**Application Deployment**

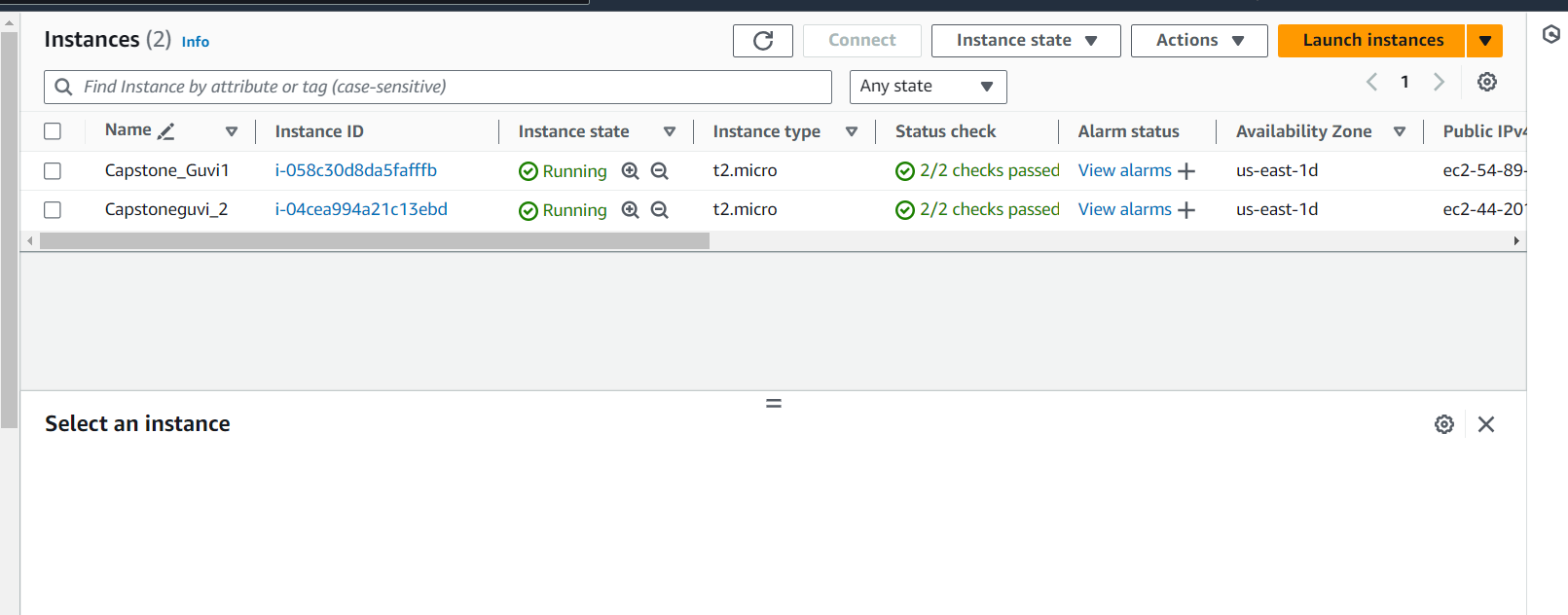
**Deploy the given react application to production ready state**

The outline of this process is deploying of a React application on an EC2 instance and setting up a Jenkins-based CICD pipeline and monitoring the heath check using Prometheus and Grafana.

**The tools utilized in this project are as follows:**

1. AWS-EC2
2. GitHub
3. Docker
4. Jenkins
5. Prometheus and Grafana

**Step 1: Set up an EC2 instance**



sudo apt-get update

sudo apt-get install nodejs

sudo apt-get install npm

sudo apt-get install git

Git Hub link : <https://github.com/ReshmaSaju/reactjs-demo.git>

git clone <https://github.com/ReshmaSaju/reactjs-demo.git>

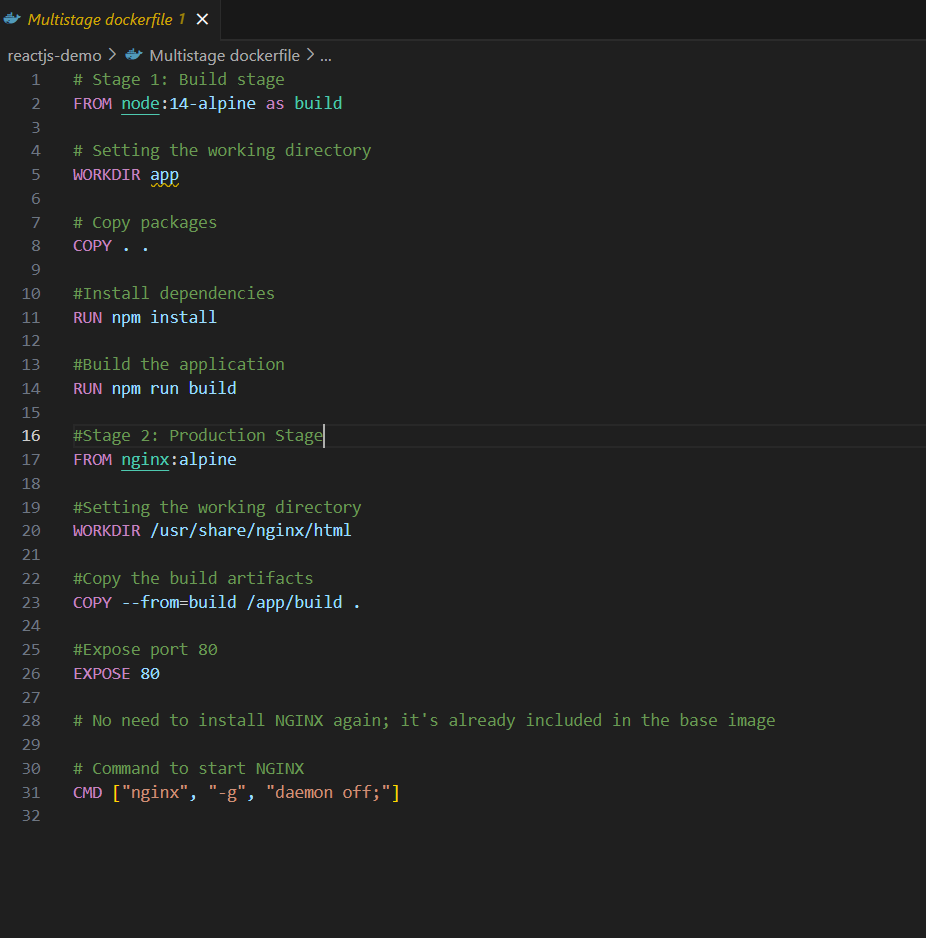
ls and move to reactjs-demo {cd reactjs-demo}

**Docker : 2**

## **Step 1: Install docker and write a Multistage Docker file:**

sudo apt-get install docker.io

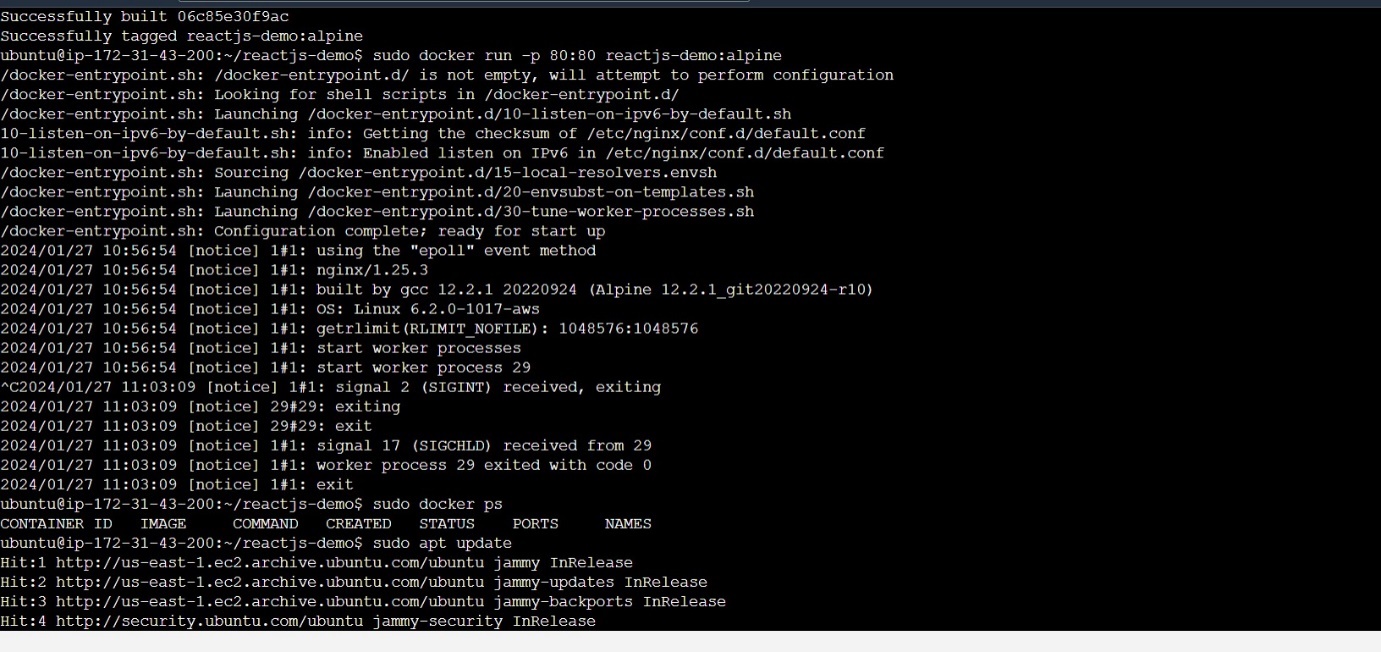
vi Dockerfile



sudo docker build -t reactjs-demo .

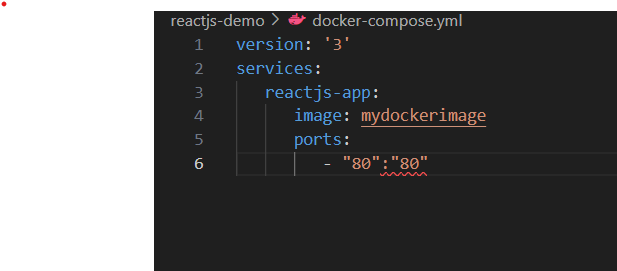
sudo docker run -itd -p 80:80 reactjs-demo:alpine

sudo docker images



**Step 2: Install docker- Compose file**

vi docker-compose.yml

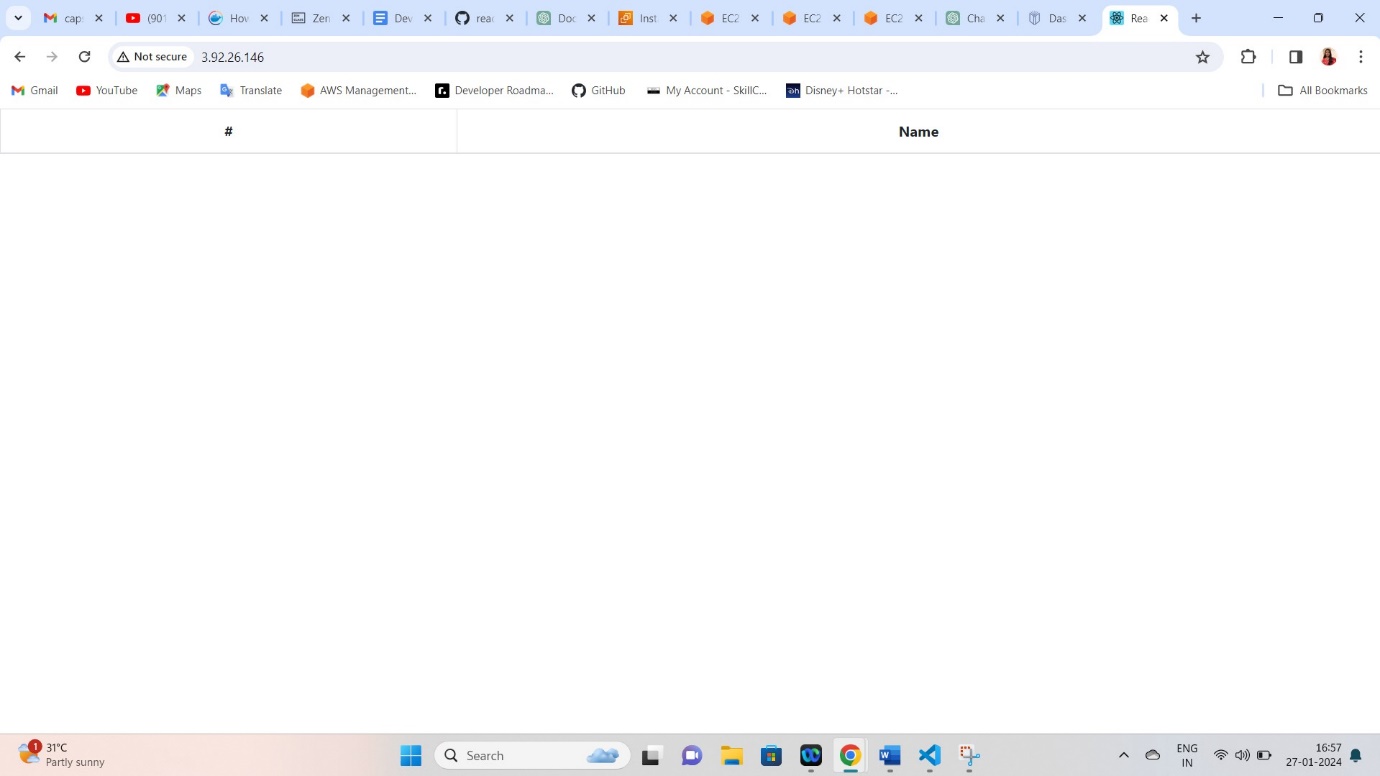


sudo apt-get update

sudo apt-get install docker-compose

sudo docker-compose –version

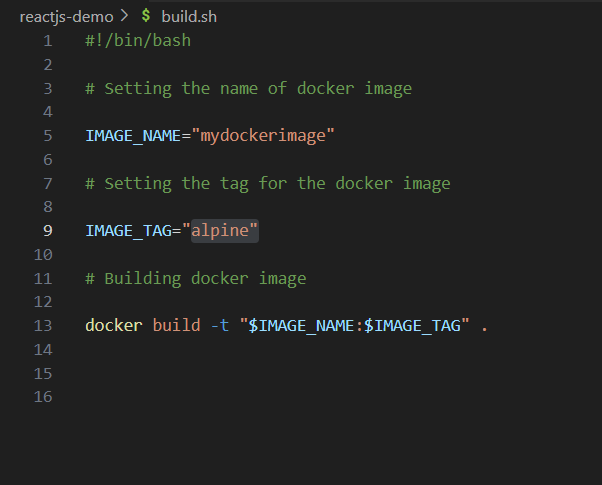
sudo docker-compose up -d



**Bash Scripting**

* Creating build.sh file for building docker image.
* Creating deploy.sh file to deploy the image in server.

