**Application Deployment**

(Deploy the given React application to a production ready state.)

**AWS:**

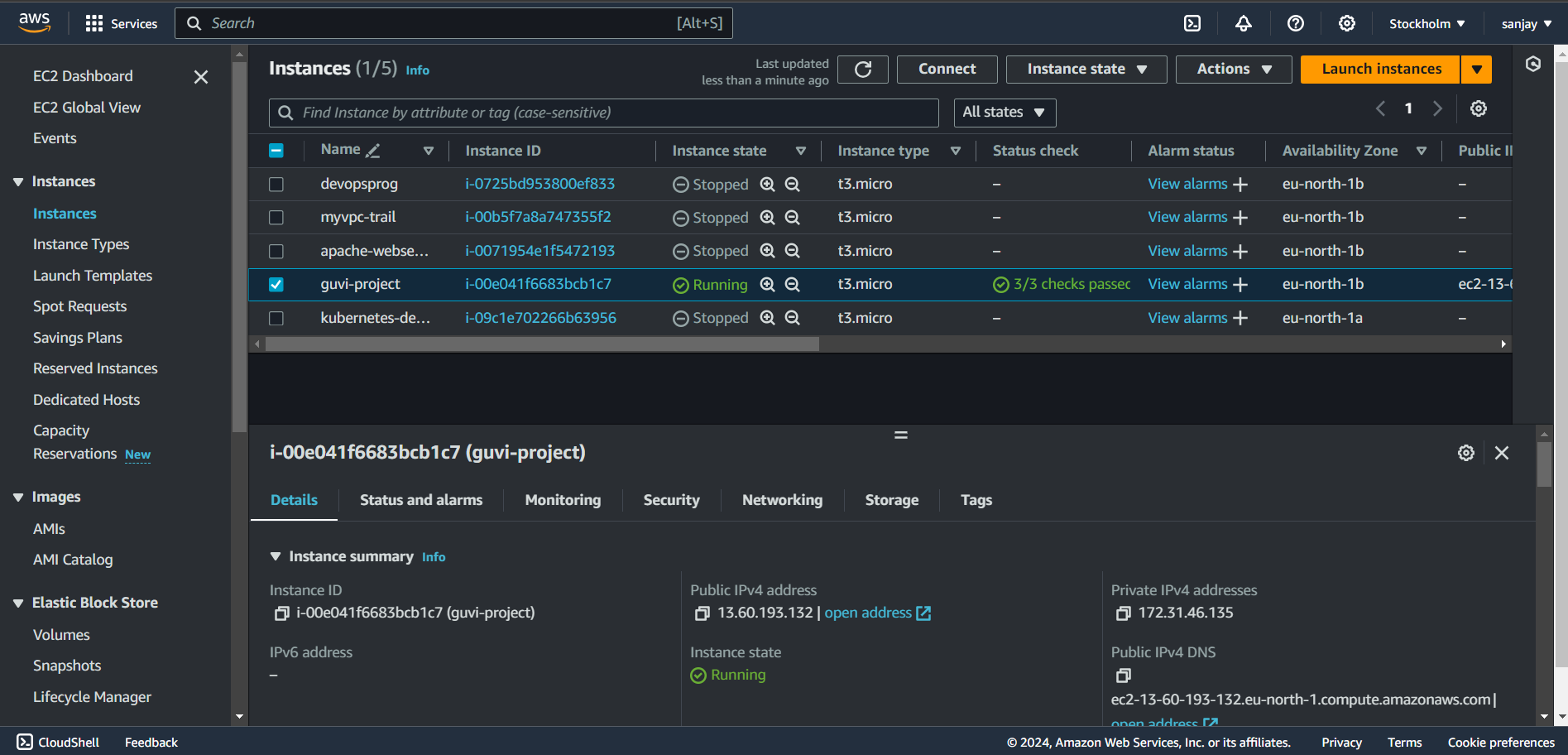
  Launched an Ubuntu, T2.micro instance and configured Security Group as below:

**Configure Inbound Rules:**

* **Rule 1: Allow SSH Access from Your IP Address:** Login to server can should be made only from your Ip address
  + Type: SSH, Protocol: TCP, Port Range: 22
  + Source: My IP (Use the current IP of your machine)
* **Rule 2: Allow HTTP/HTTPS Access from Anywhere:** Whoever has the IP address can access the application
  + Type: HTTP, Protocol: TCP, Port Range: 80
  + Source: Anywhere (0.0.0.0/0)
* Rest all settings have remained the same or proceeded with the default settings and Launched the EC2 instance naming “PROJECT”.

