Container orchestration using Dockerswarm

**1.Introduction**

In modern software development, efficiently managing and deploying containerized applications across multiple servers is essential. Docker Swarm, a native Docker orchestration tool, provides a powerful yet simple solution for this task. It enables you to manage a cluster of Docker nodes as a single virtual system, ensuring seamless application deployment and scalability**.**

Docker Swarm offers key features such as load balancing, automatic scaling, and service discovery, making it an excellent choice for running microservices and distributed applications. By distributing services across multiple nodes, Docker Swarm ensures high availability and fault tolerance, automatically redistributing workloads in case of node failures.

**2.Tools And Methodologies:**

In this project I have used number of tools including some devops tools and the techniques are listed below.

Programming language: Python

Framework: Flask

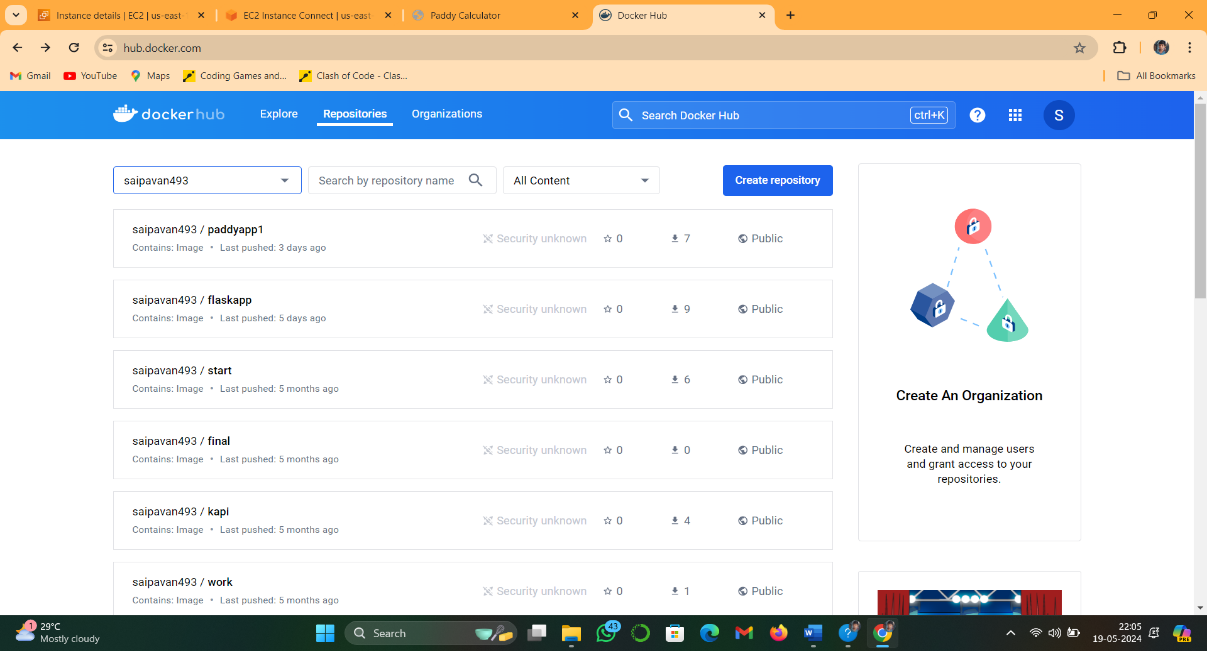
Cloud Platform : AWS

Devops tools: Dockerswarm

Container Repository: Dockerhub

**3.Containerization:**

I have used Docker to create container for my application. For local environment,I have installed docker desktop.Dockerize the application and build an image and run in a local host next push into docker hub. By following these commands.

* Build Image: docker build -t . paddyapp .
* Run image :docker run -d -p 5000:5000 paddyapp1
* Tag image: docker tag paddyapp1:latest saipavan493/paddyapp1:latest
* Login Dockerhub:docker login
* Push to Docker hub: docker push saipavan493/paddyapp1:latest

Application pushed into Dockerhub

**4. Container orchestration with Dockerswarm**

**Step 1:** Create two or three Ec2 instances.one will be manger and remaining are

Workers.

**Step 2:** Edit the security group and add the following rules.and save it

* -TCP port 2376 for secure Docker client communication. This port is required for Docker Machine to work. Docker Machine is used to orchestrate Docker hosts.

-TCP port 2377. This port is used for communication between the nodes of a Docker Swarm or cluster. It only needs to be opened on manager nodes.

-TCP and UDP port 7946 for communication among nodes (container network discovery).

- UDP port 4789 for overlay network traffic (container ingress networking).