**vi build.sh**



**Jenkins**

# Update the package list

**sudo apt update**

# Install OpenJDK 11

**sudo apt install openjdk-11-jre**

# Add Jenkins GPG key to the keyring

**curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null**

# Add Jenkins repository to the package sources

**echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null**

# Update the package list again

**sudo apt-get update**

# Install Jenkins

**sudo apt-get install jenkins**

# Enable Jenkins to start on boot

**sudo systemctl enable jenkins**

# Start the Jenkins service

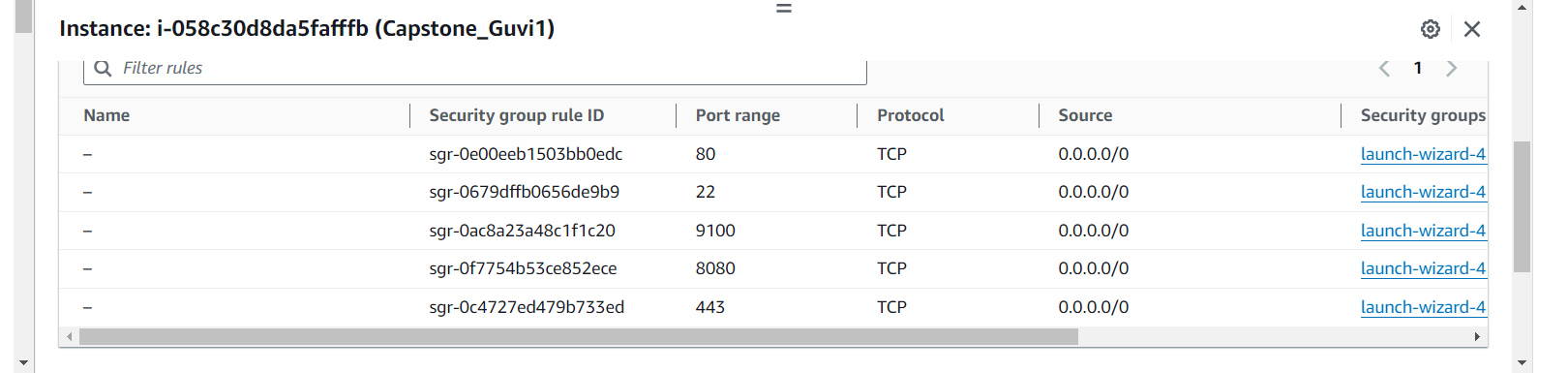
**sudo systemctl start jenkins**

# Check the status of the Jenkins service

**sudo systemctl status Jenkins**

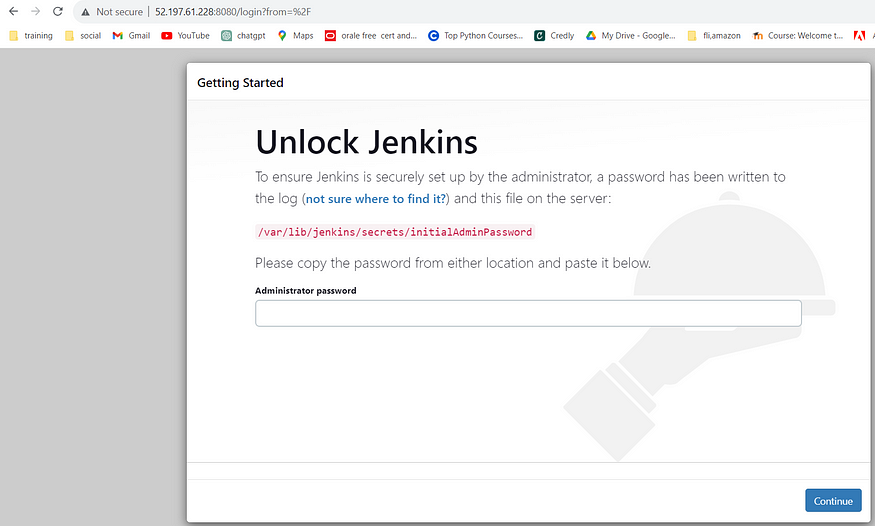
## **To access Jenkins**

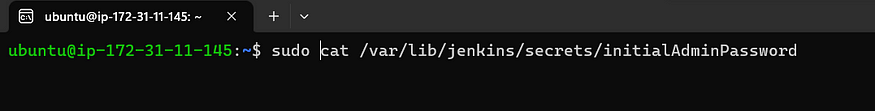
* For Jenkins we have to open port 8080 in SG.
* I have set the inbound rules for the security group as below.

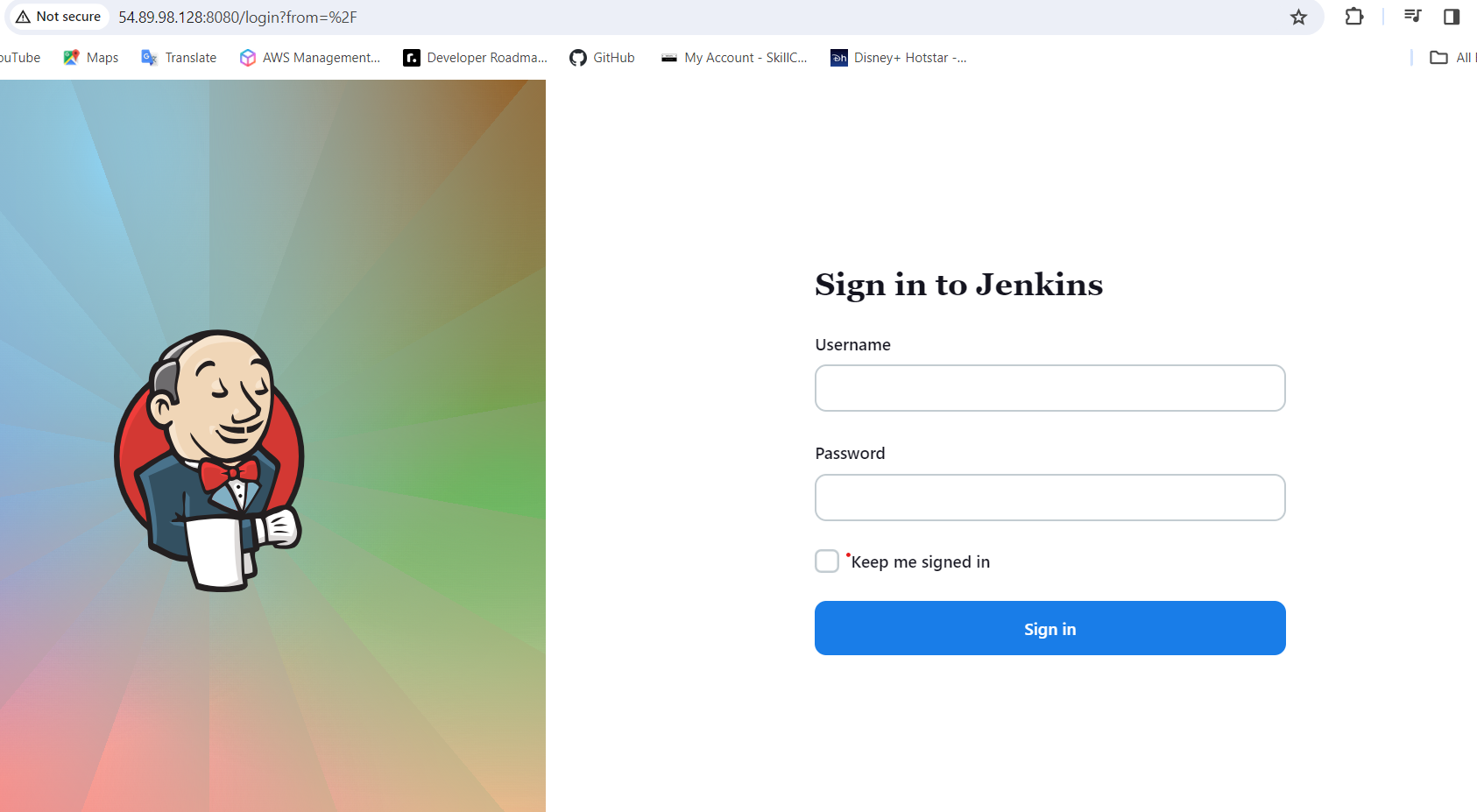


You can access the application using the format “public-ip:8080”, where “public-ip” is the public IP address of your instance.

Upon accessing Jenkins, you will be directed to a page where you can obtain the password. Use the “cat” command with the provided path on that page to retrieve the password.



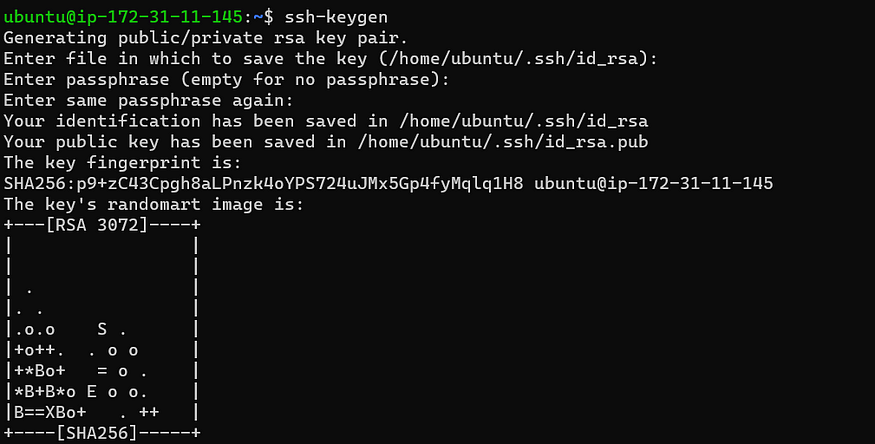


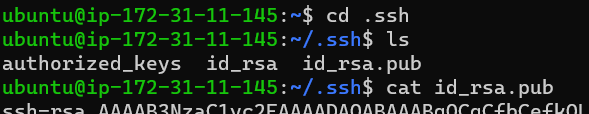


* Install all suggested plugins.

## To connect your Jenkins job with a GitHub repository, follow these steps:

* Before configuring Jenkins, you must add the public key to facilitate communication between Jenkins and GitHub. This allows Jenkins to access the source code from the GitHub repository.





