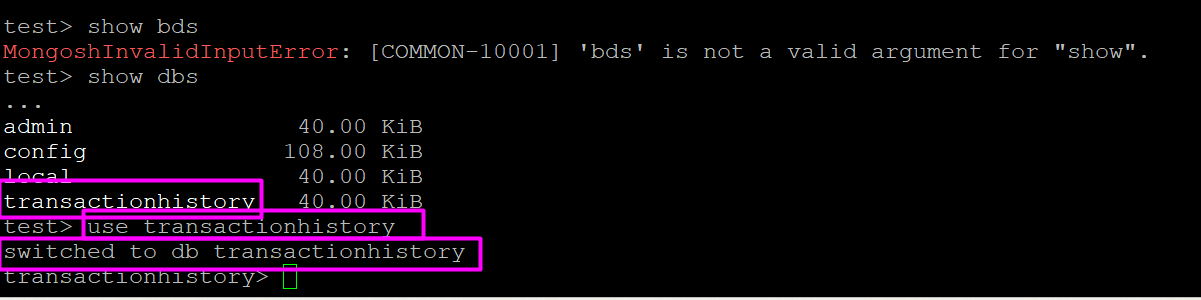
* Once inside the container, start the MongoDB shell using: mongosh



* Inside the MongoDB shell, run: show dbs



* Switch to a Database
* Select the database you want to view (replace <database\_name> with the actual name):

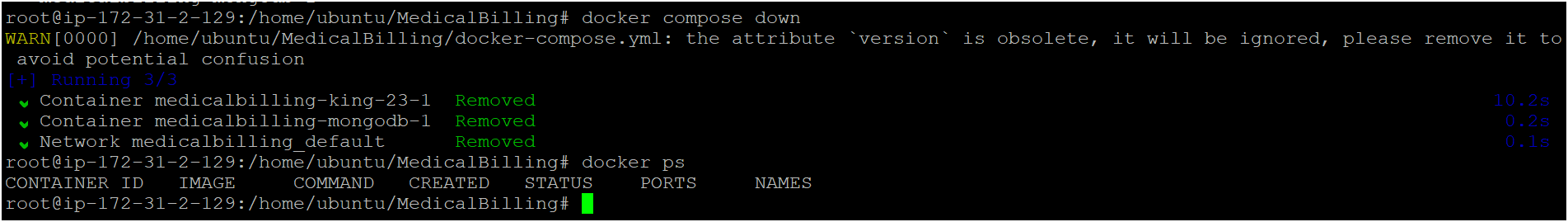


* To see the collections (similar to tables)

Run : show collections

* Now delete all containers using below command

docker compose down



* Delete your ec2 instance and other resources which you used for this app.