A screenshot of a computer

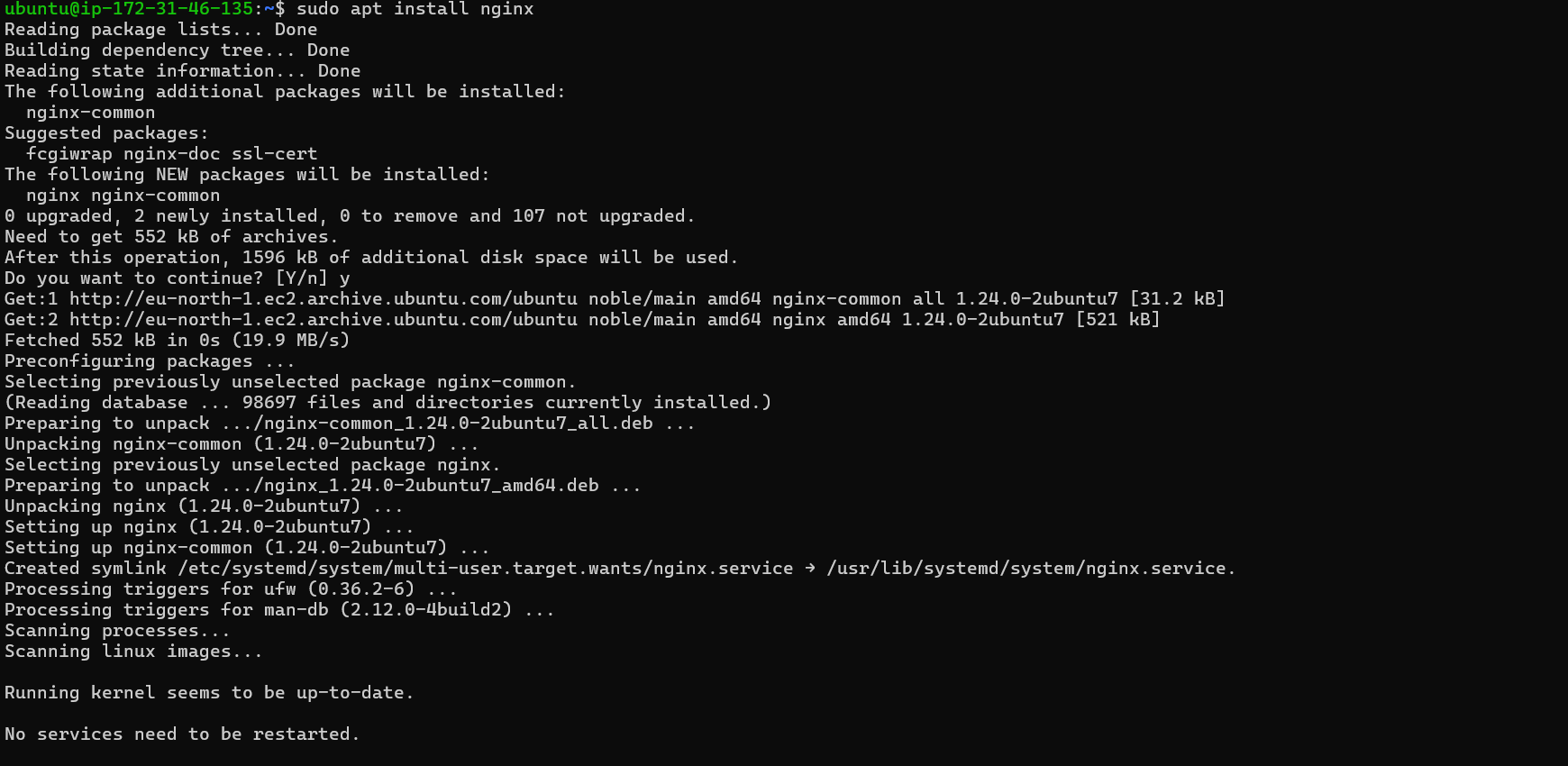
Description automatically generated

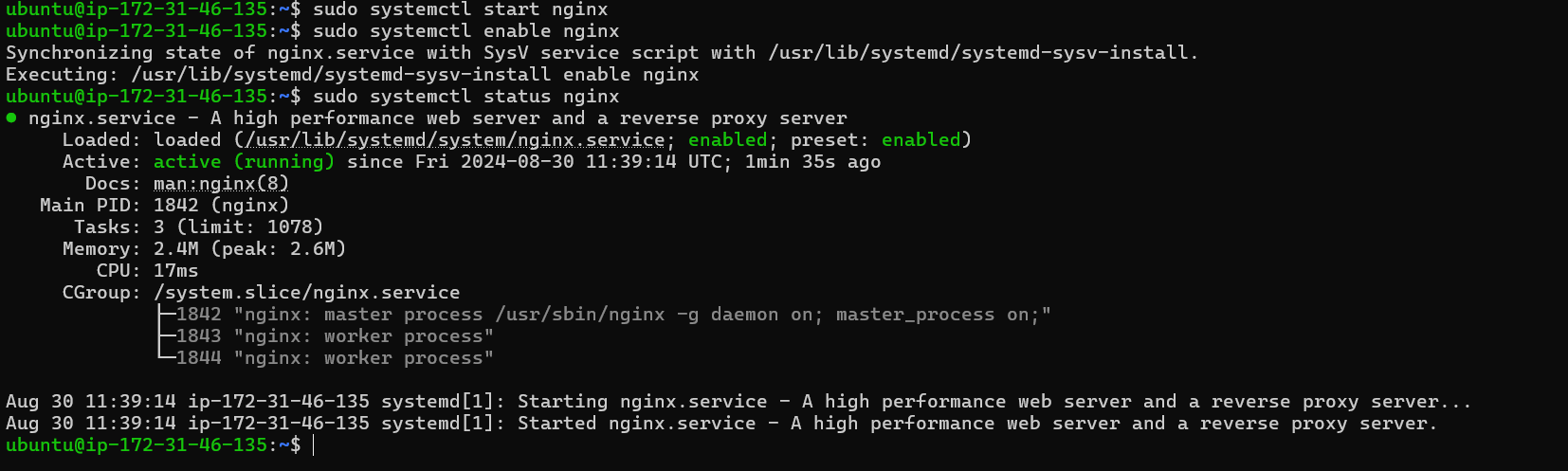


Once the Instance is created and running, Launched the terminal and to execute the given application we need Nginx application on the machine. Here are the commands used to update and install Nginx.

|  |
| --- |
| sudo apt update sudo apt install nginx -y sudo systemctl start nginx sudo systemctl enable nginx sudo systemctl status nginx |

* As we have to run the application on port 80 I have made changes to the Nginx configuration file to listen to port 3000 as default for Nginx.



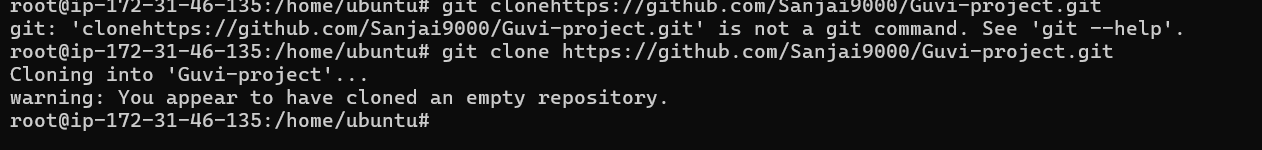


**Version Control:**

* Prerequisites for creating git ignore and docker ignore files are to install Git and Docker and you should have access to the repository and necessary permissions. Below is Git install & version

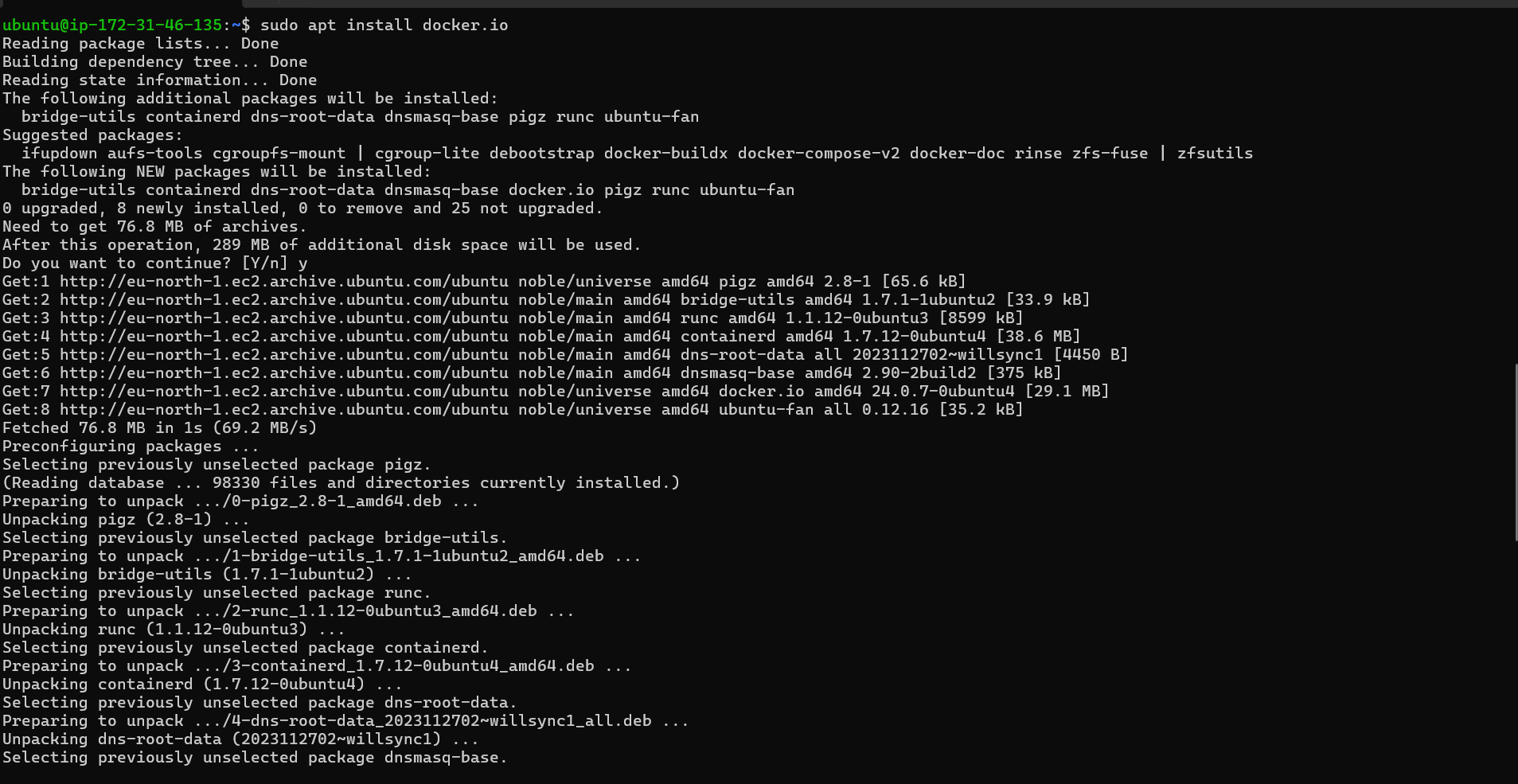
|  |
| --- |
| sudo apt-get update sudo apt-get install git git config --global user.name "Your Name" git config --global user.email "your.email@example.com" |