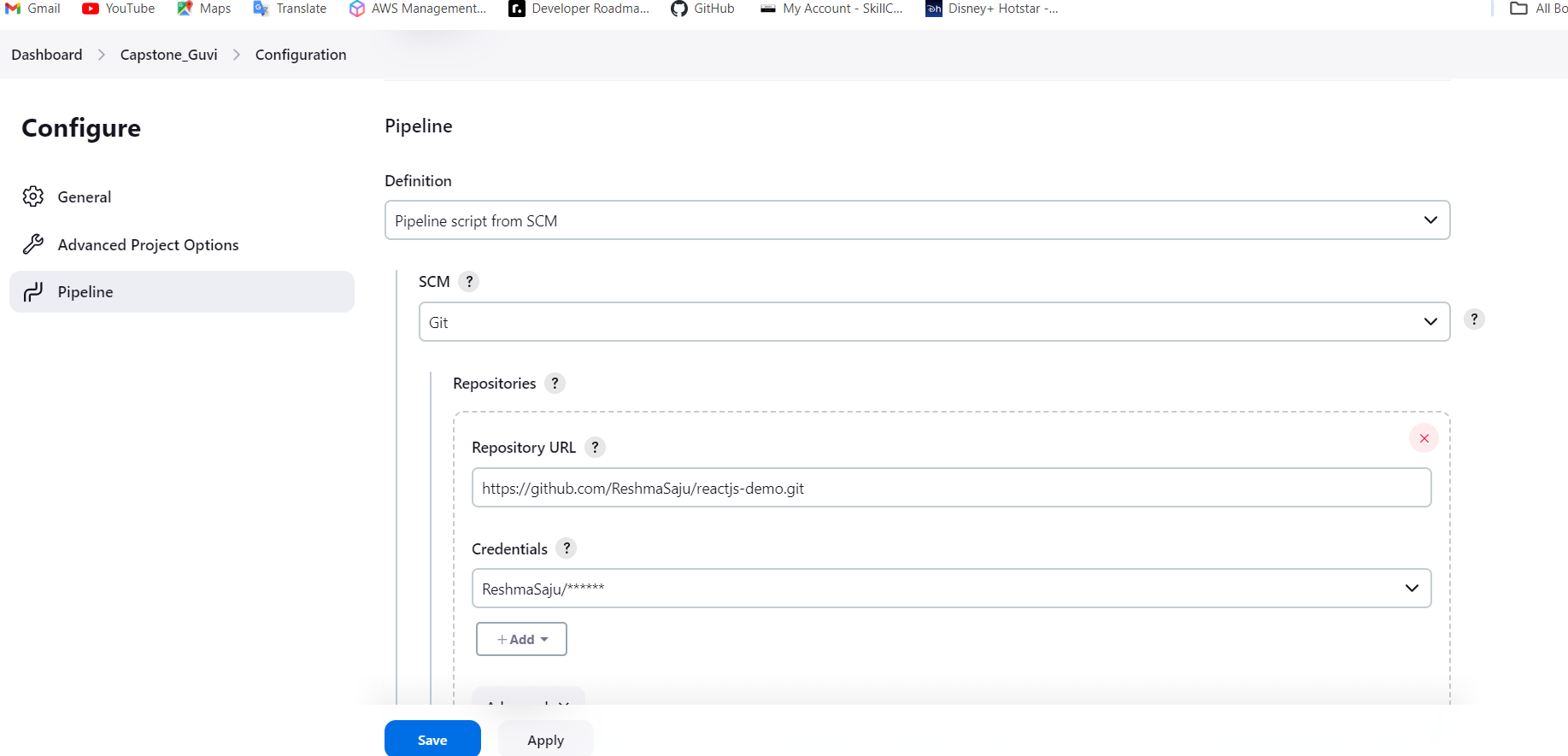
## **To enable Jenkins integration with GitHub, follow these guidelines:**

1. Open your GitHub account settings.
2. Find and access the “SSH and GPG keys” section in settings
3. Insert the public key generated using the “ssh-keygen” command. Be sure to choose the correct key-type, like “Authentication key.”

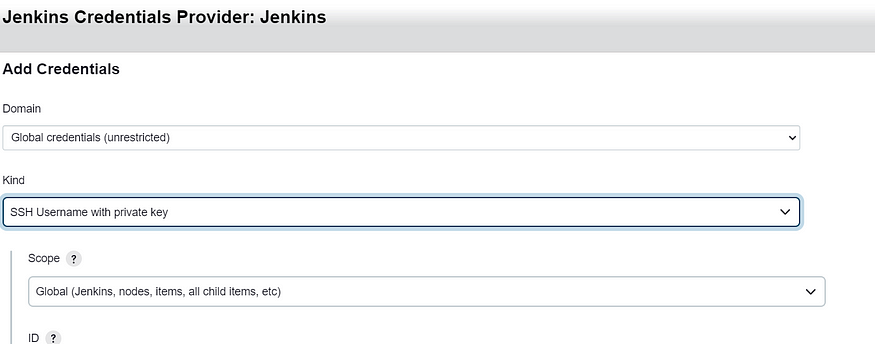
**This is the step-by-step guide to configuring Jenkins for your Reactjs application**



In the project configuration, provide the GitHub project URL to link it with your repository.



To allow Jenkins access to the GitHub code, add the credentials in the “Source Code Management” section.

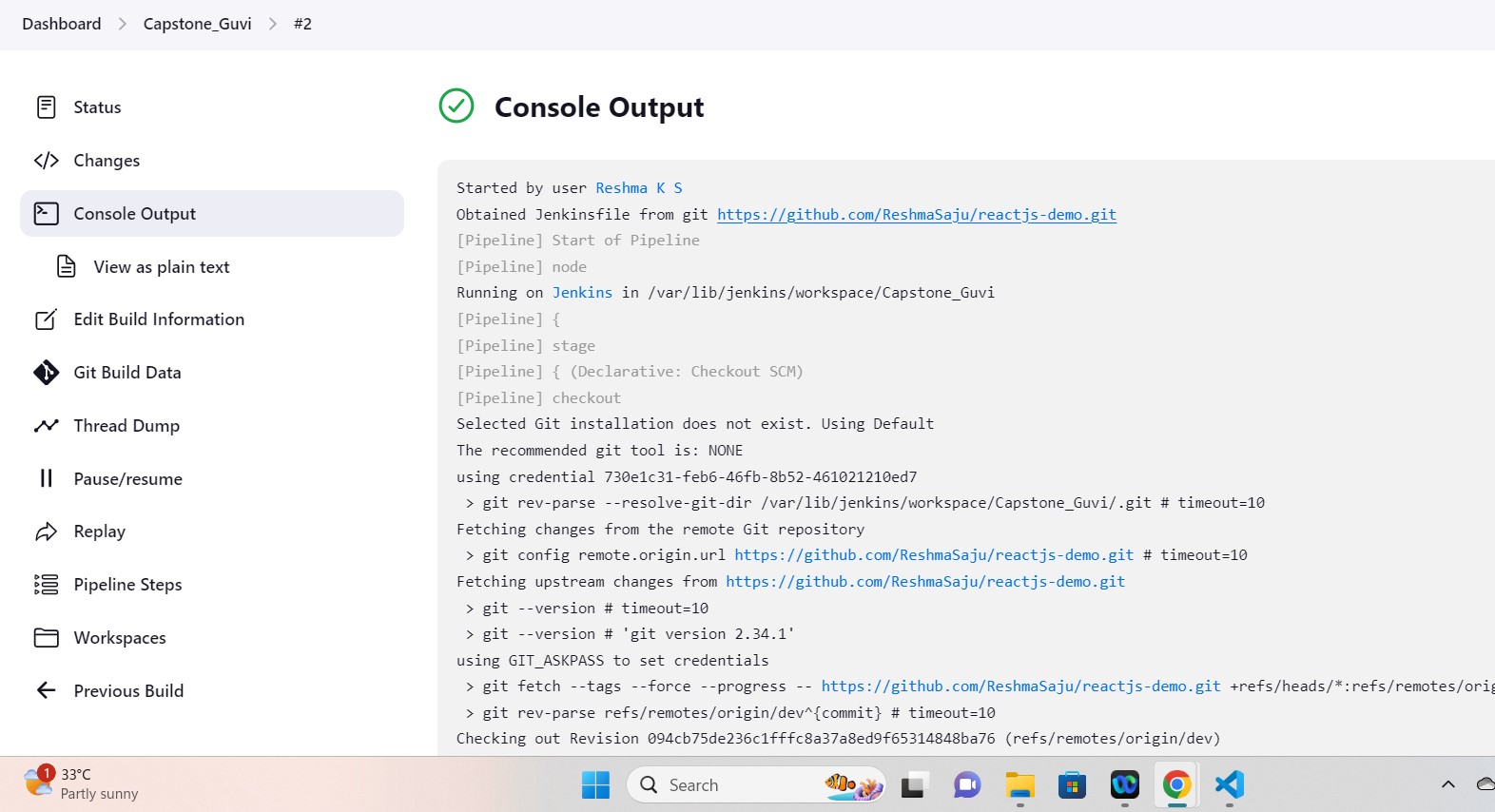


Click “Add” and input the private key generated with “ssh-keygen” as the authentication method.

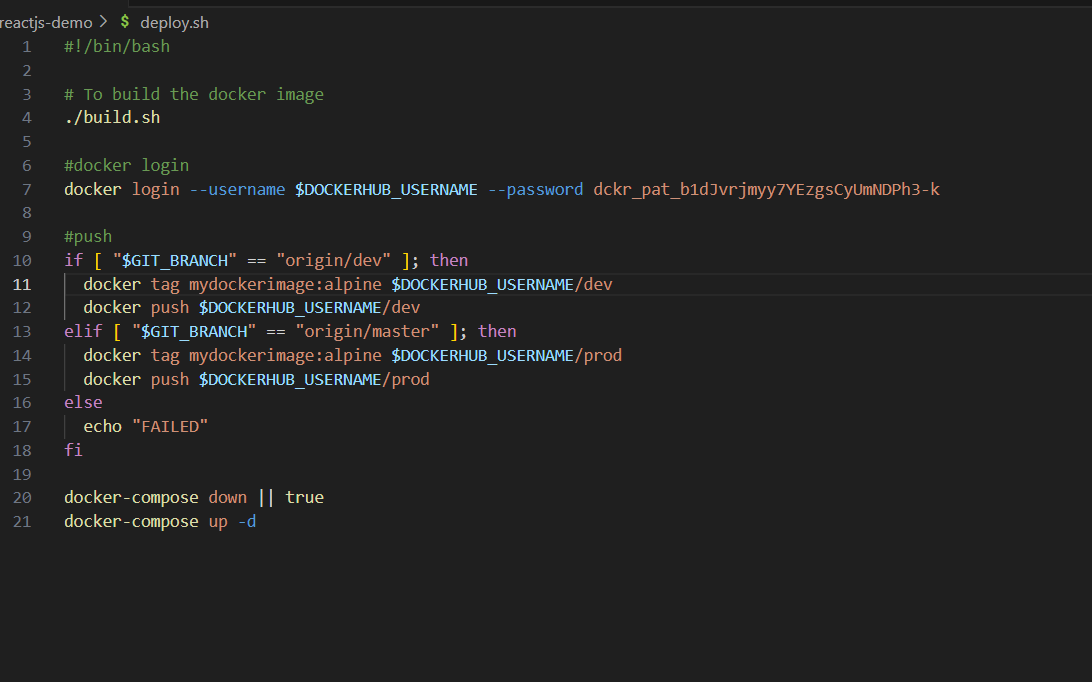
Provide a unique ID and description for the credentials. Paste the private key in the “Private Key” field and add the passphrase if applicable. Click “Add” to save the credentials.

Save the project configuration.

Now, you’re all set to build the job. Click “Build Now” to initiate the build process.

Now we have to write deploy.sh file

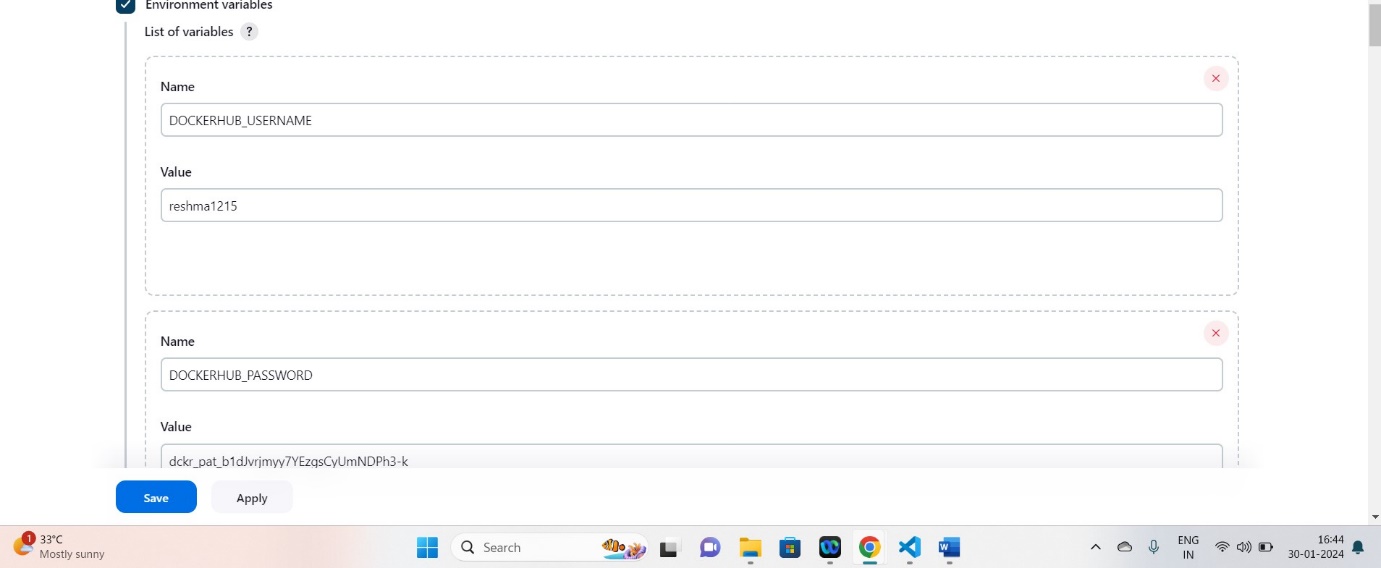
* vi deplo.sh



* Now we have to set the environment variable in Manage Jenkins- system -Environment variable