-TCP port 22 to SSH into our instances remotely

**Step 3:**Install docker on each machine.by using following script

#!/bin/bash

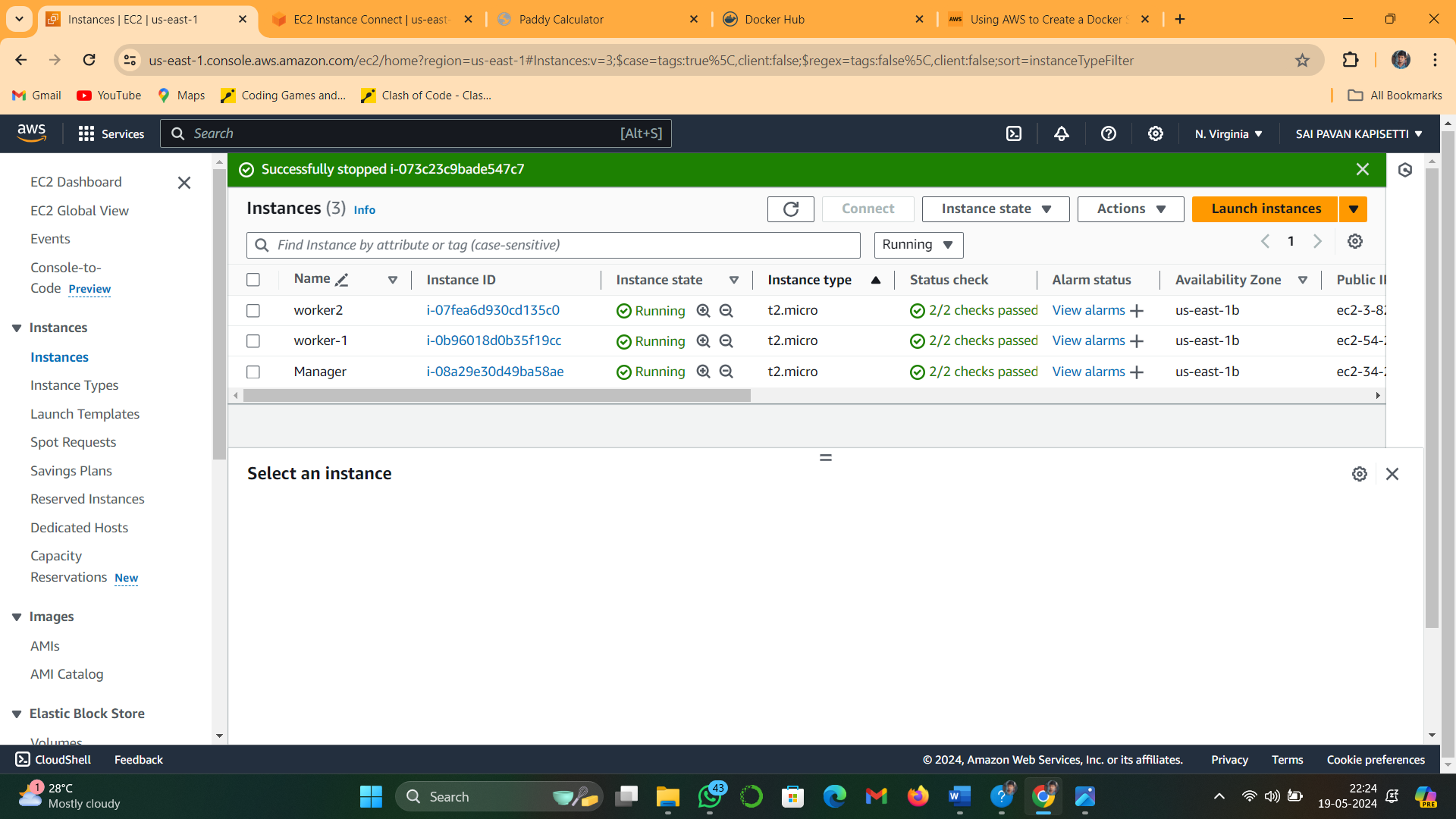
sudo yum update

sudo yum -y install docker

service docker start

usermod -a -G docker ec2-user

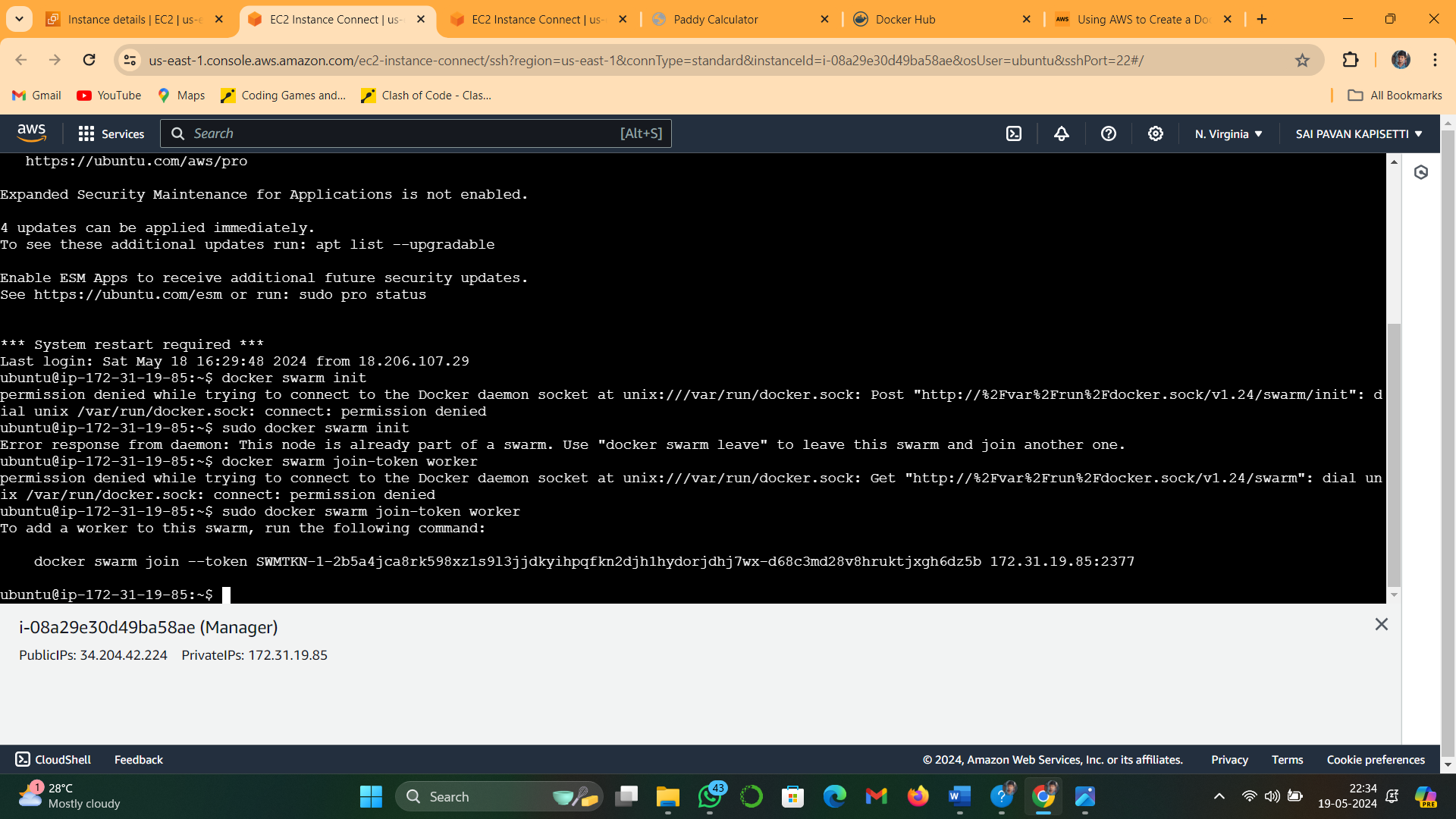
**step 4:** after sucesfully installed docker on all machines assing one instance as manager and remaining are workers.



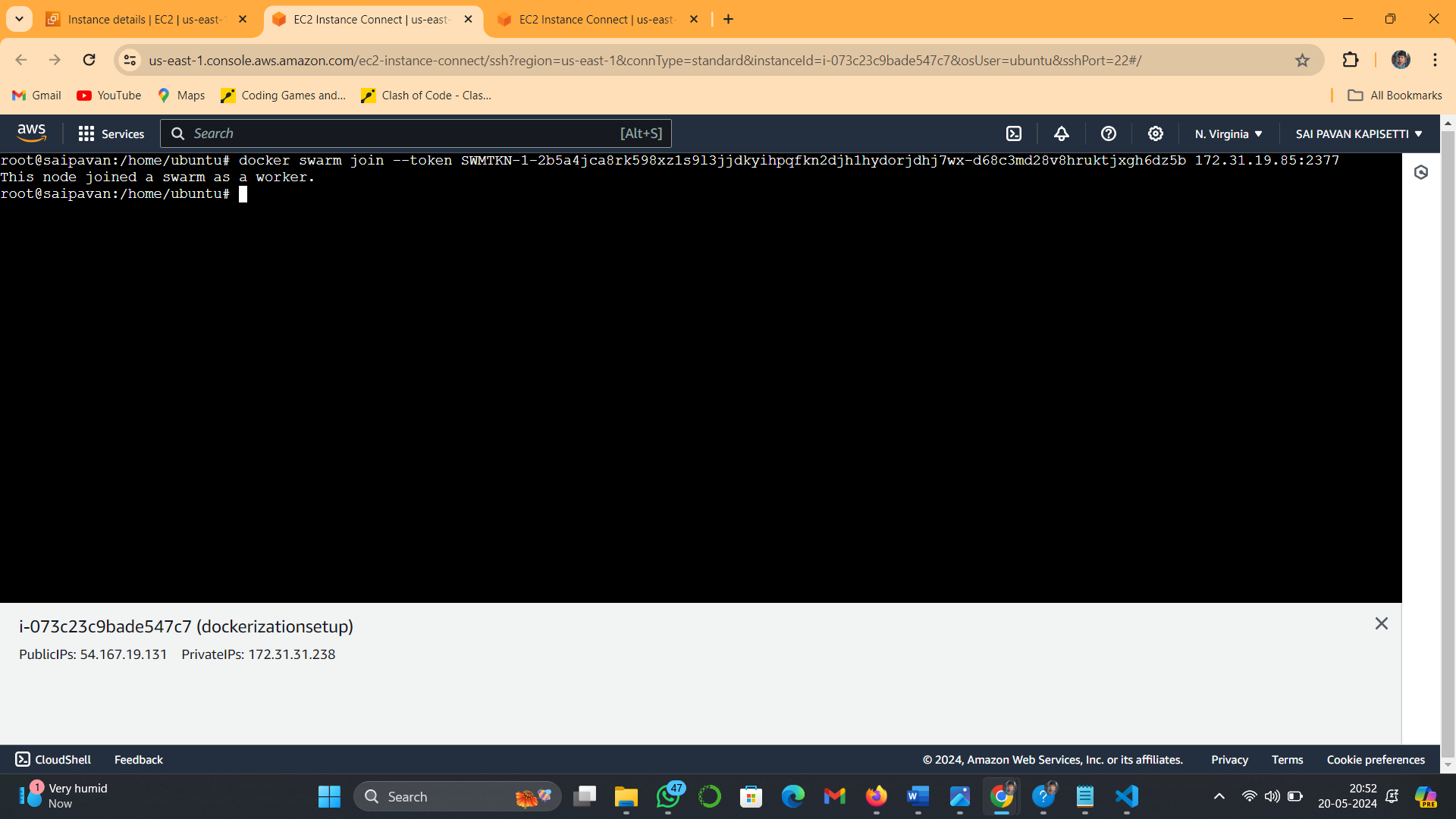
**Step 5:** to setup dockerswarm cluster.open the docker swarm manager node and enter the following command**.**