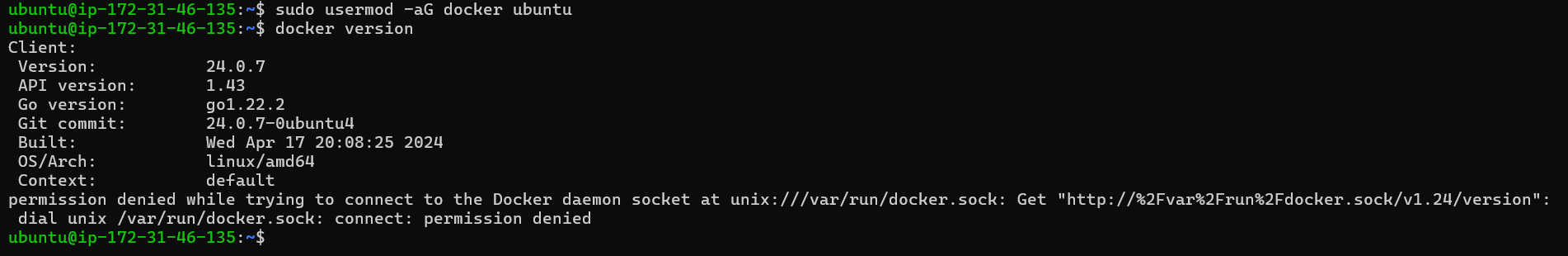
Go to GitHub and create a new repository.



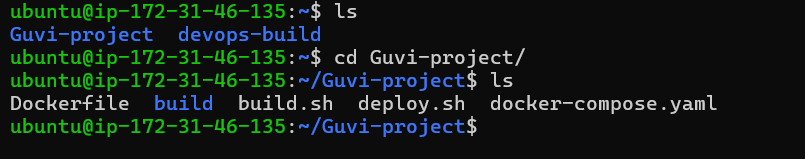
* Docker install & version:

|  |
| --- |
| sudo apt update sudo apt install docker-ce -y sudo systemctl start docker sudo systemctl enable docker sudo docker –version sudo usermod -aG docker $USER docker login |