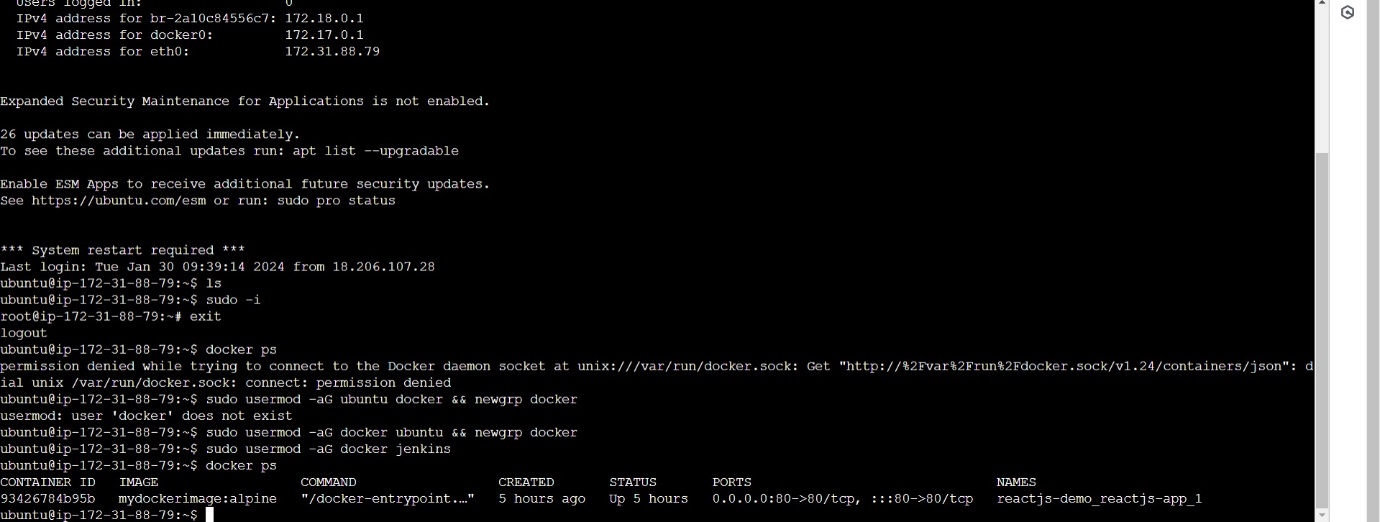
Docker Hub Username : reshma1215

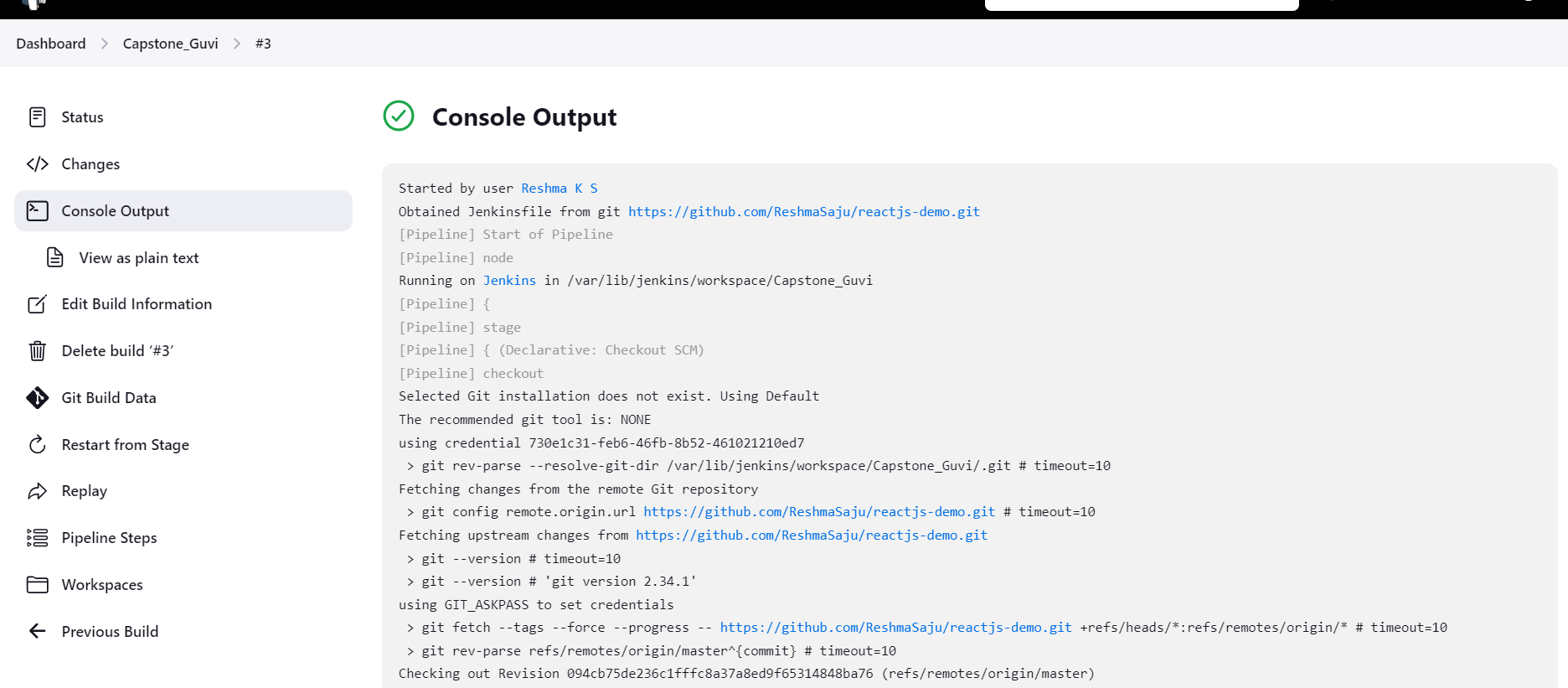
Password : Token from docker hub

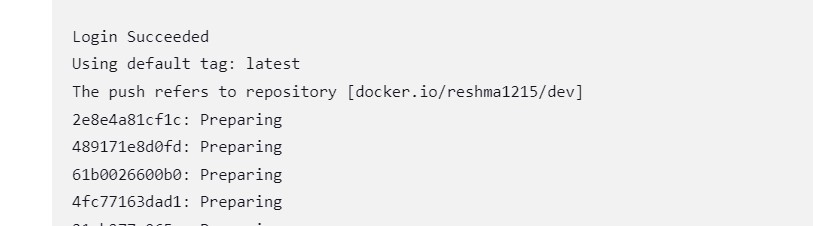


**sudo usermod -aG docker ubuntu && newgrp docker**

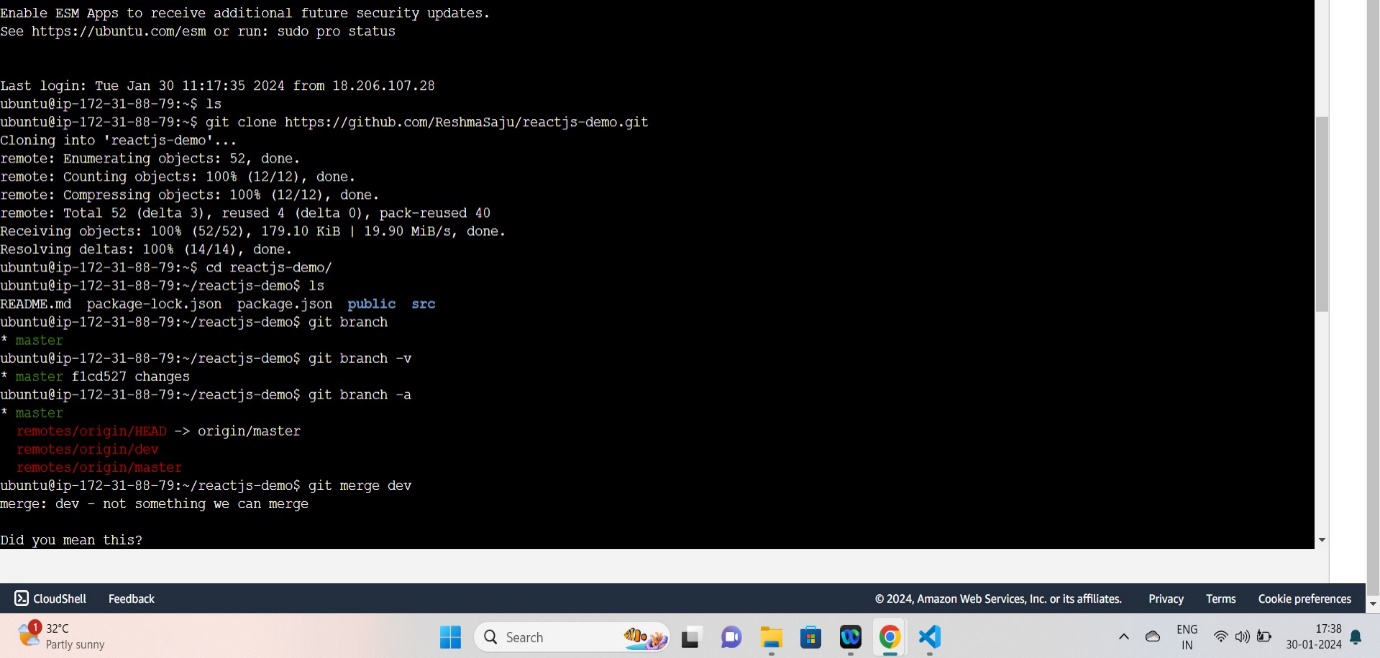
**sudo usermod - aG docker Jenkins**

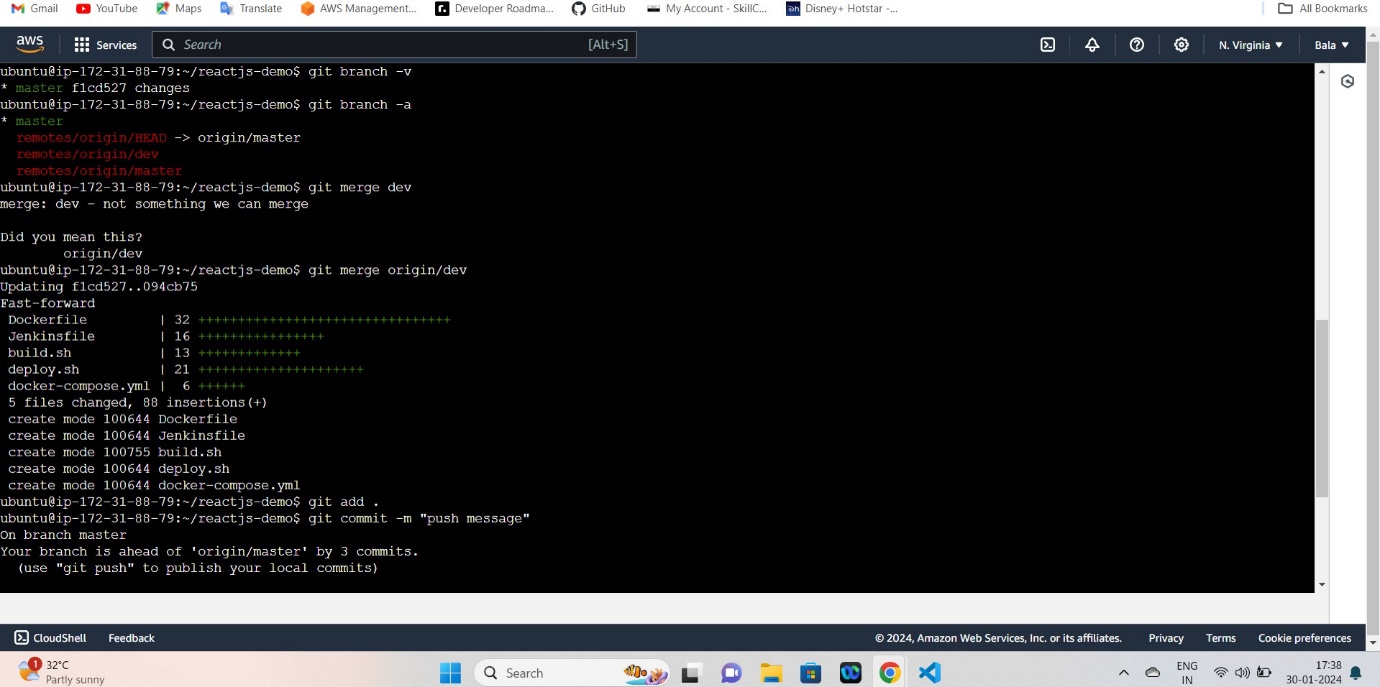






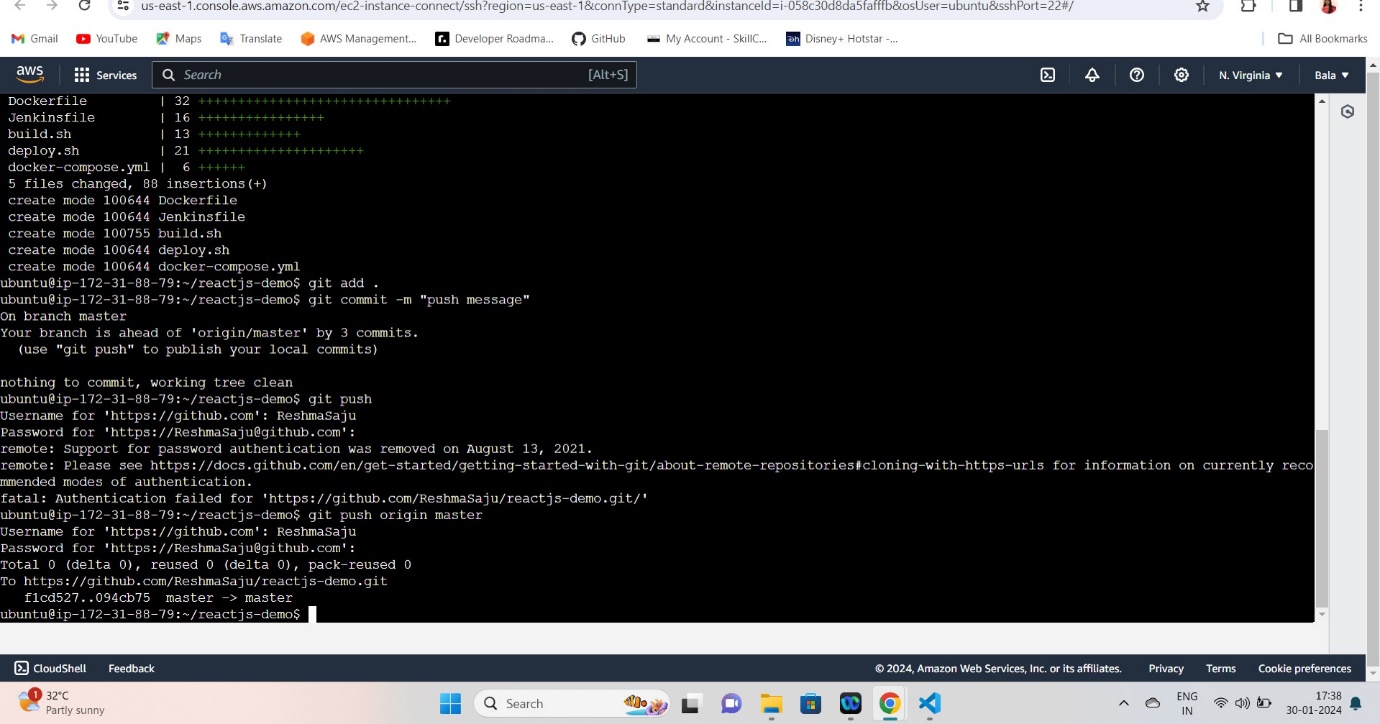
Next we are going to merge to dev branch





**User name of GitHub : ReshmaSaju**

**Password of GitHub : Token of GitHub**



**Now launch the second instance and update as below :**

**Sudo apt-get update**

Install docker

**sudo apt-get install docker.io**

pull dockerimage from dev repo

**docker pull reshma1215/dev:latest**

after giving above image it showed permission denied:

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=reshma1215%2Fdev&tag=latest": dial unix /var/run/docker.sock: connect: permission denied

sudo usermod -aG docker ubuntu

sudo usermod -aG docker ubuntu && newgrp docker

sudo service docker restart