**docker swarm init**

generated a token enter these token to join a cluster in worker nodes



**step 6:** open the worker nodes and join by using following command.

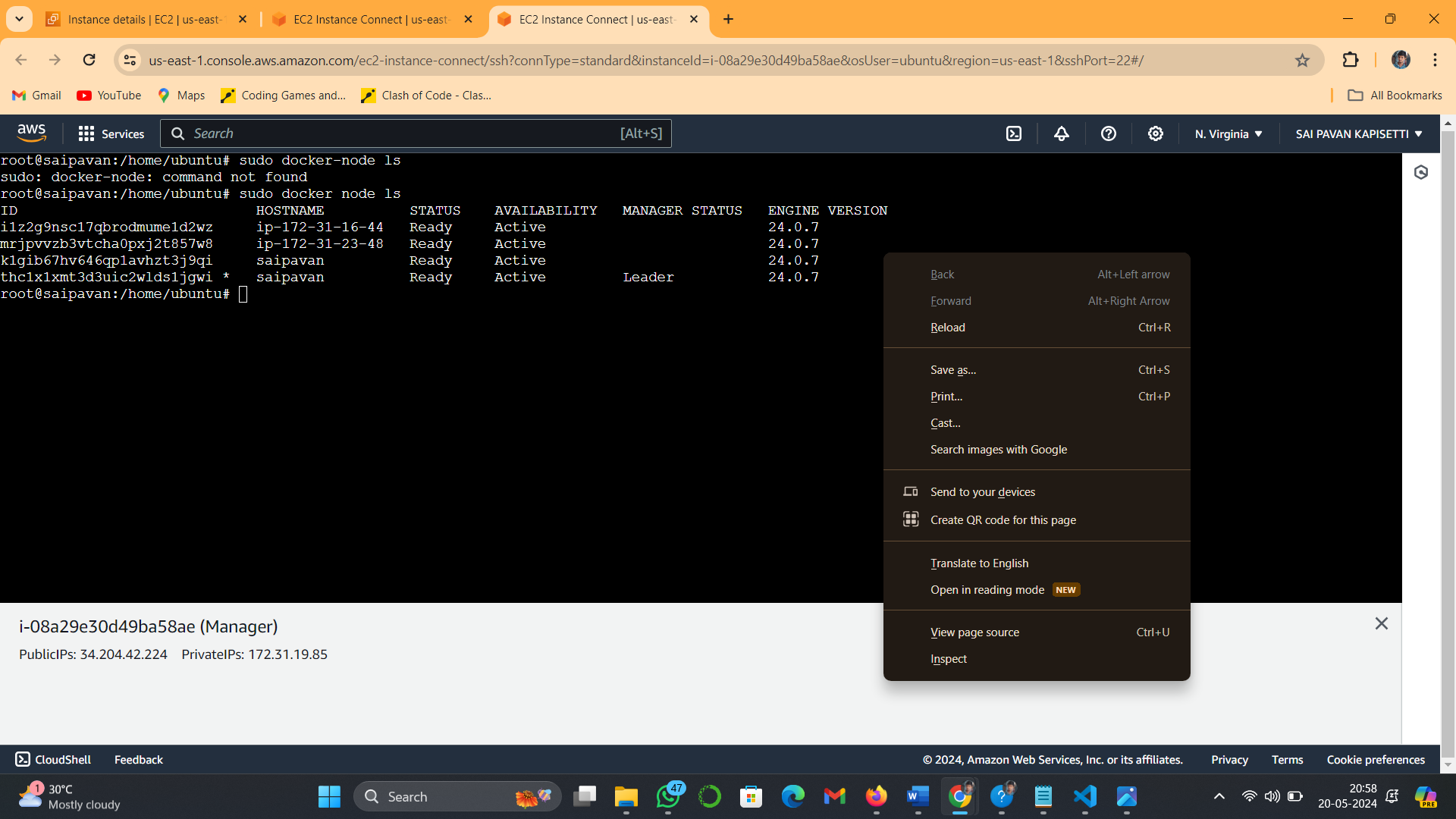
 **docker swarm join --token SWMTKN-1-2b5a4jca8rk598xz1s9l3jjdkyihpqfkn2djh1hydorjdhj7wx-d68c3md28v8hruktjxgh6dz5b 172.31.19.85:2377**

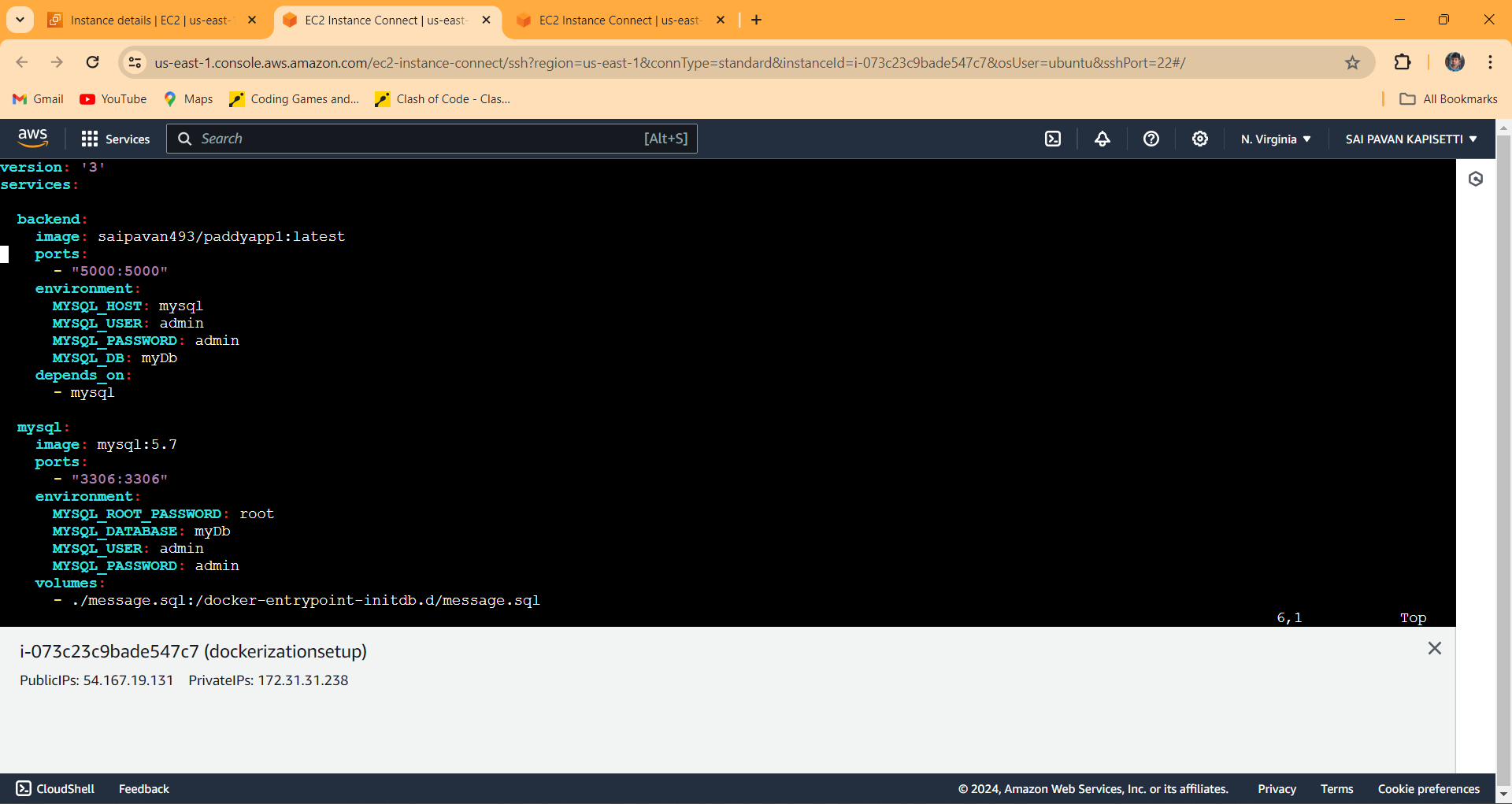
after sucessfully joined.displays message like “This node joined as a worker”