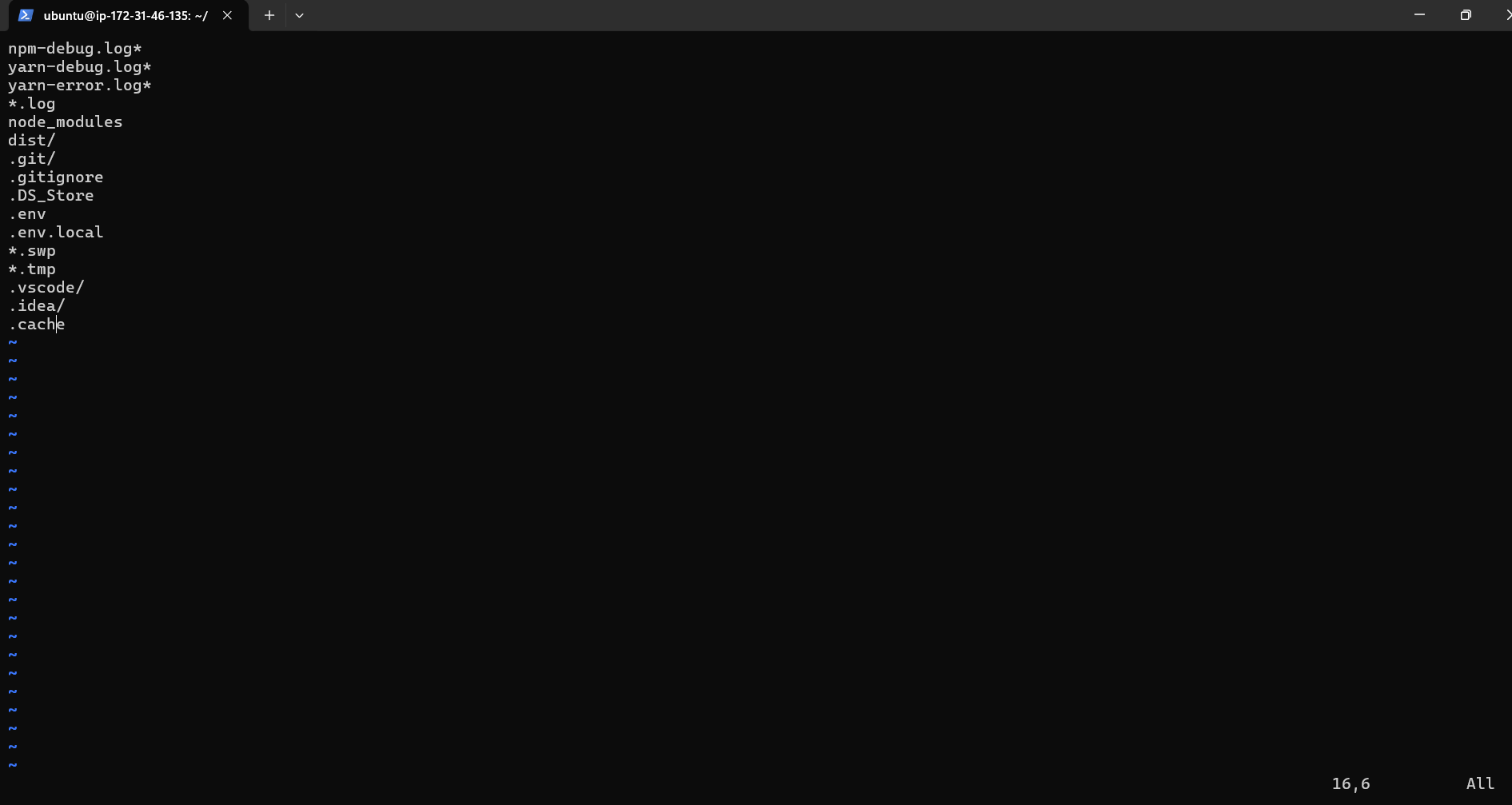




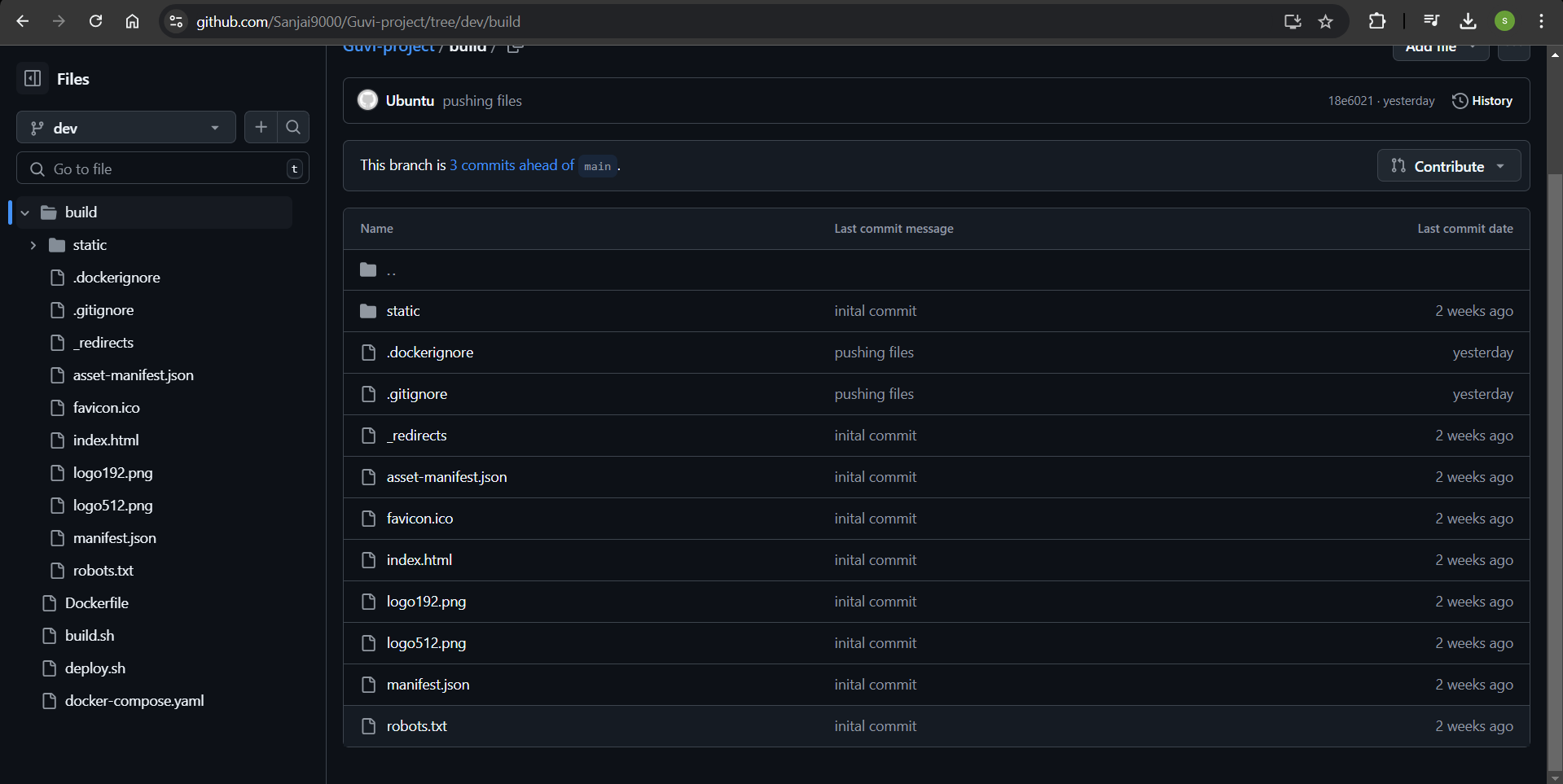
* Now open the terminal and Clone the given Repository with the build application. And navigate to the directory and located all the files.



* Now firstly creating the Git ignore file and the Docker ignore file .After creating both the files pushed the given repository to my newly created repository with the same name ‘devops-build’ to new ‘dev’ branch.

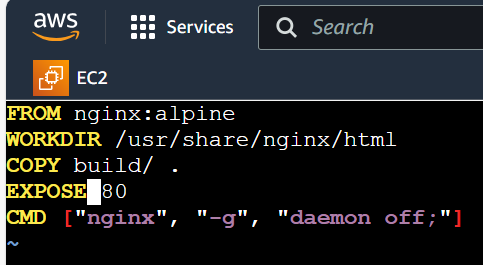


* Here we can see the ‘.gitignore’ & ‘.dockerignore’ files which have been created.



**Docker:**

* Dockerize the application by creating the Dockerfile.



* To create a Docker-compose file for the created image firstly we need to install Docker-compose:

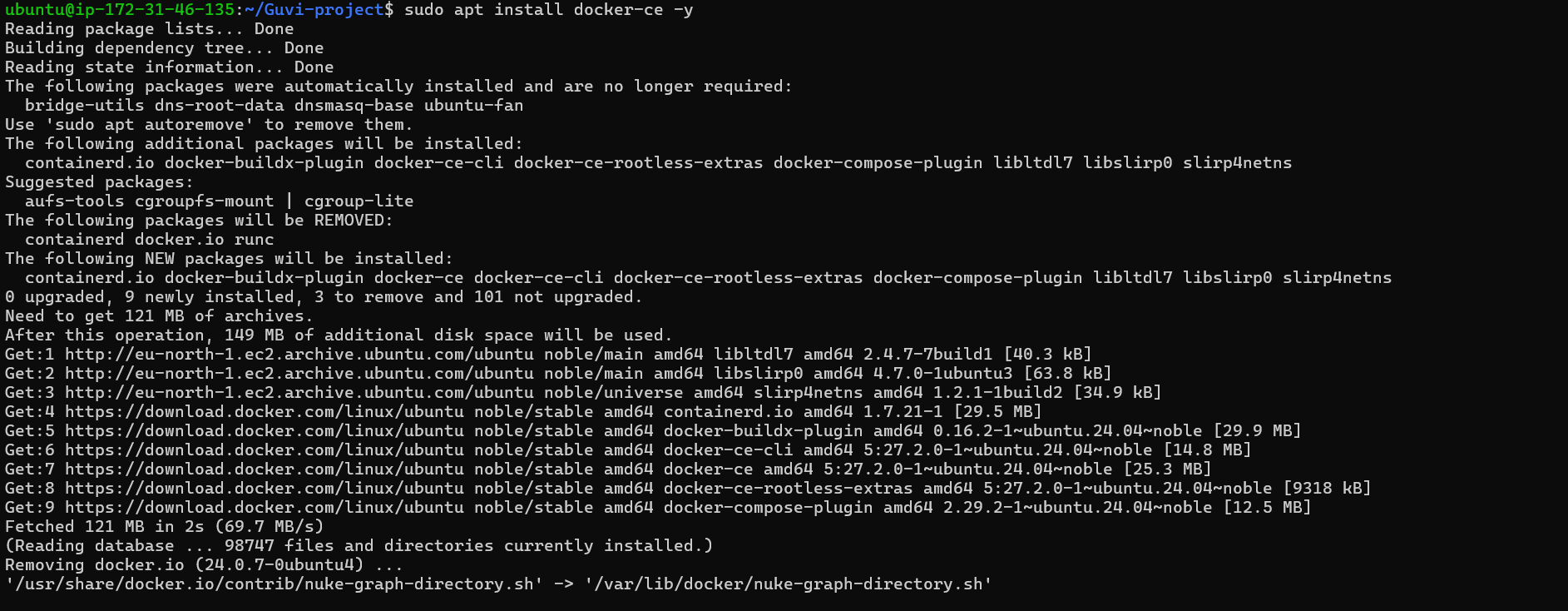
|  |
| --- |
| sudo apt update sudo apt install -y docker-compose |