ubuntu@ip-172-31-88-78:~$ sudo systemctl restart docker

ubuntu@ip-172-31-88-78:~$ docker info



ls -l /var/run/docker.sock

sudo chown root:docker /var/run/docker.sock

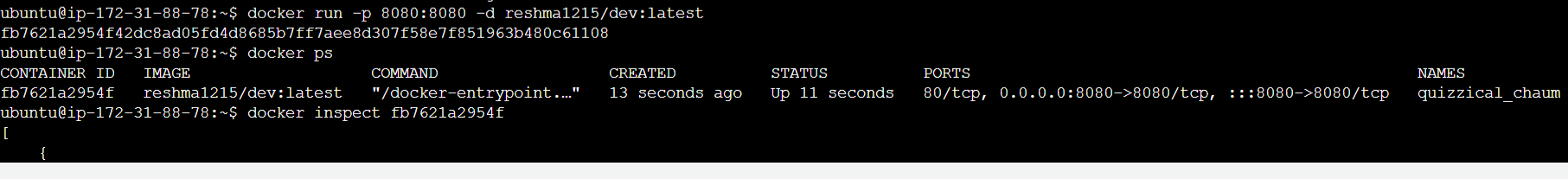
docker pull reshma1215/dev:latest

docker run -it --rm reshma1215/dev:latest

docker run -it --rm -p 8080:80 reshma1215/dev:latest

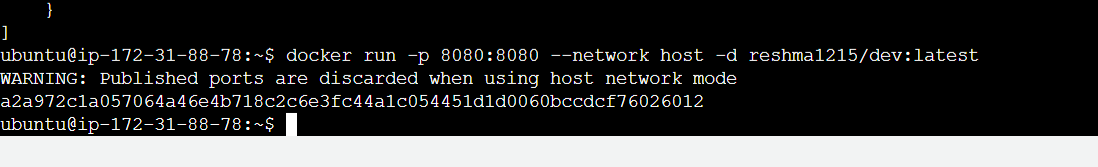
docker run -p 8080:8080 -d reshma1215/dev:latest

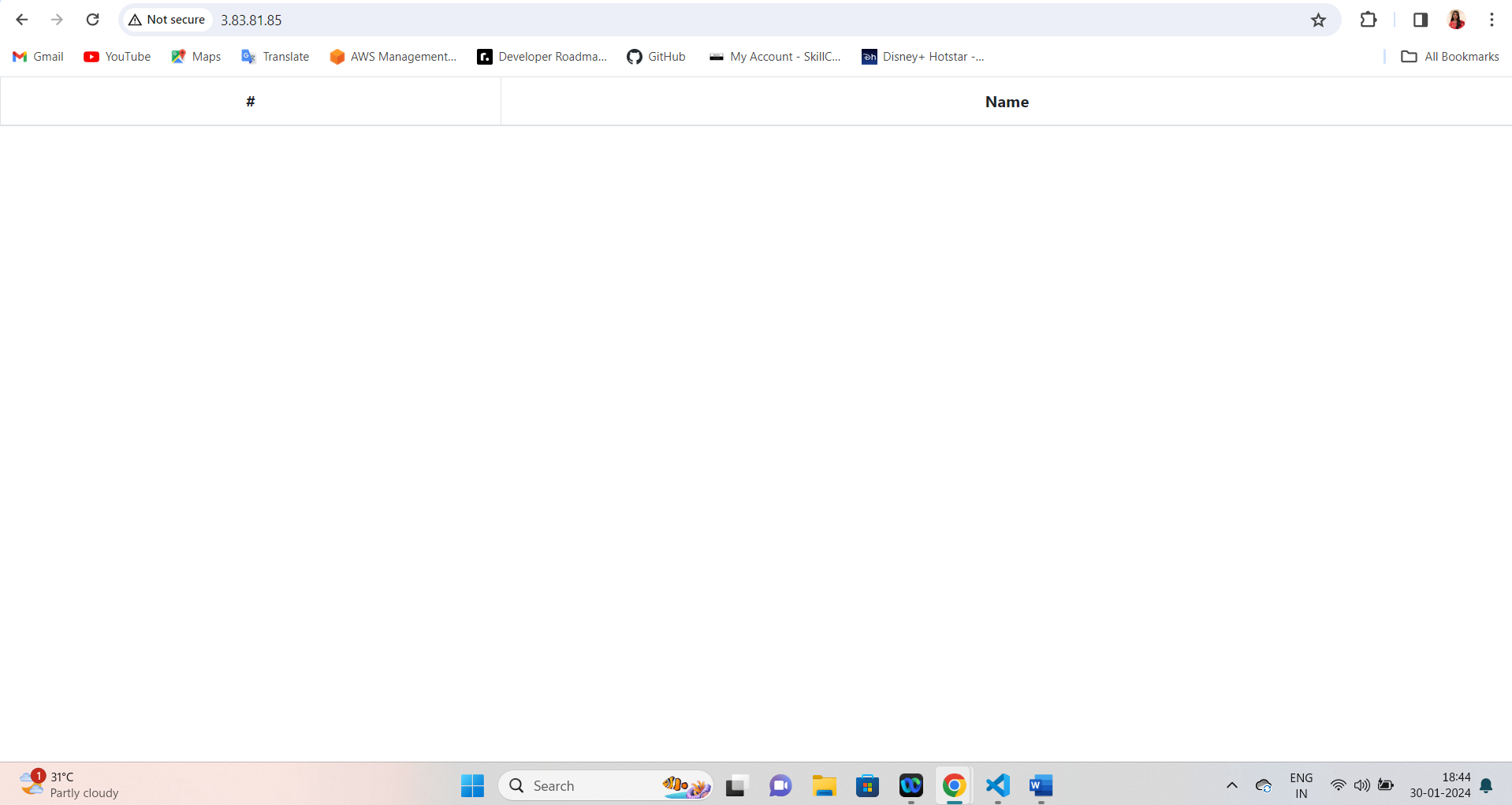
docker ps



docker inspect fb7621a2954f (image ID)

docker run -p 8080:8080 --network host -d reshma1215/dev:latest





**Monitoring setup**

* **sudo apt-get update**
* **GIT HUB Repo :** [**https://github.com/ReshmaSaju/Promethus-and-grafana.git**](https://github.com/ReshmaSaju/Promethus-and-grafana.git)
* git clone[**https://github.com/ReshmaSaju/Promethus-and-grafana.git**](https://github.com/ReshmaSaju/Promethus-and-grafana.git)
* **ls ----** we can see **Promethus-and-grafana**
* **cd Promethus-and-grafana**
* **ls and then give permission to all three files**