**step 7**: to check the status of nodes by using the following commands. Open the manager machine.

**Sudo docker node ls**

Displays list of all worker nodes and manger node



**Step 8:** open the manager machine .and write the docker-compose.yml file with the help of vi editor.

Yaml file :

**version: '3'**

**services:**

**backend:**

**image: saipavan493/paddyapp1:latest**

**ports:**

**- "5000:5000"**

**environment:**

**MYSQL\_HOST: mysql**

**MYSQL\_USER: admin**

**MYSQL\_PASSWORD: admin**

**MYSQL\_DB: myDb**

**depends\_on:**

**- mysql**

**mysql:**

**image: mysql:5.7**

**ports:**

**- "3306:3306"**

**environment:**

**MYSQL\_ROOT\_PASSWORD: root**

**MYSQL\_DATABASE: myDb**

**MYSQL\_USER: admin**

**MYSQL\_PASSWORD: admin**

**volumes:**

**- ./message.sql:/docker-entrypoint-initdb.d/message.sql**

**- mysql-data:/var/lib/mysql**

**volumes:**

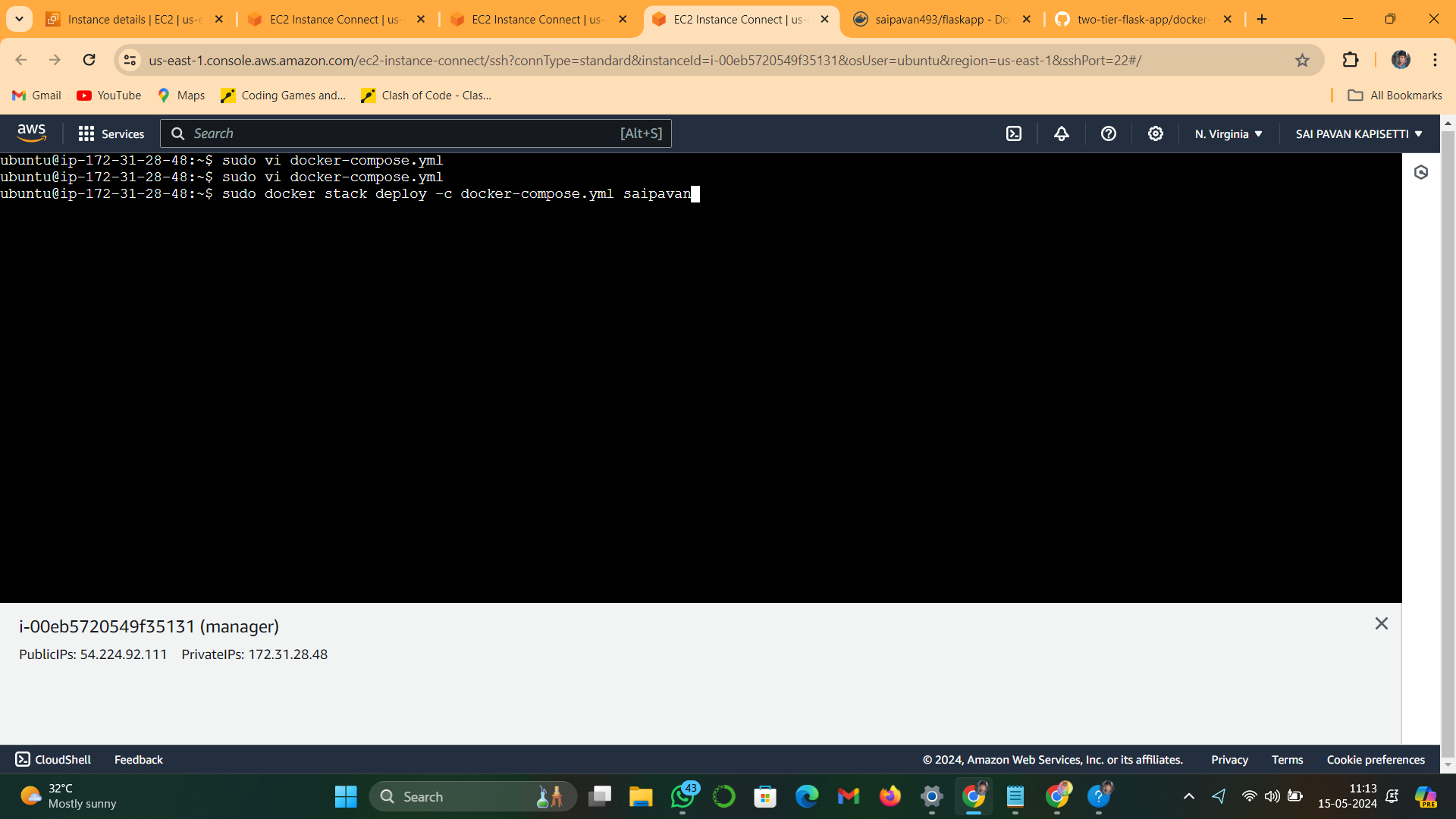
**mysql-data:**

save the file with extension of yml/yaml

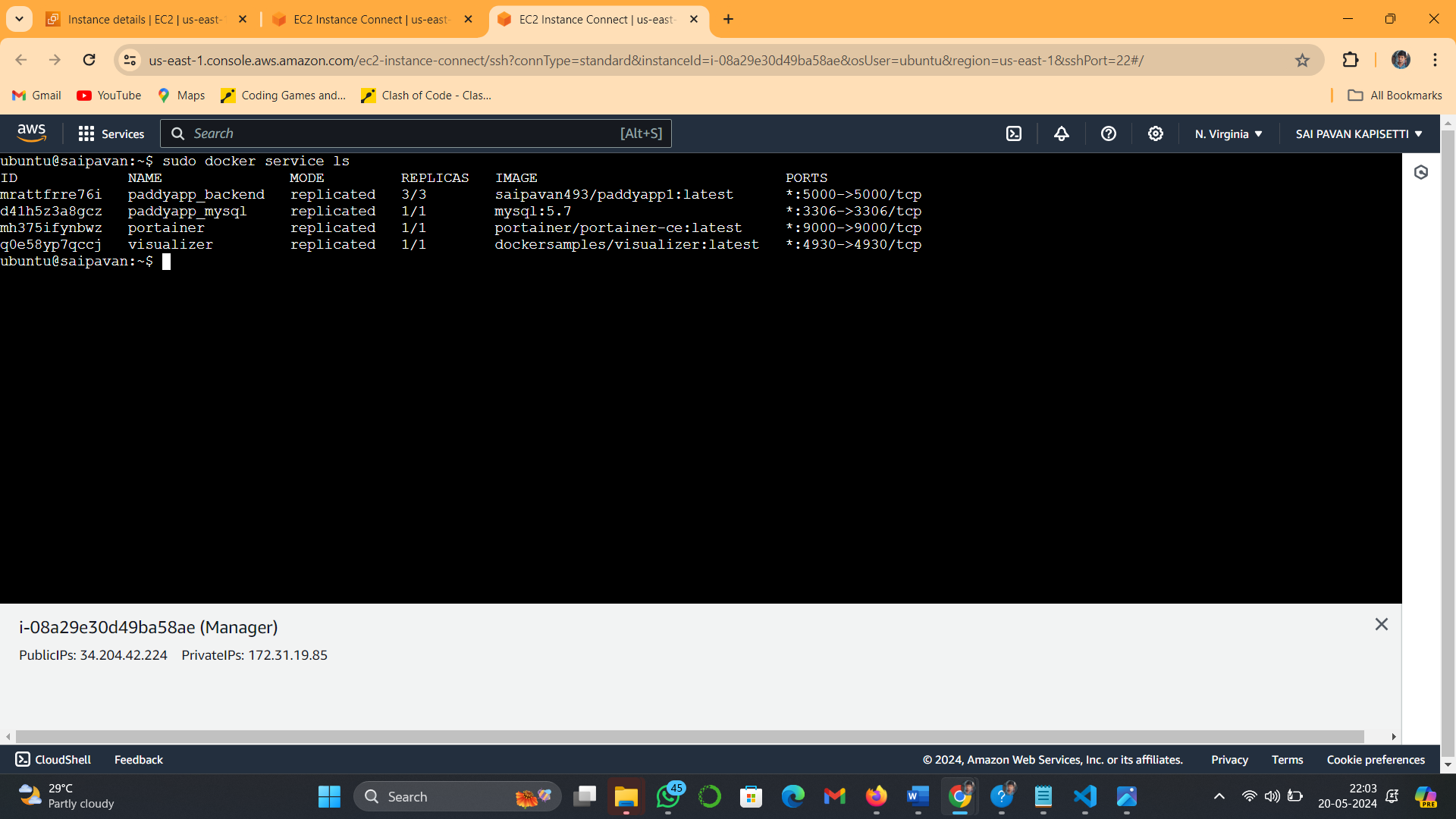
**step 9:** after saving.and execute the compose with help of the following command