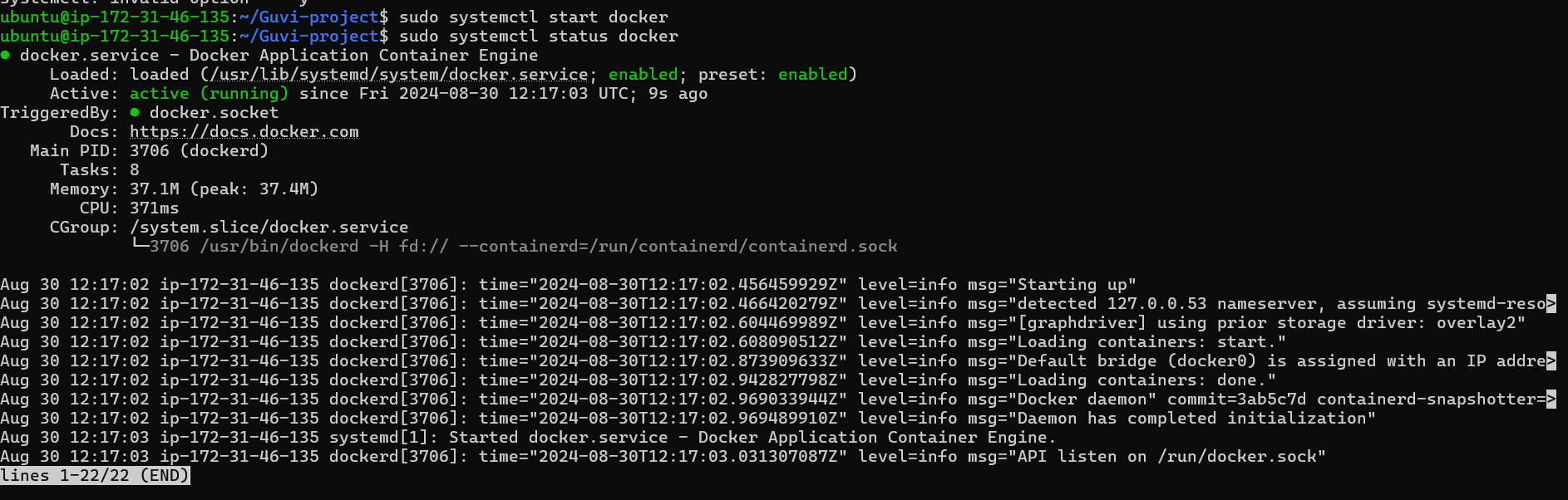
* Apply executable permissions to the Docker Compose binary:

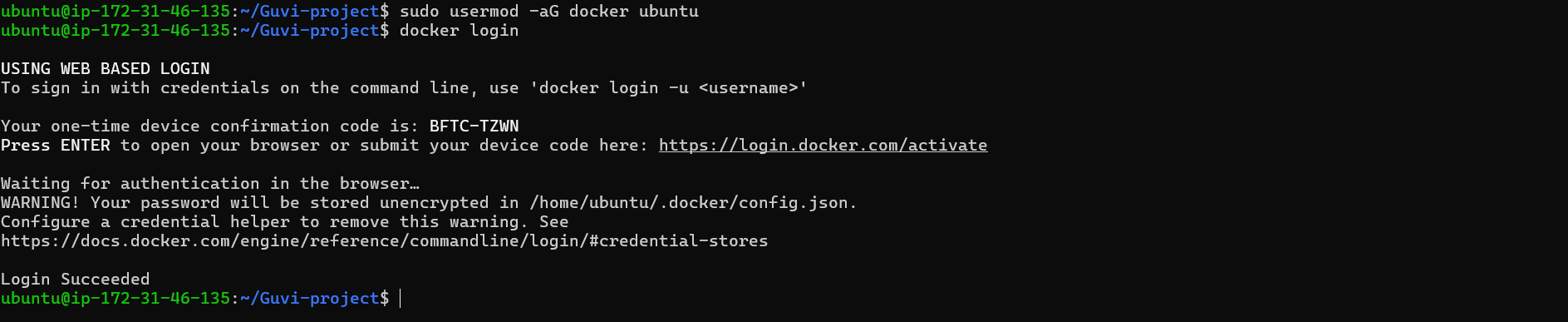
|  |
| --- |
| sudo chmod +x /usr/local/bin/docker-compose |

* Verify the installation:

|  |
| --- |
| docker-compose –version |







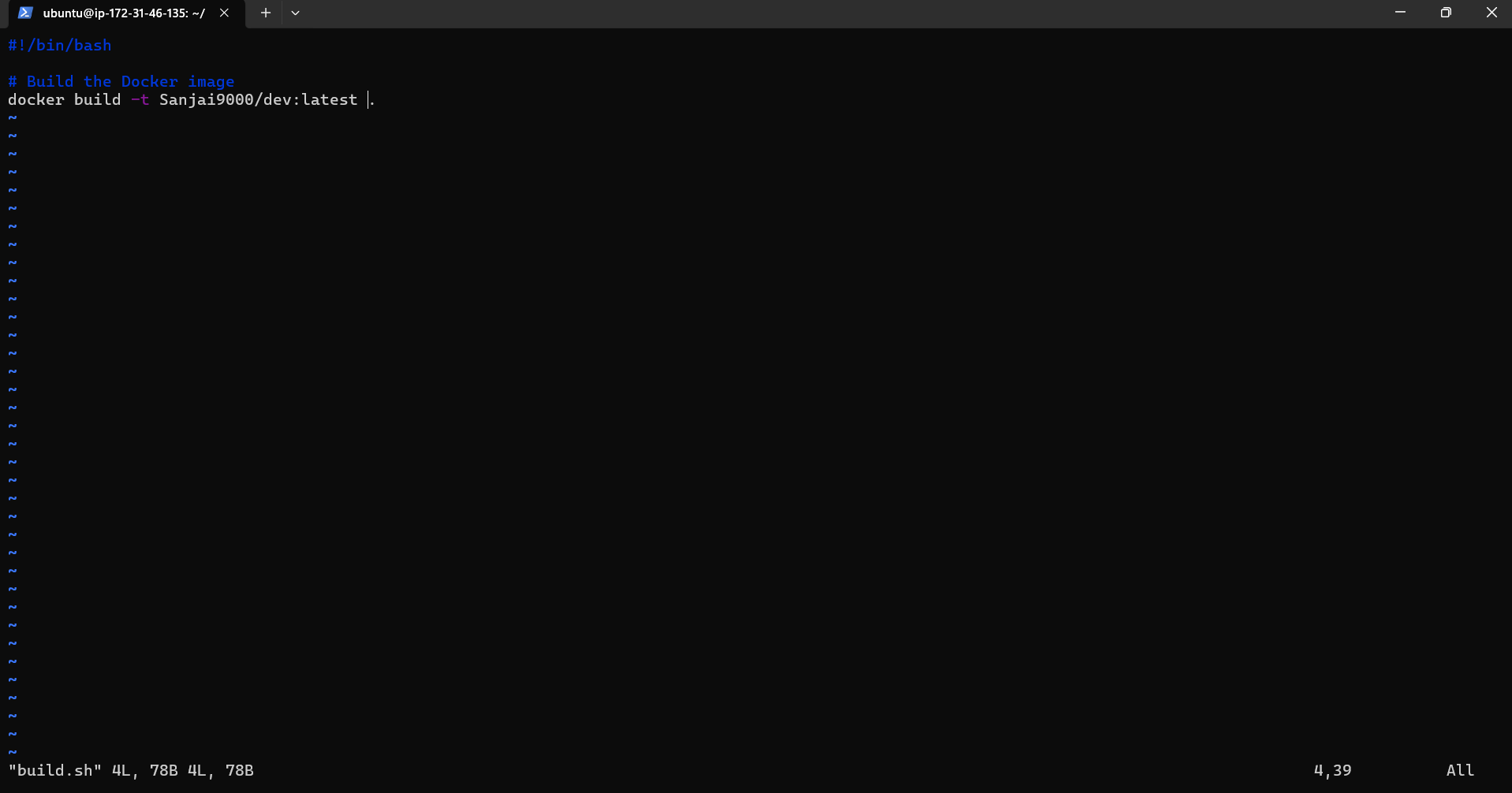
* Below is the docker-compose.yaml file.