**chmod +x Prometheus.sh**

**chmod +x Grafana.sh**

**chmod +x node\_exporter.sh**

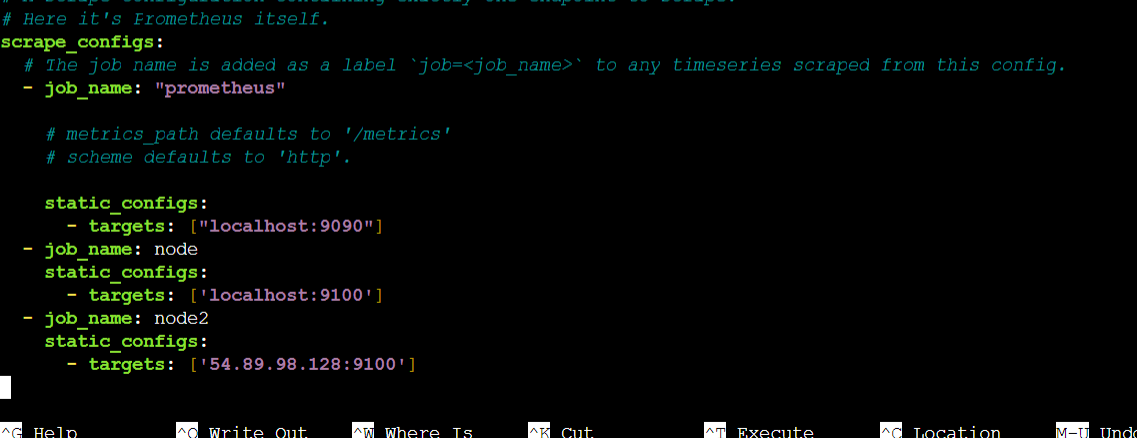
**Now build it :**

**./ Prometheus.sh**

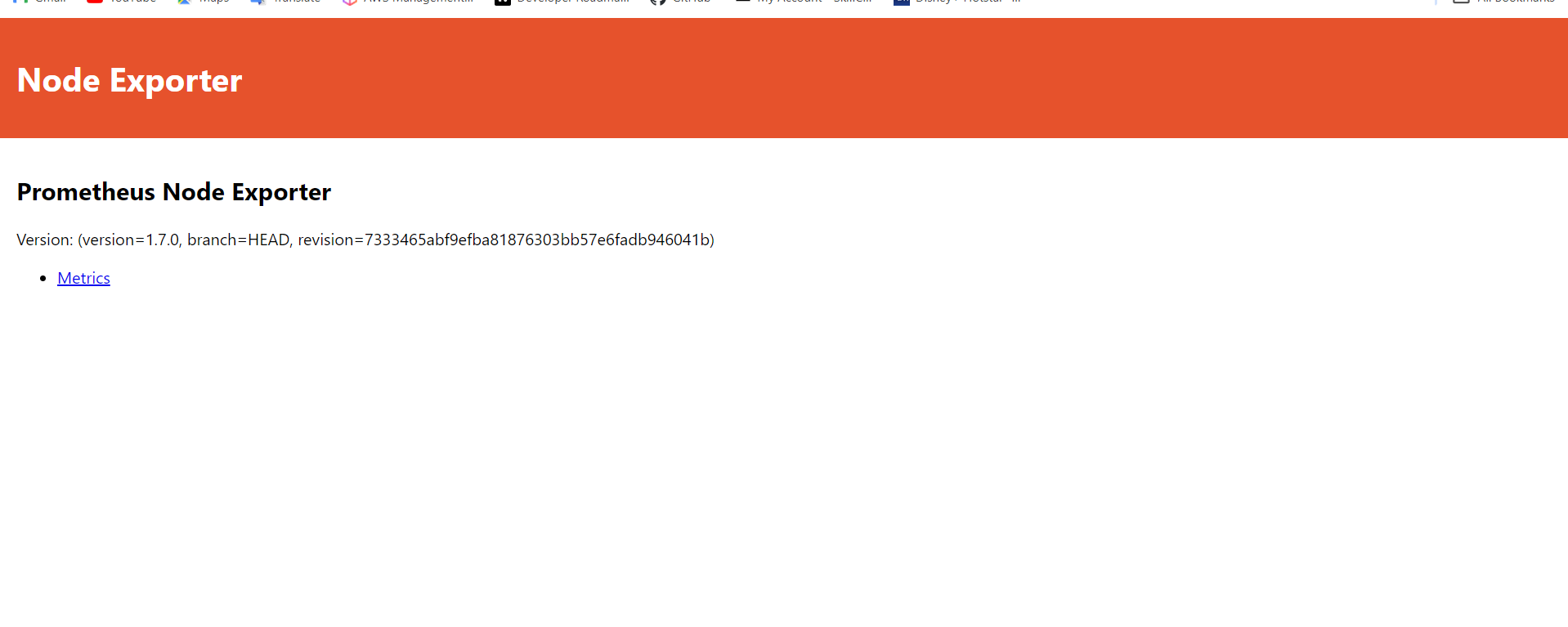
**./ Grafana.sh**

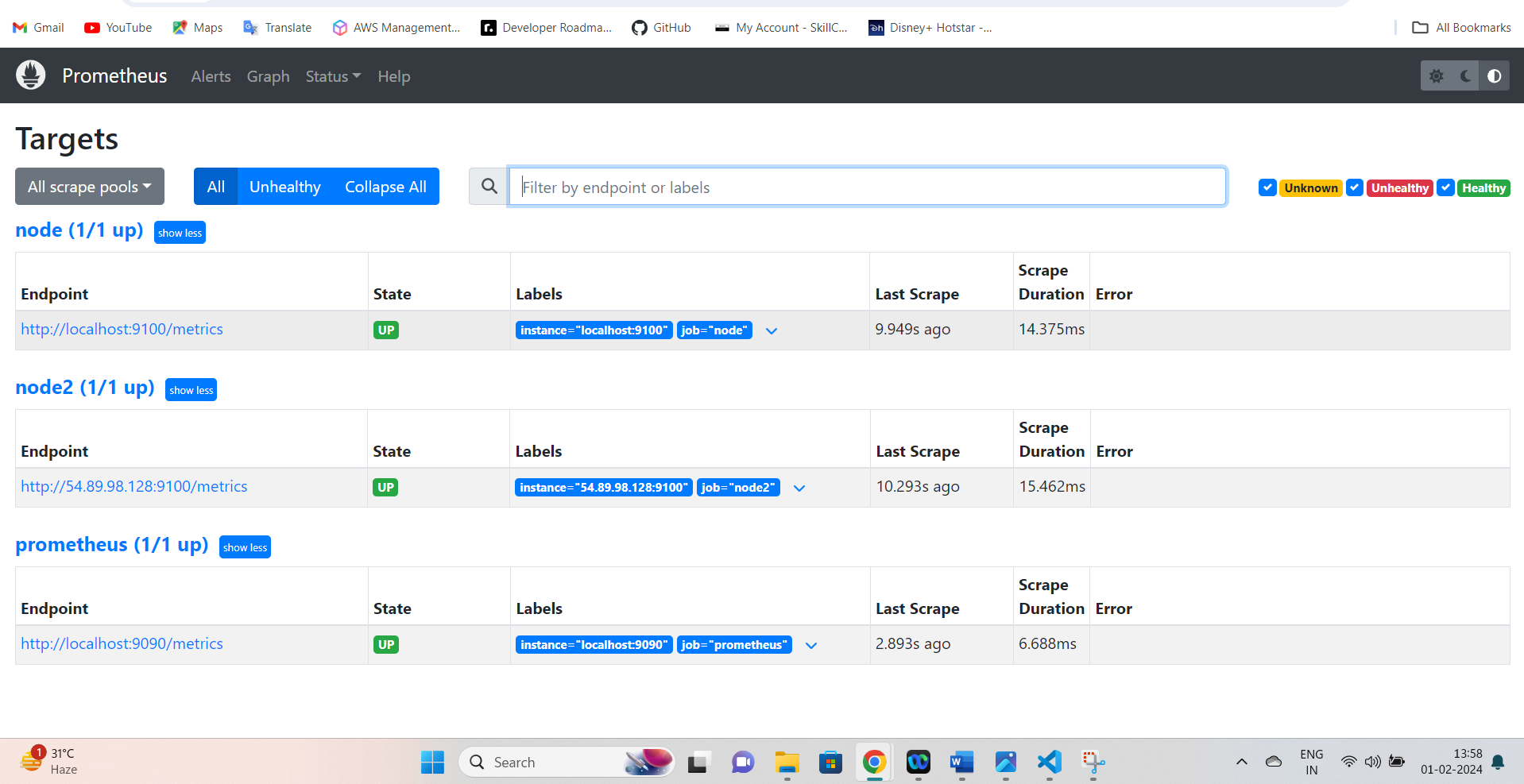
**./ node\_exporter.sh**

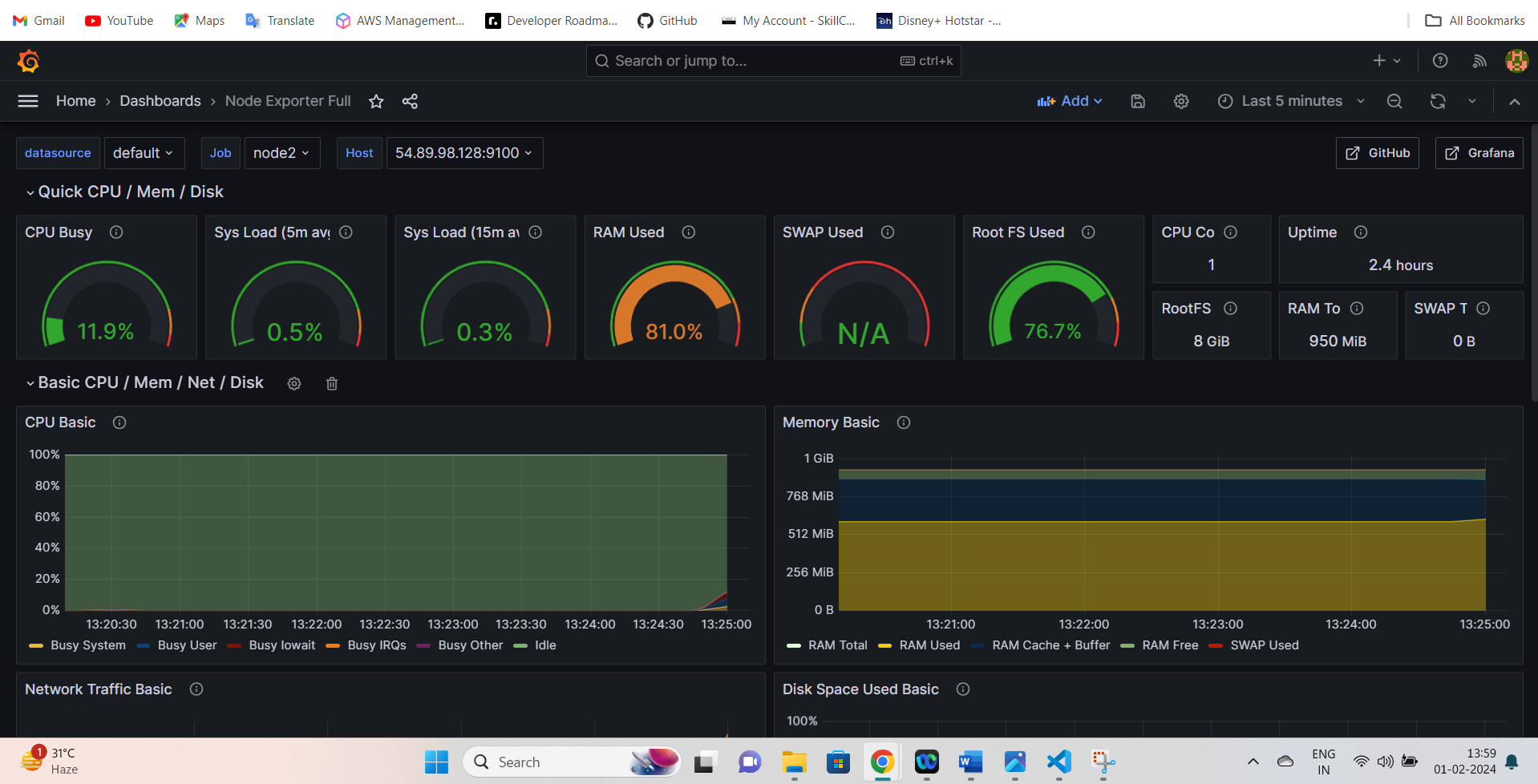
* **service prometheus restart**
* **service prometheus status**
* **systemctl status node-exporter**
* **Copy public ip address :9100 --- Node exporter will be working.**
* **Cd ..**
* **Cd**
* **Cd /etc/Prometheus/**
* **ls ----- now we can see Prometheus.yml file**
* **now nano Prometheus.yml**