**docker stack deploy -c docker-compose.yml saipavan**

and run the following command

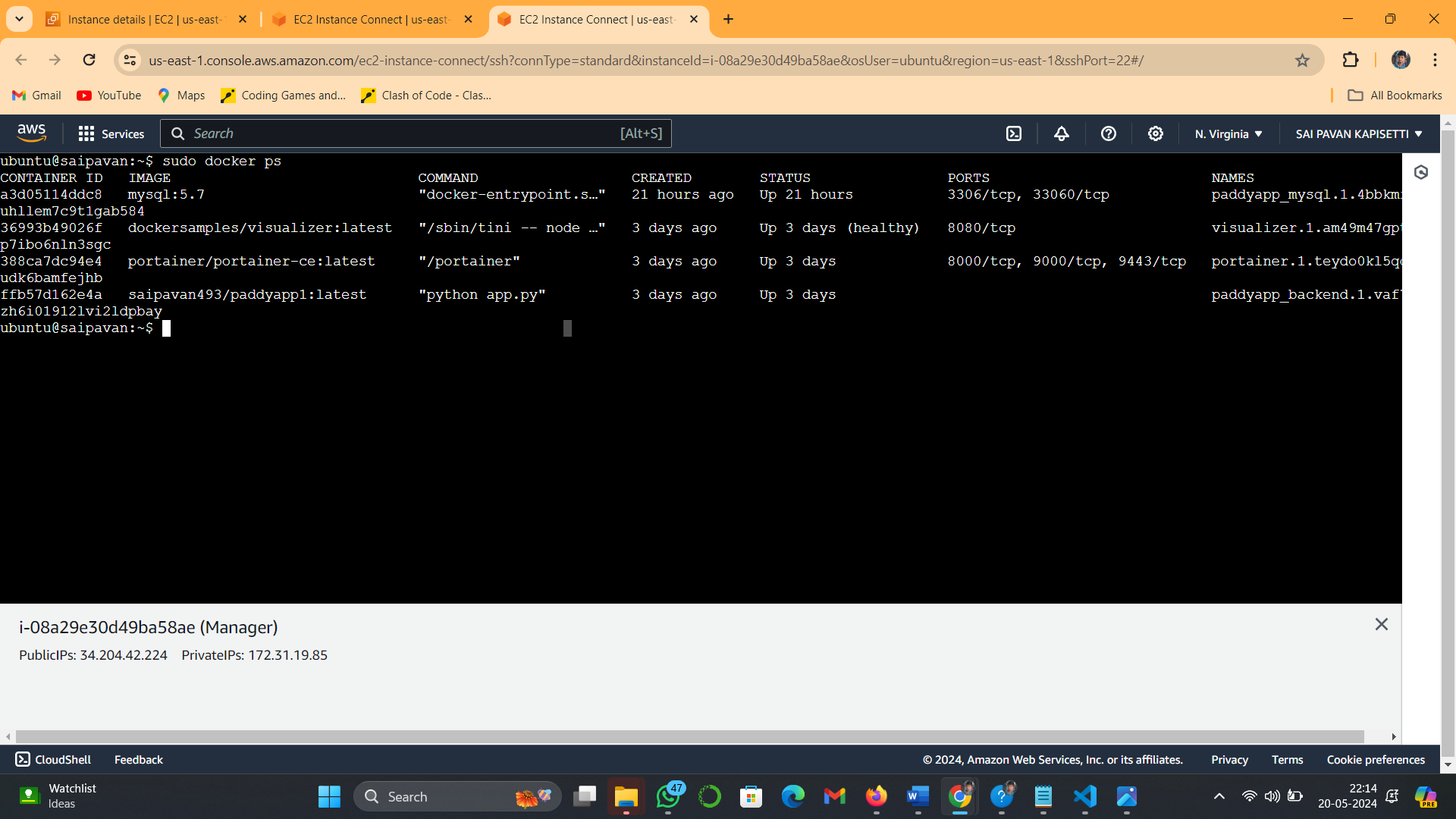


**step 10:** to see the status use the following command

 **docker service ls**

**step 11:** to see the list of containers using the following comand

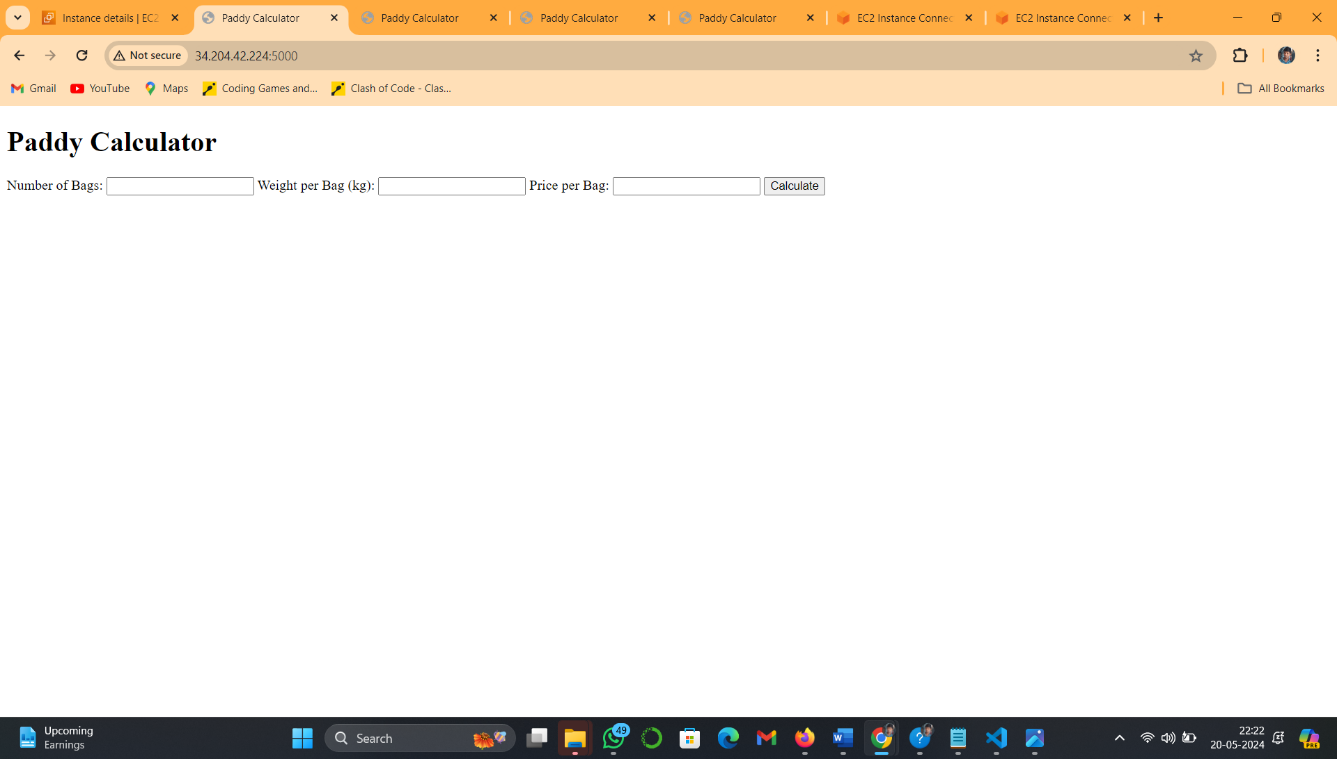
**docker ps**



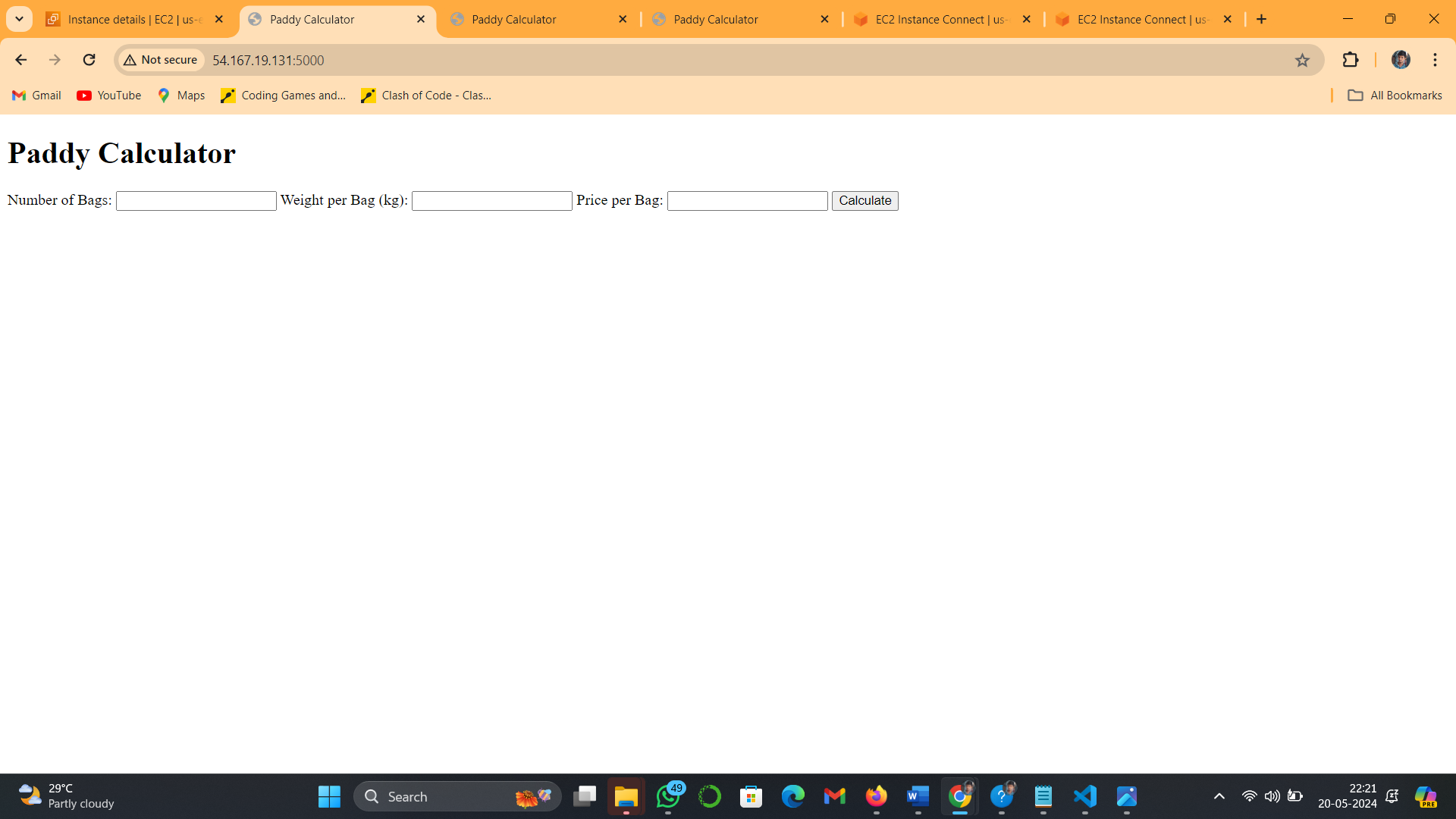