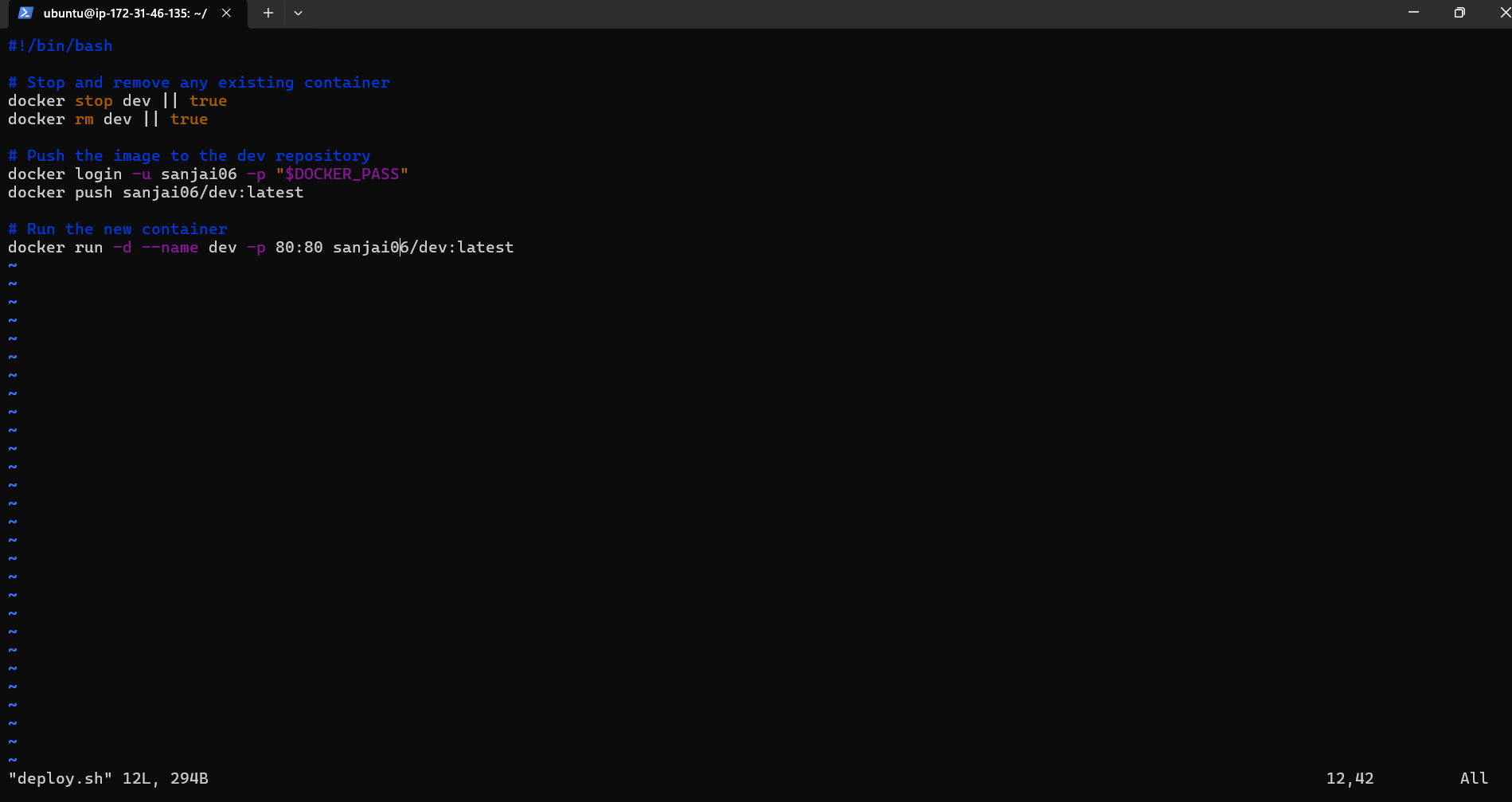
**Bash Scripting:**

Here are the created 2 scripts

* build.sh - for building docker images and given chmod +x build.sh to execute the file we can use the command ‘./build.sh’

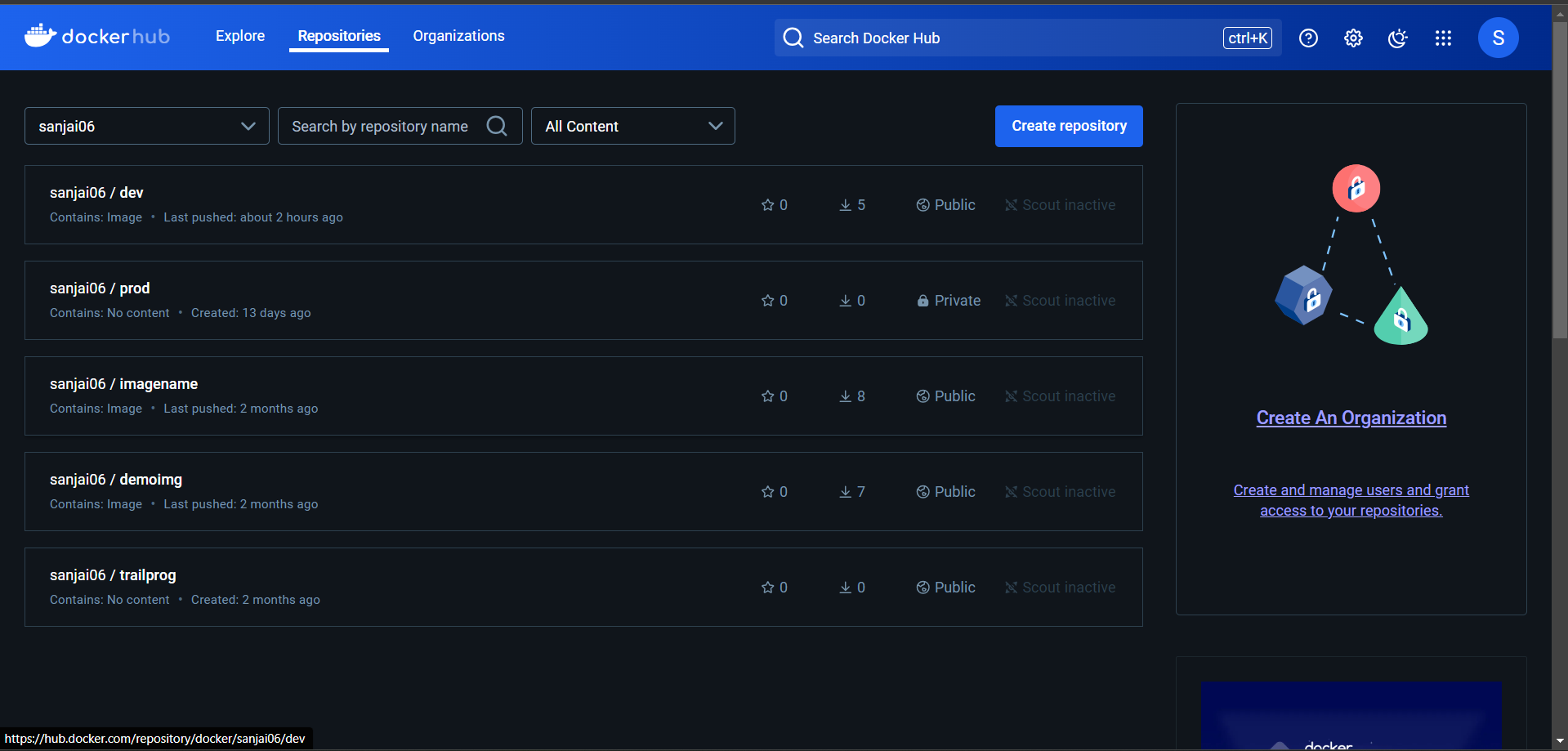


* deploy.sh - for deploying the image to the server , chmod +x deploy.sh and ./deploy.sh



**Docker hub:**

* Created 2  repos “dev” and “prod” to push images with the rules “Prod” repo must be private and “dev” repo can be public.