****

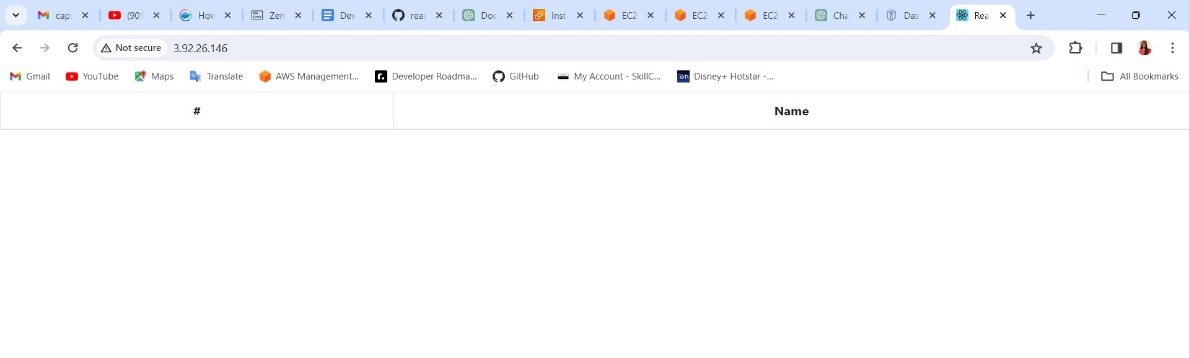
* **service Prometheus restart**

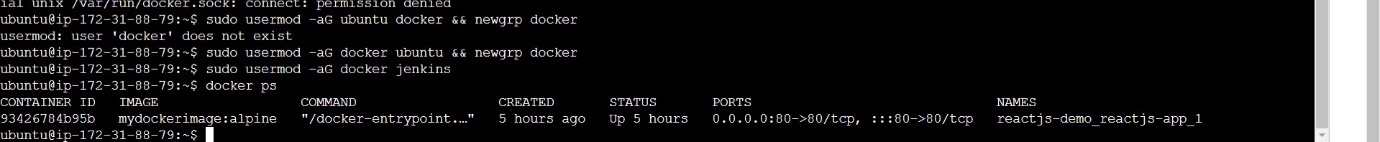
****

****

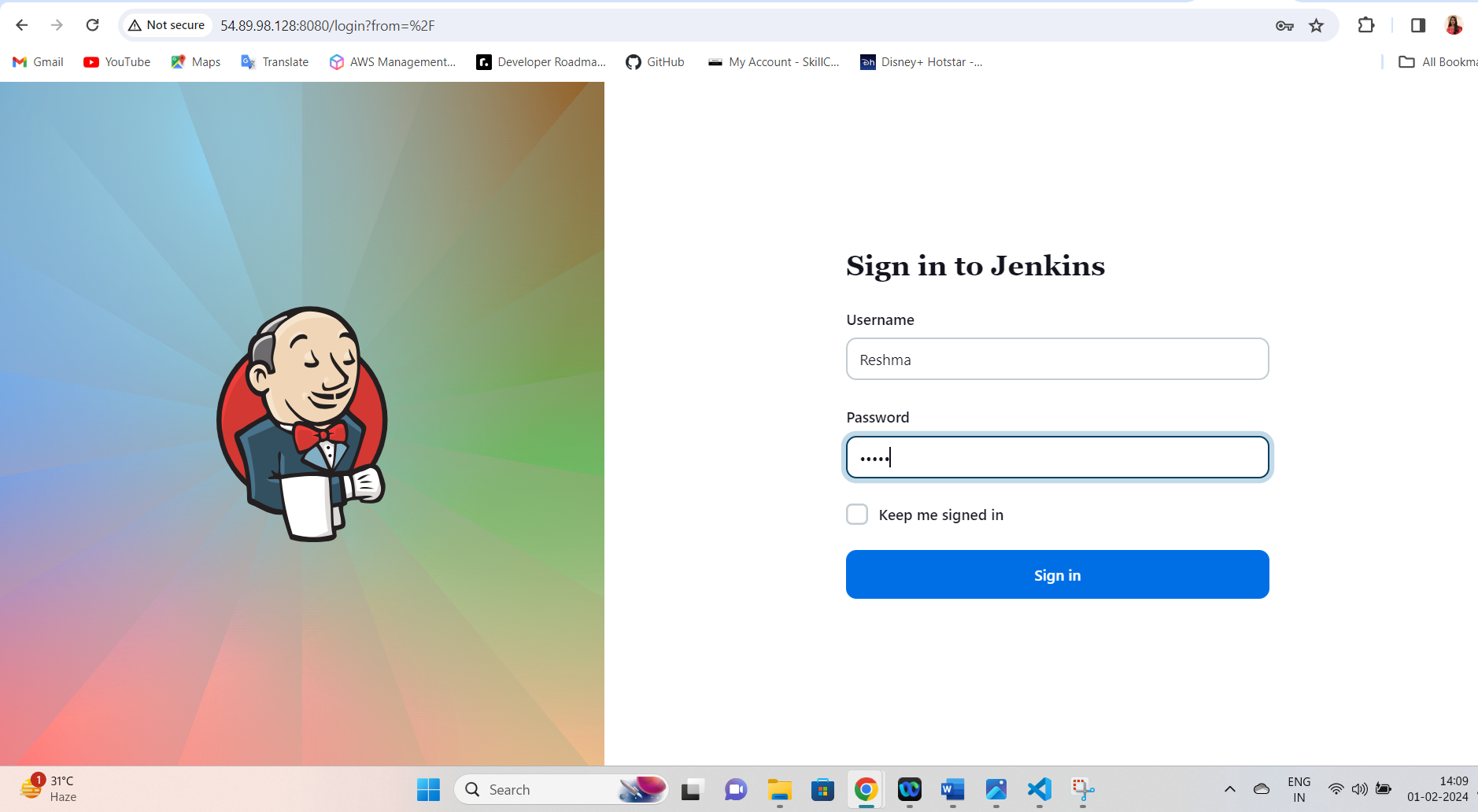
****

**Result :**

1. **Git Hub URL :** [**https://github.com/ReshmaSaju/reactjs-demo.git**](https://github.com/ReshmaSaju/reactjs-demo.git)
2. **Deployed site URL** 
3. Docker images



1. Jenkins login page

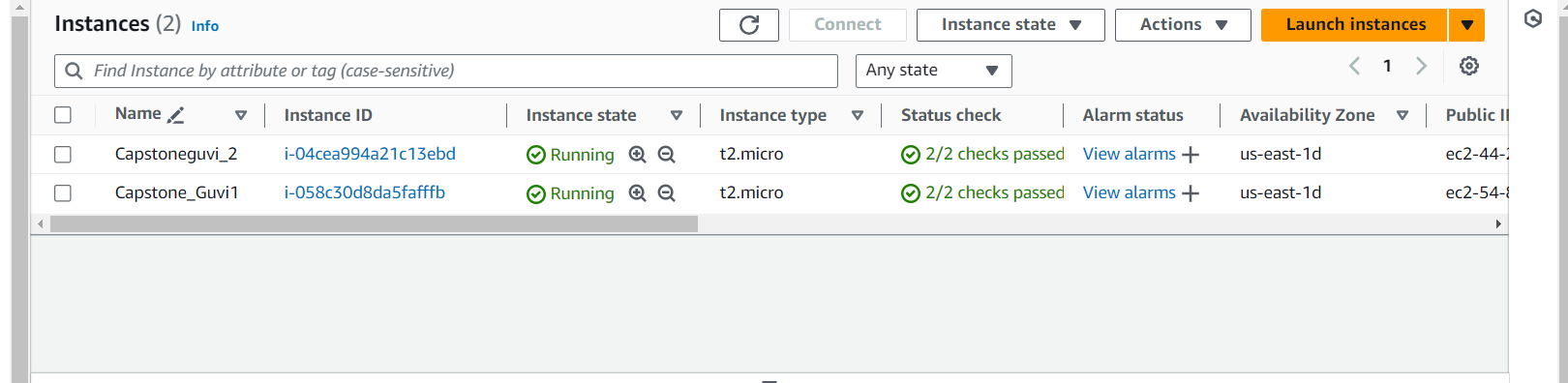


1. Execute setup command :

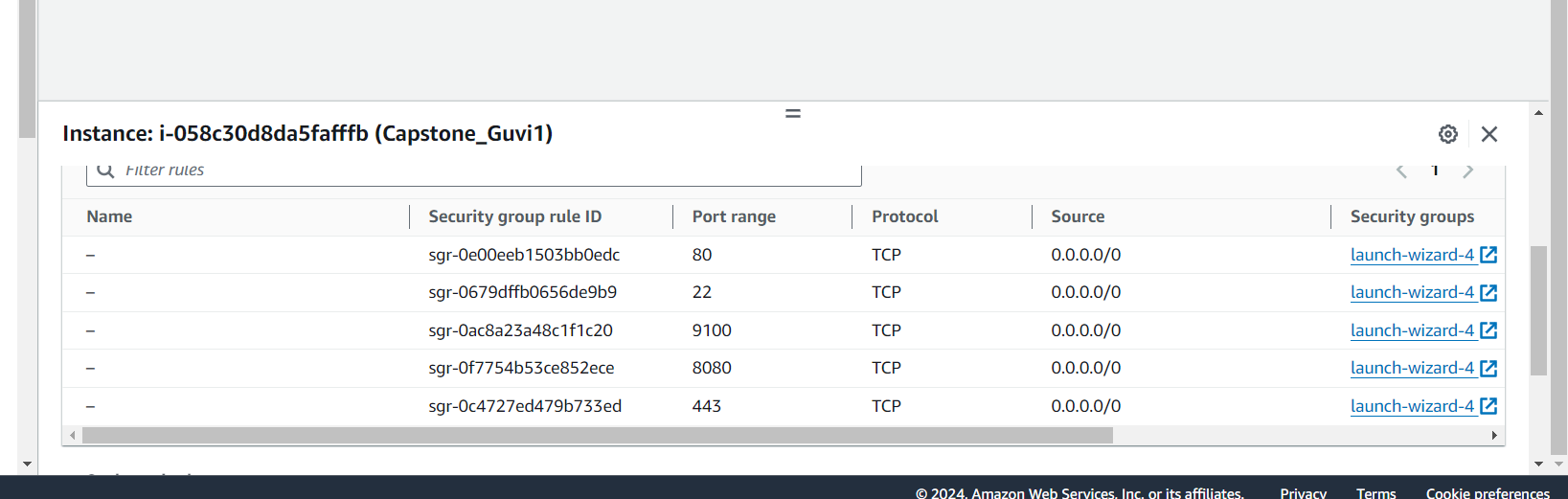
**Sudo usermod -a G docker ubuntu && newgrp docker**