**Step 12:** after sucessfully completed all steps .acess the application with instance ip along with port number in browser. Application will run in manager and worker machines.let us check one by one

Open the browser and enter the url

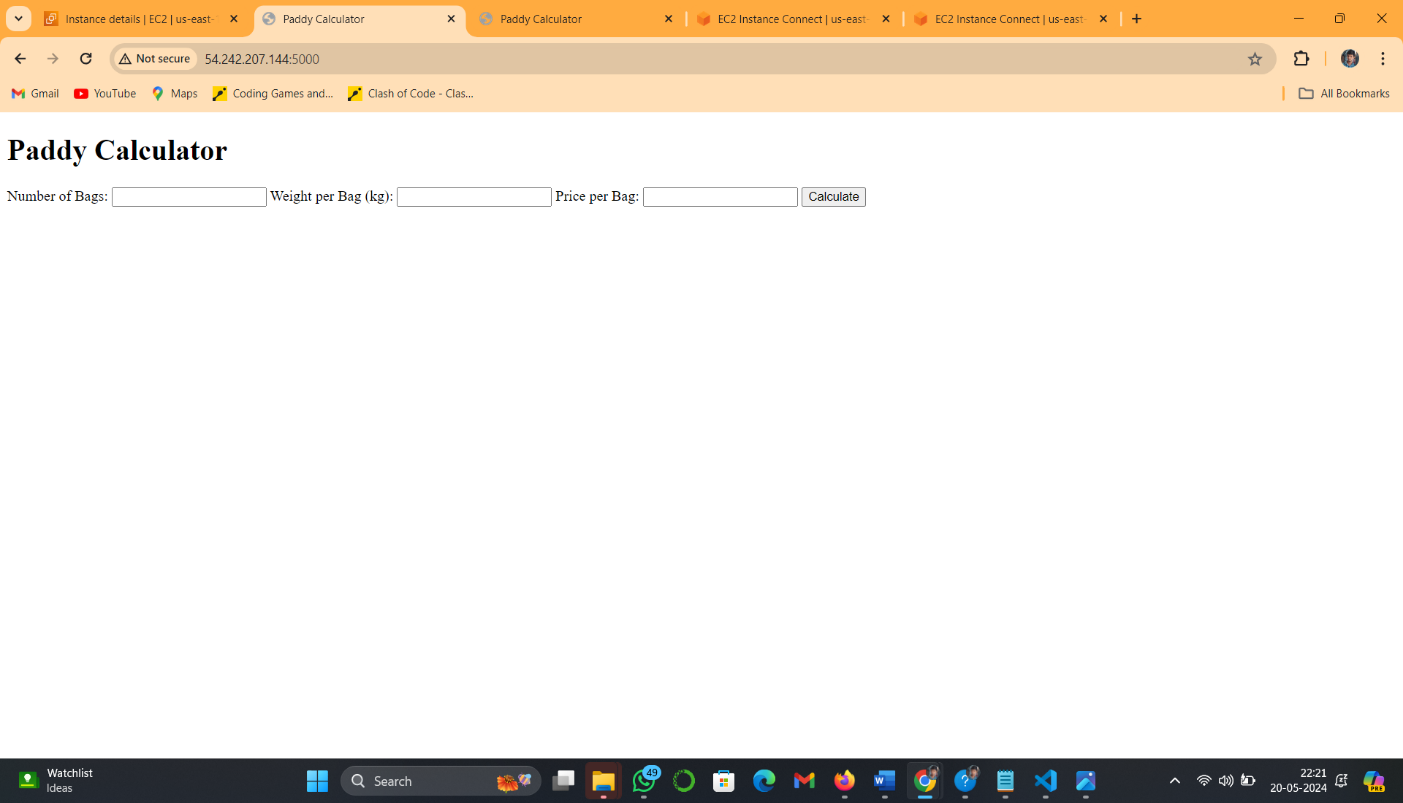
http://<instance-ip (manger/workers)>:portnumber