**Jenkins:**

* Install and configure jenkins build step as per needs to build, push and deploy the application:

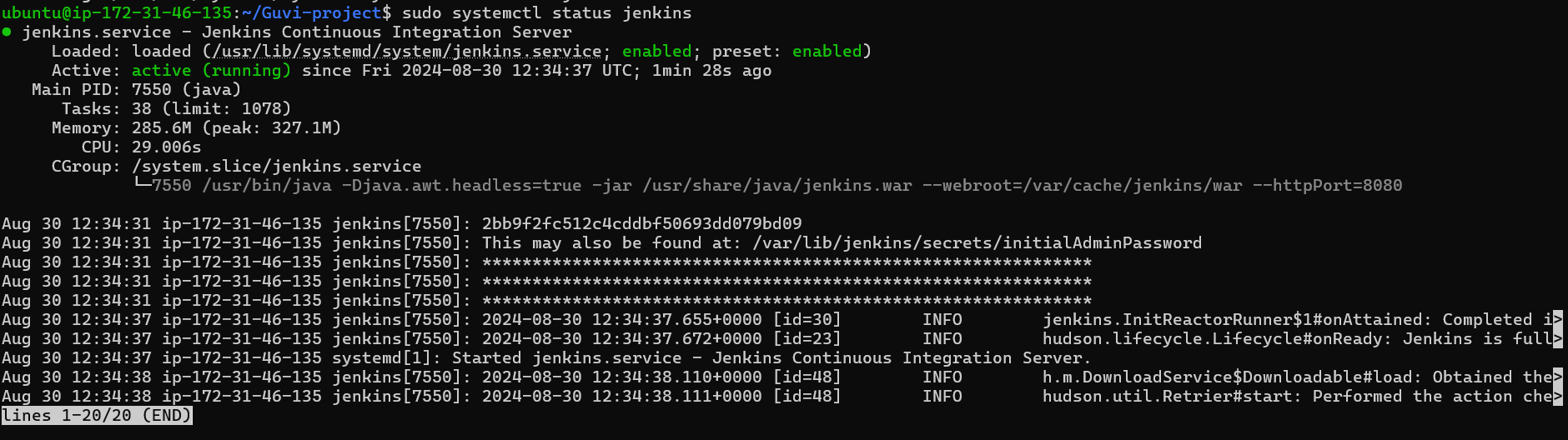
|  |
| --- |
| sudo apt update sudo apt install openjdk-17-jdk -y java -version |

* Jenkins is not included in the default Ubuntu repositories, so you'll need to add the official Jenkins repository:

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

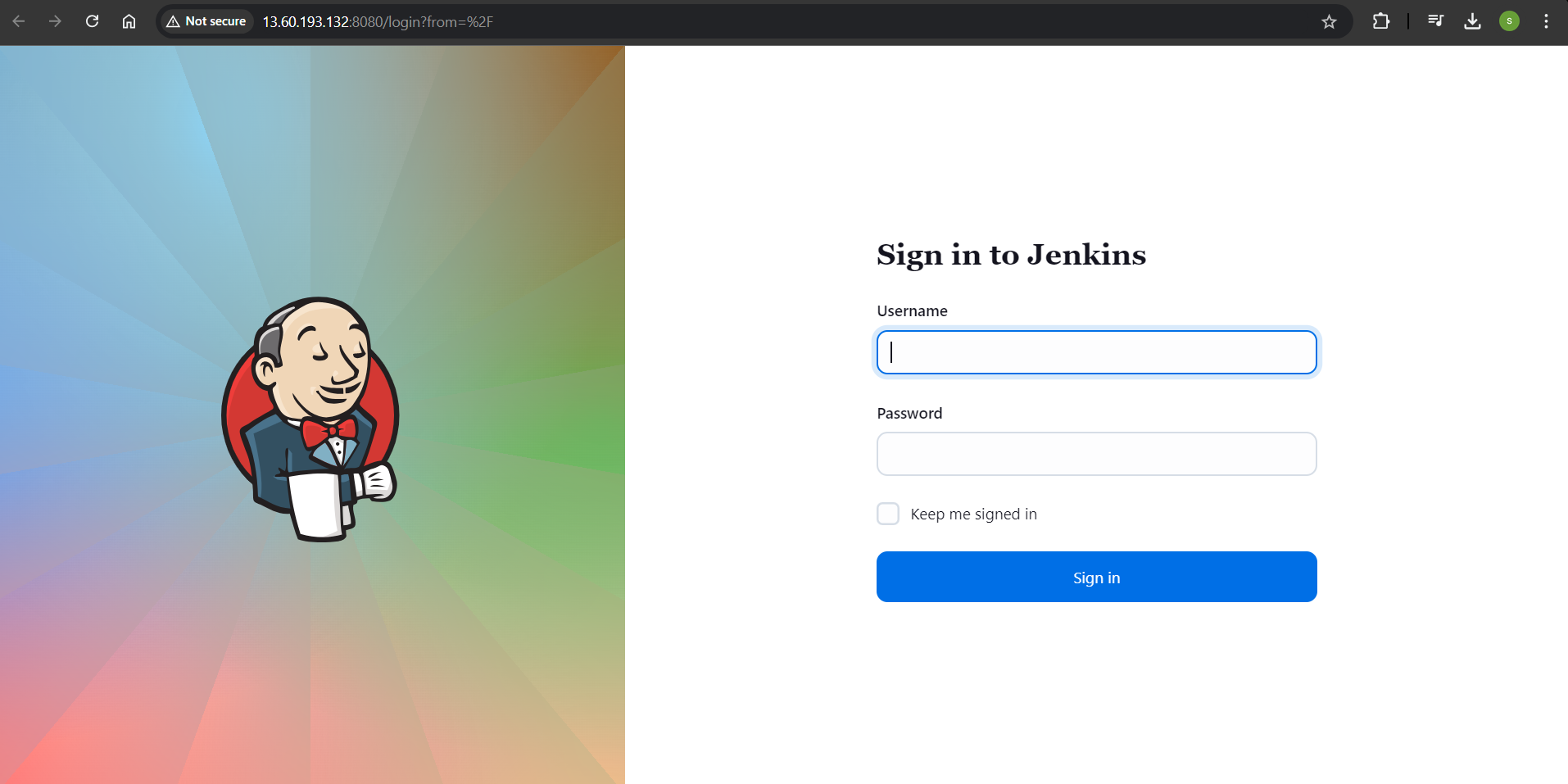
* Installing Jenkinks

|  |
| --- |
| sudo apt update sudo apt install jenkins -y sudo systemctl start Jenkins sudo systemctl enable Jenkins sudo systemctl status Jenkins |