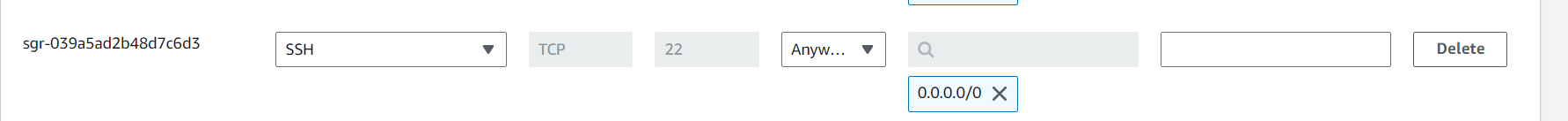
**Sudo usermod -a G docker Jenkins**

1. AWS EC2 console and SG Configuration screenshots :

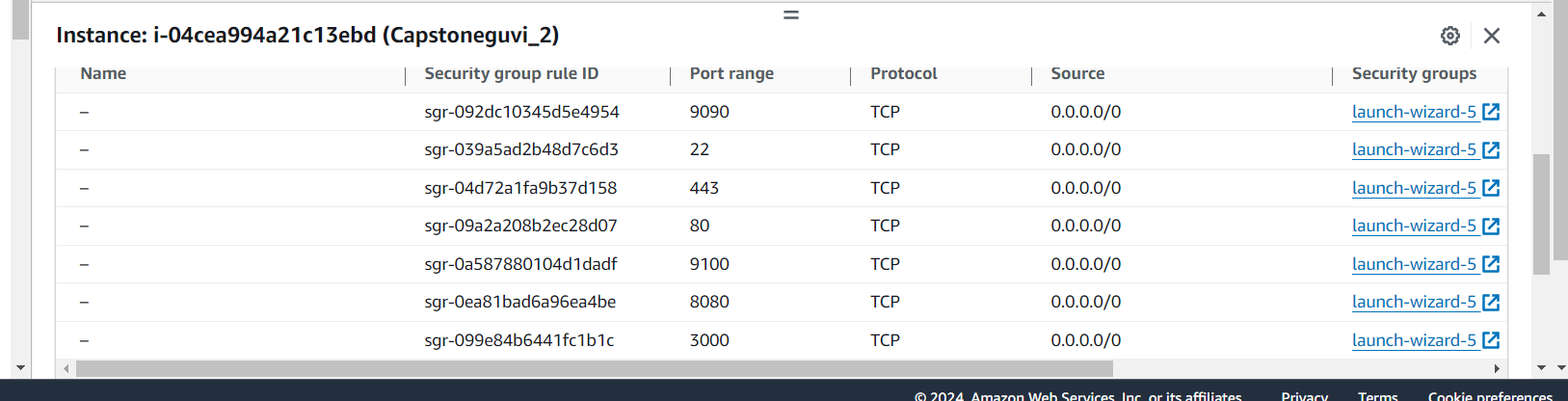


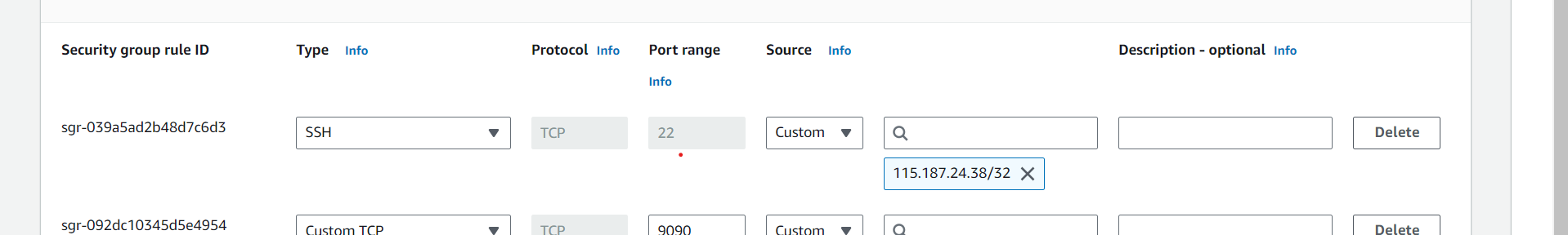
Capstone\_guvi1



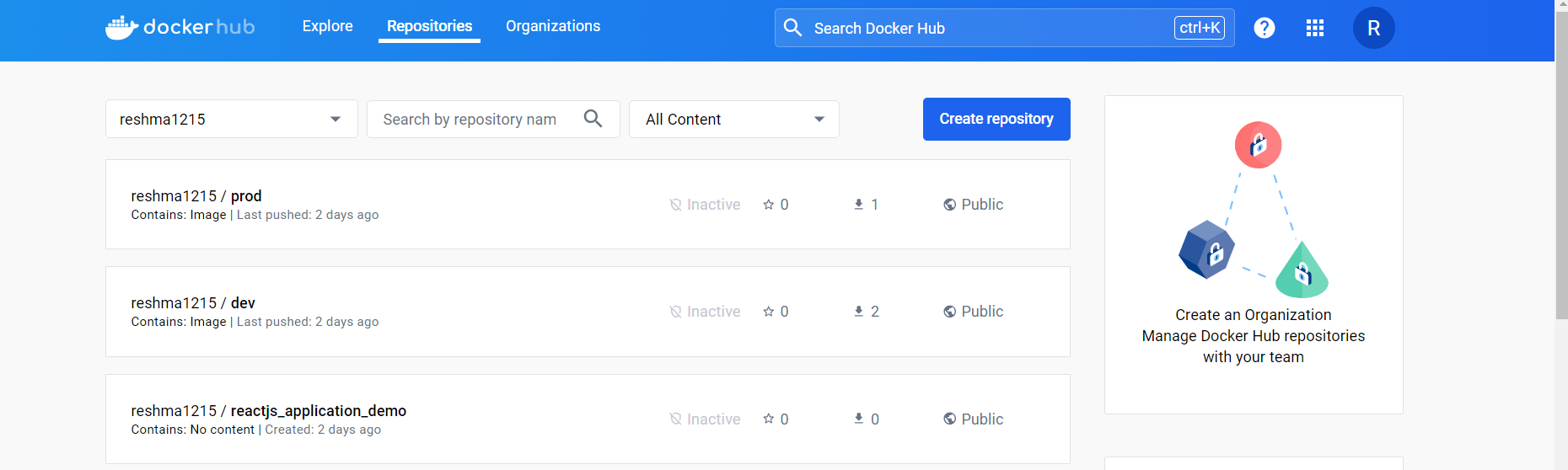


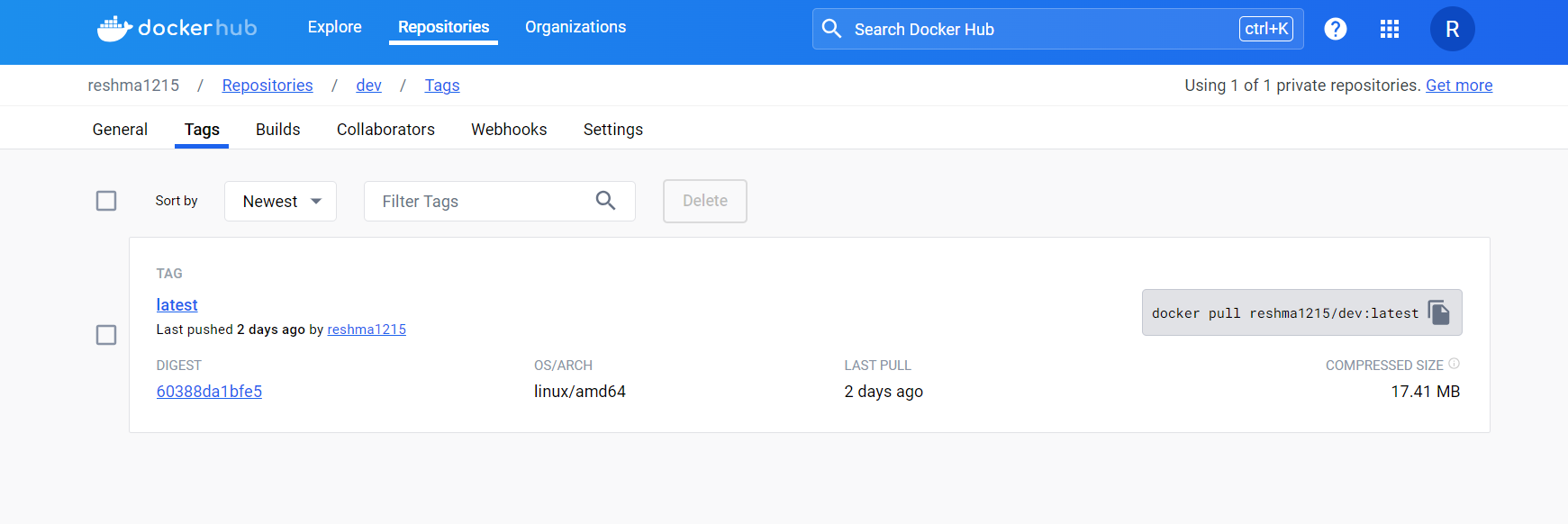
Capstone\_guvi 2

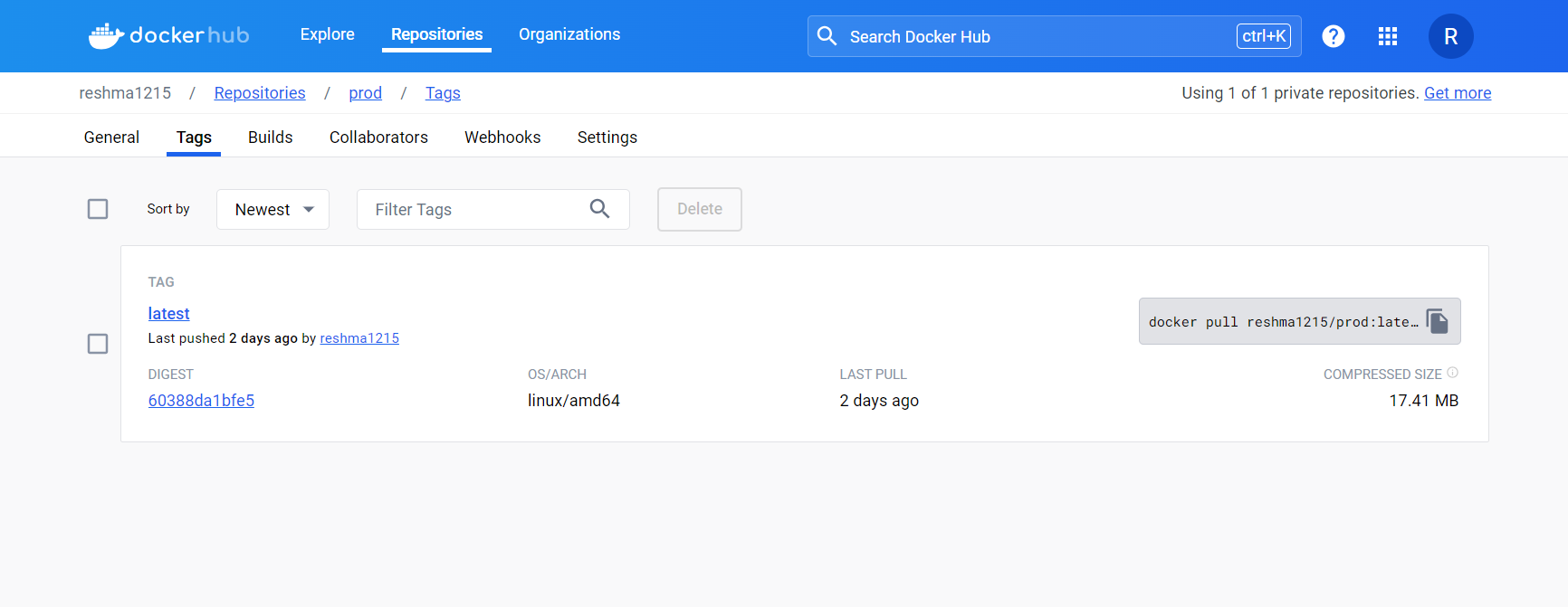




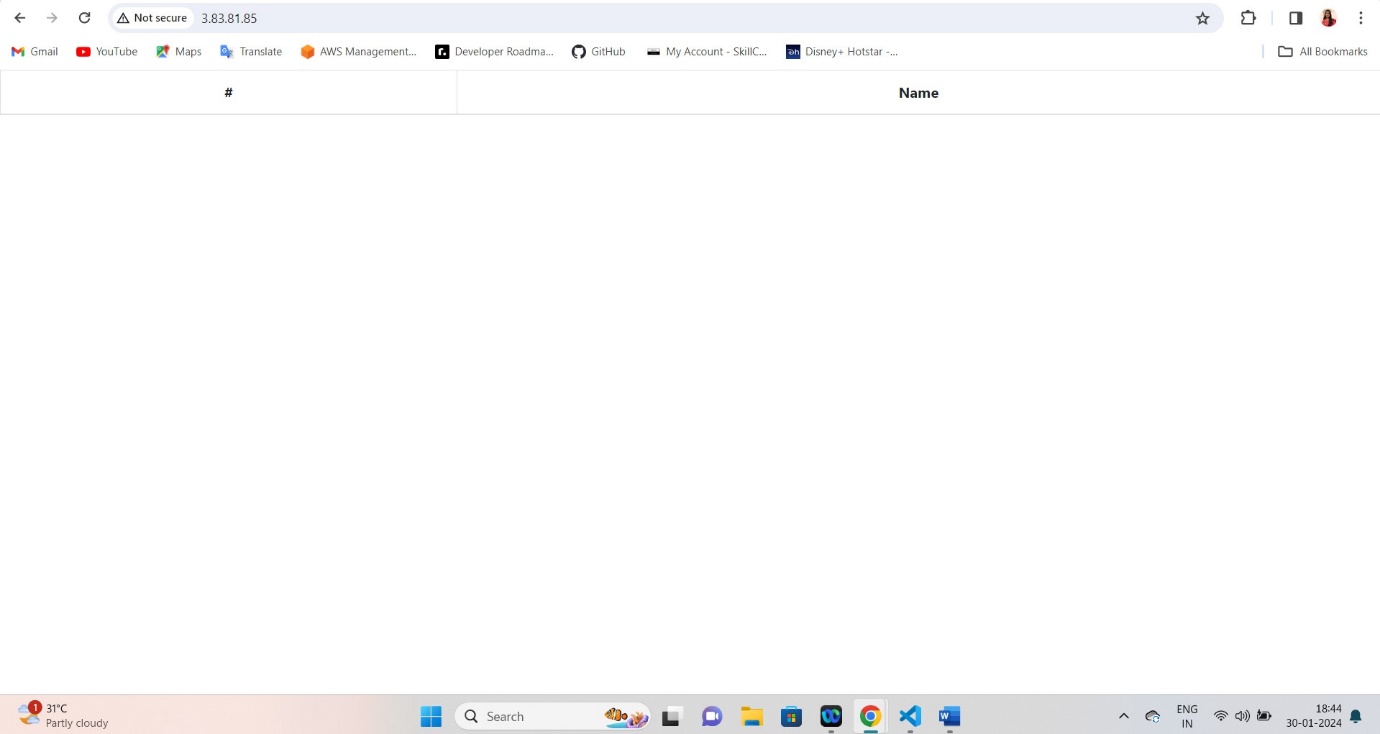
1. **Docker Hub repo with image tags**



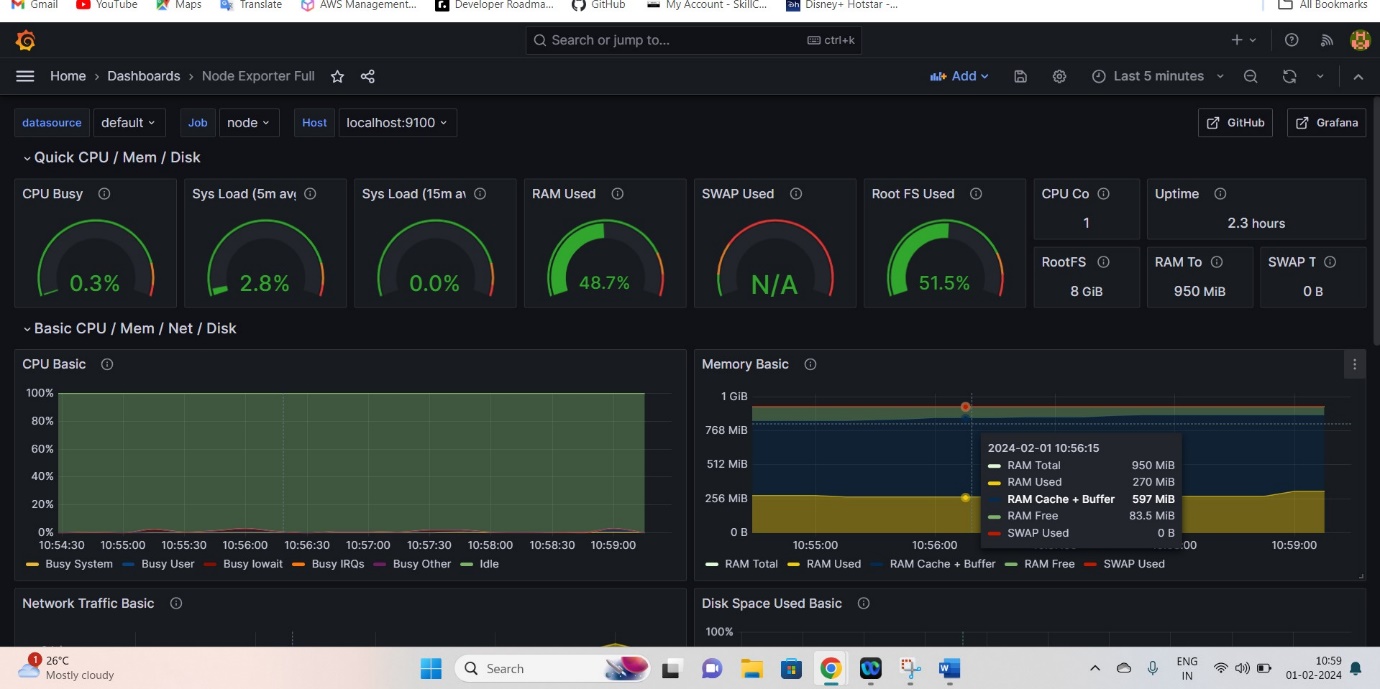




1. Deployed site page:



1. Monitoring health checkup status



Finally Deployed site and status

URL 3.93.217.131: 3000