**Manager:**

**worker 1:**

****

**worker 2:**

****

**portainer setup:**

Portainer is a GUI management tool for docker.by portainer we can easily manage containers,stacks,volumes..etc

Portainer will acessible in port 9000 on manager machine.

To setup portainer in dockerswarm open the manager machine and execute the following command.

docker service create \

--name portainer \

--publish 9000:9000 \

--replicas=1 \

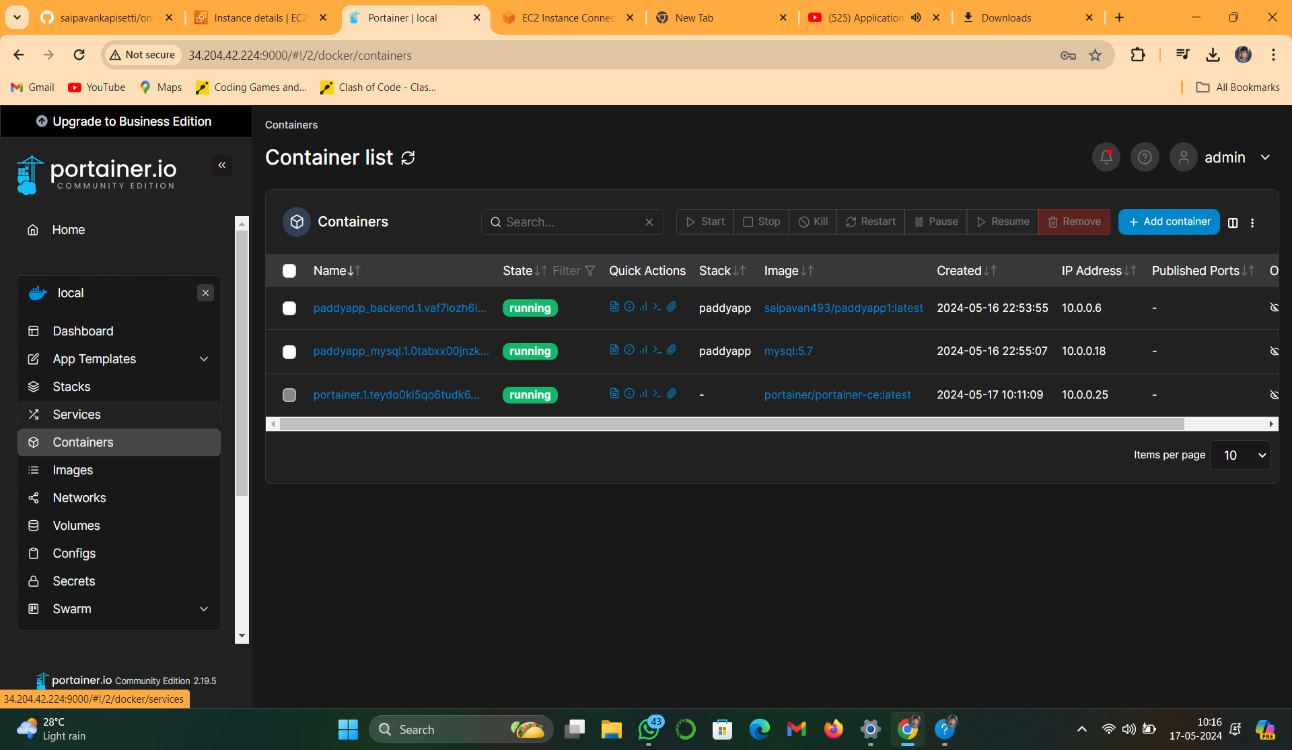
--constraint 'node.role == manager' \

--mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \

--mount type=volume,src=portainer\_data,dst=/data \

portainer/portainer-ce

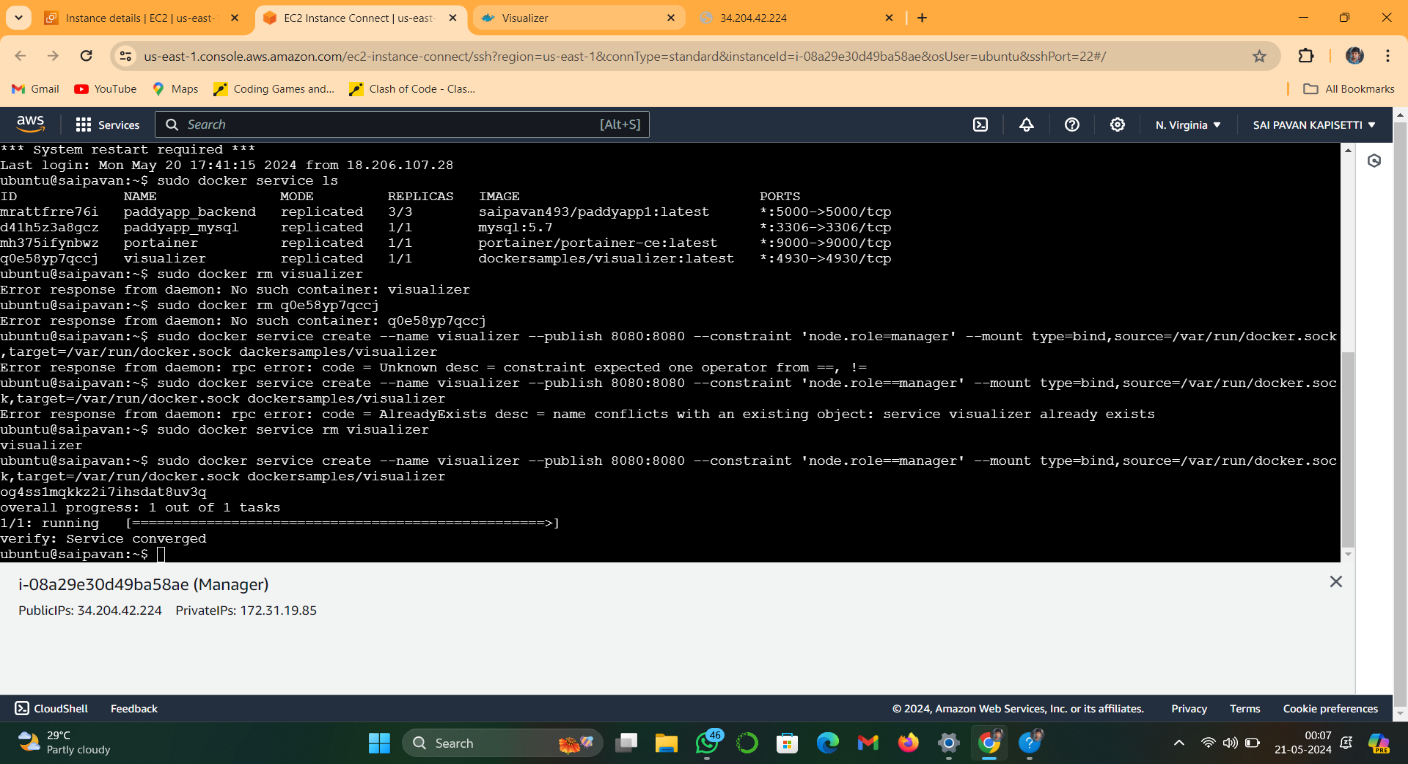
Acess portainer with http://<manager-ip>:portnumber=9000



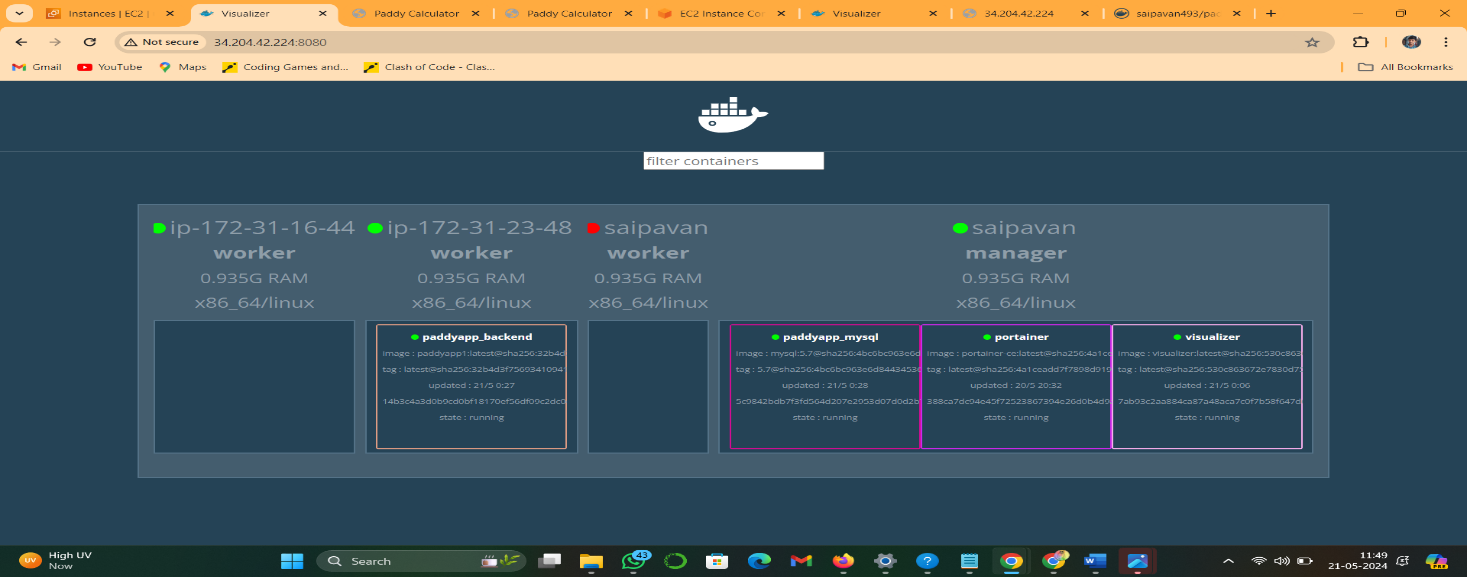
In portainer dashboard we can see list of containers,volumes,stacks,services ..etc

**Docker Graphic visualizer:**

To set up docker graphic visualizer using the following command.

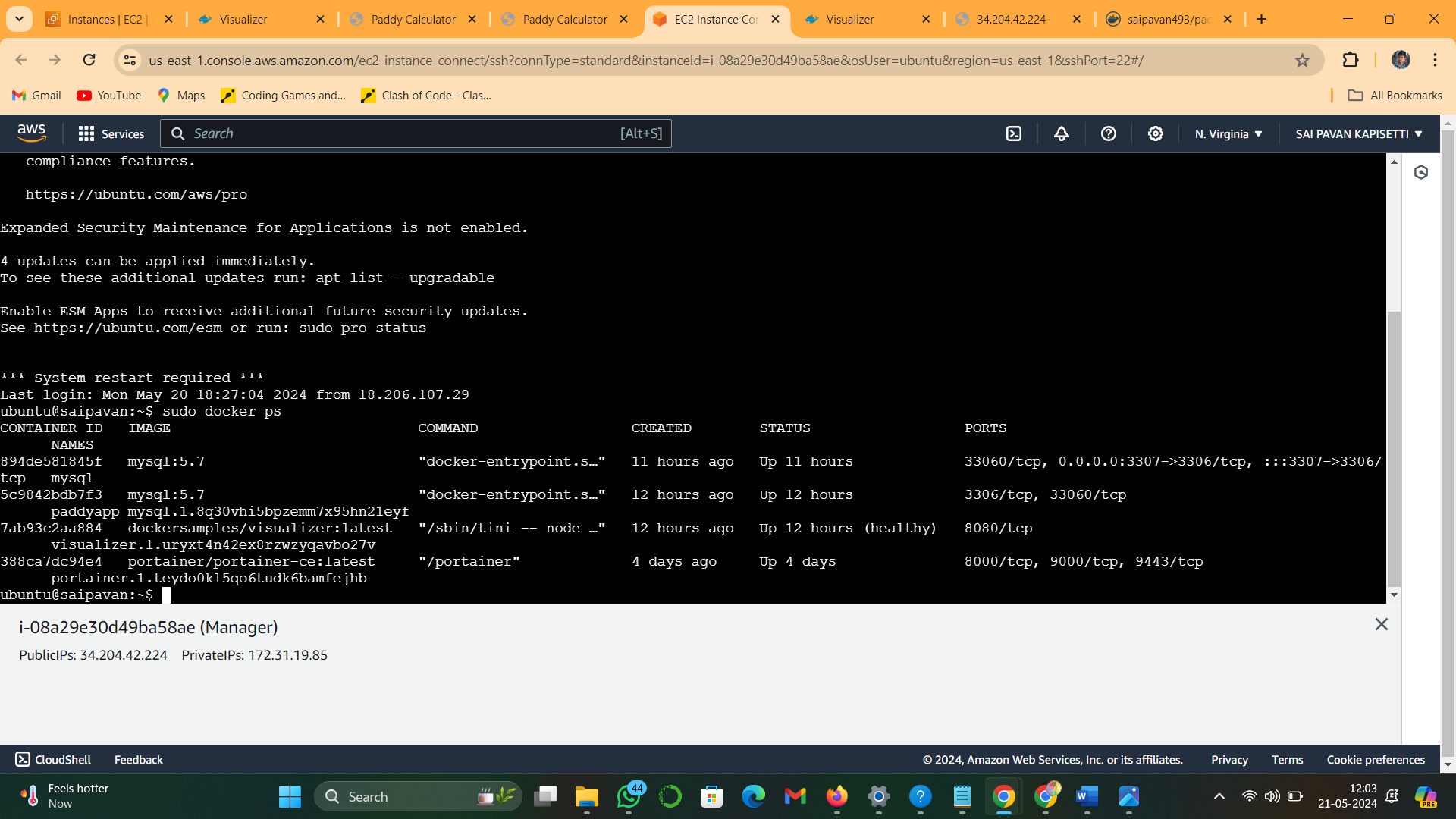
**Docker service create –name visualizer –publish 8080:8080 –constraint ‘node.role==manager’ –mount type =bind,source=/var/run/docker.sock,target=/var/run/docker.sock dockersamples/visualizer**

After sucessfully executing the visualizer .acess the application with manager machine on port 8080.

 http://<managerip>:port:8080

here displays list worker and manager node containers what are running on it.

To view the list of all running containers by using the following command on manager machine

 **sudo docker ps**

**Result:**

The application will easily acessible in manager machines and worker machines also.