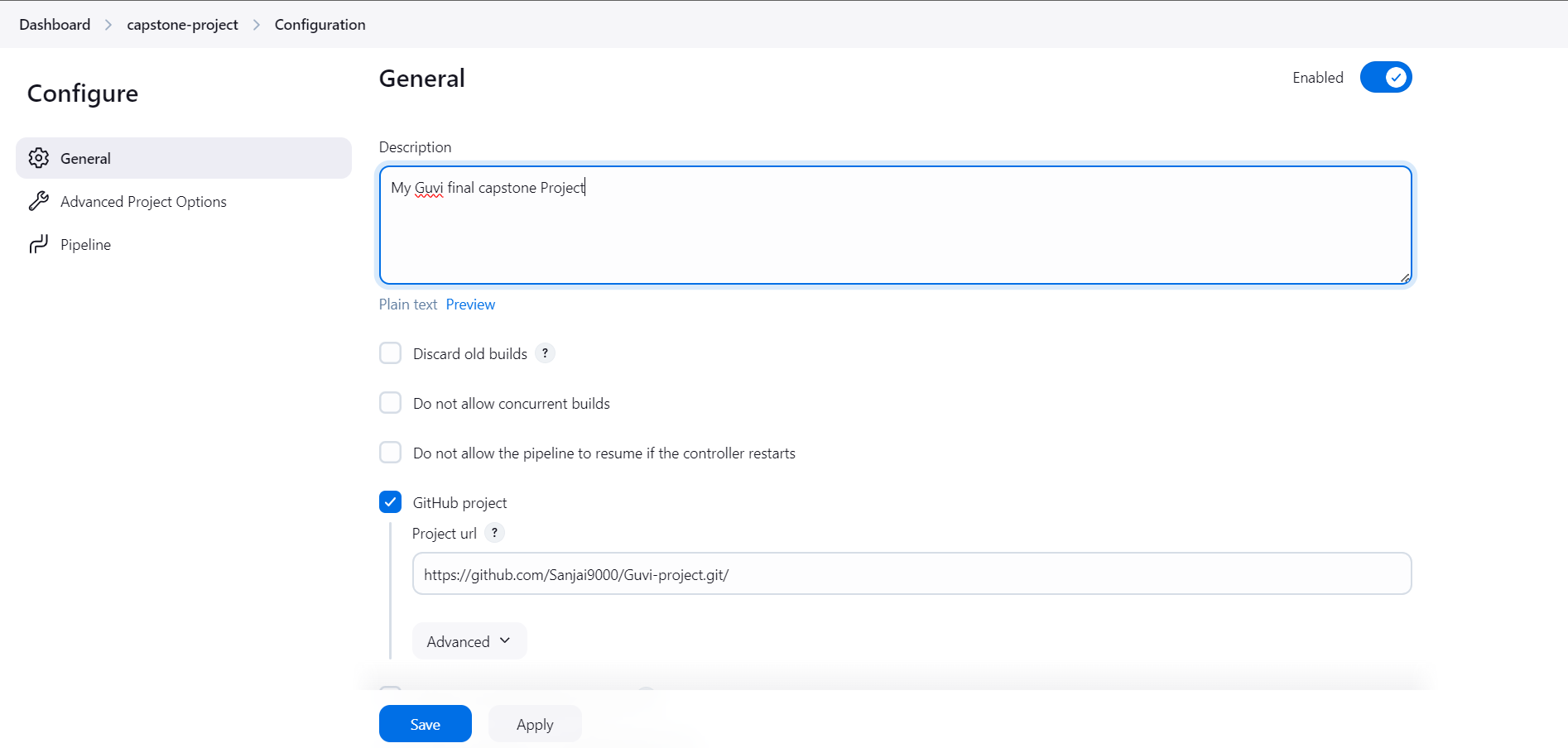


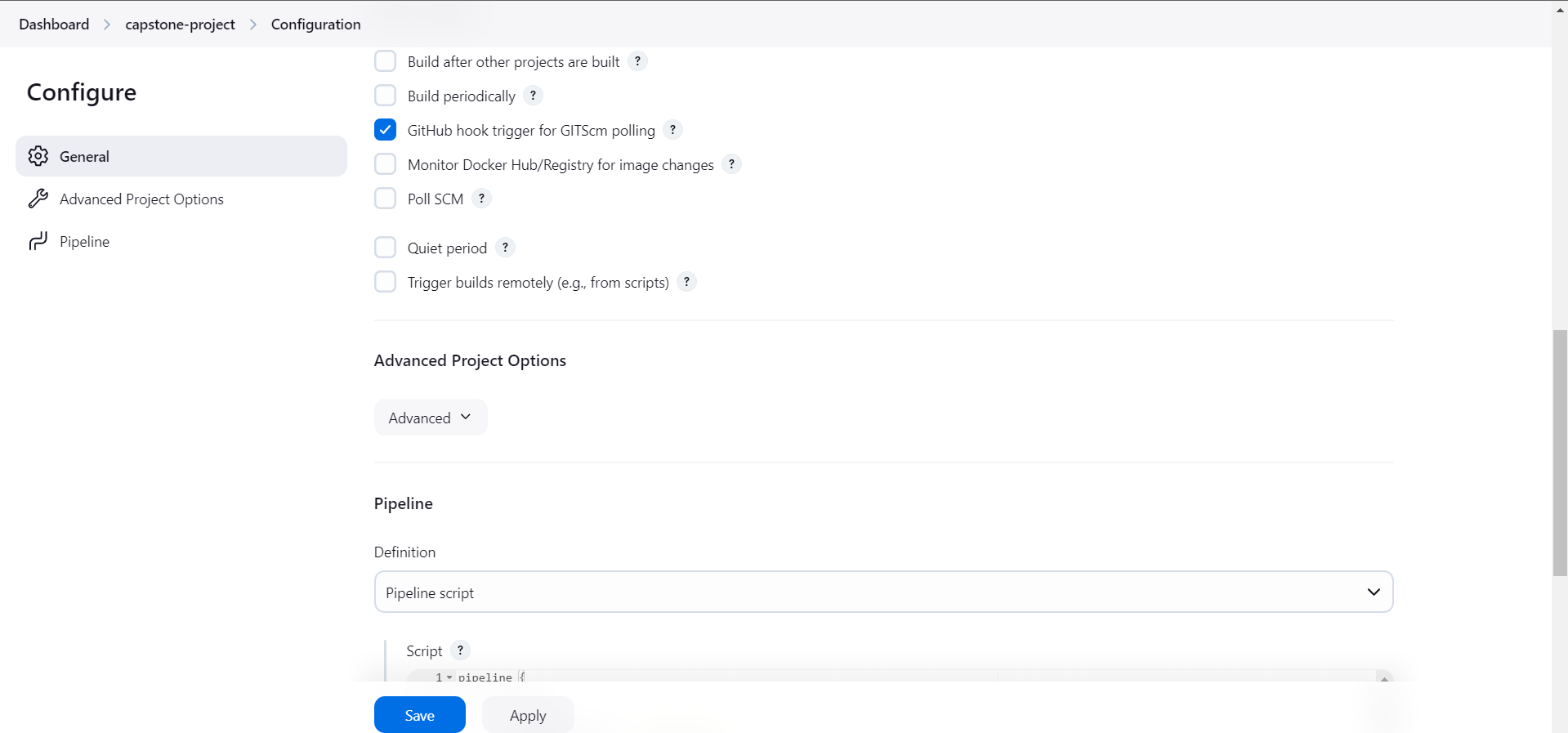
* To Access Jenkins complete the installation, you need to set up Jenkins by accessing it in a web browser. Open your web browser and navigate to http://<your\_server\_ip>:8080, here is my Jenkins URL http://13.60.193.132:8080/
* You'll see the "Unlock Jenkins" screen. To retrieve the initial password, run:

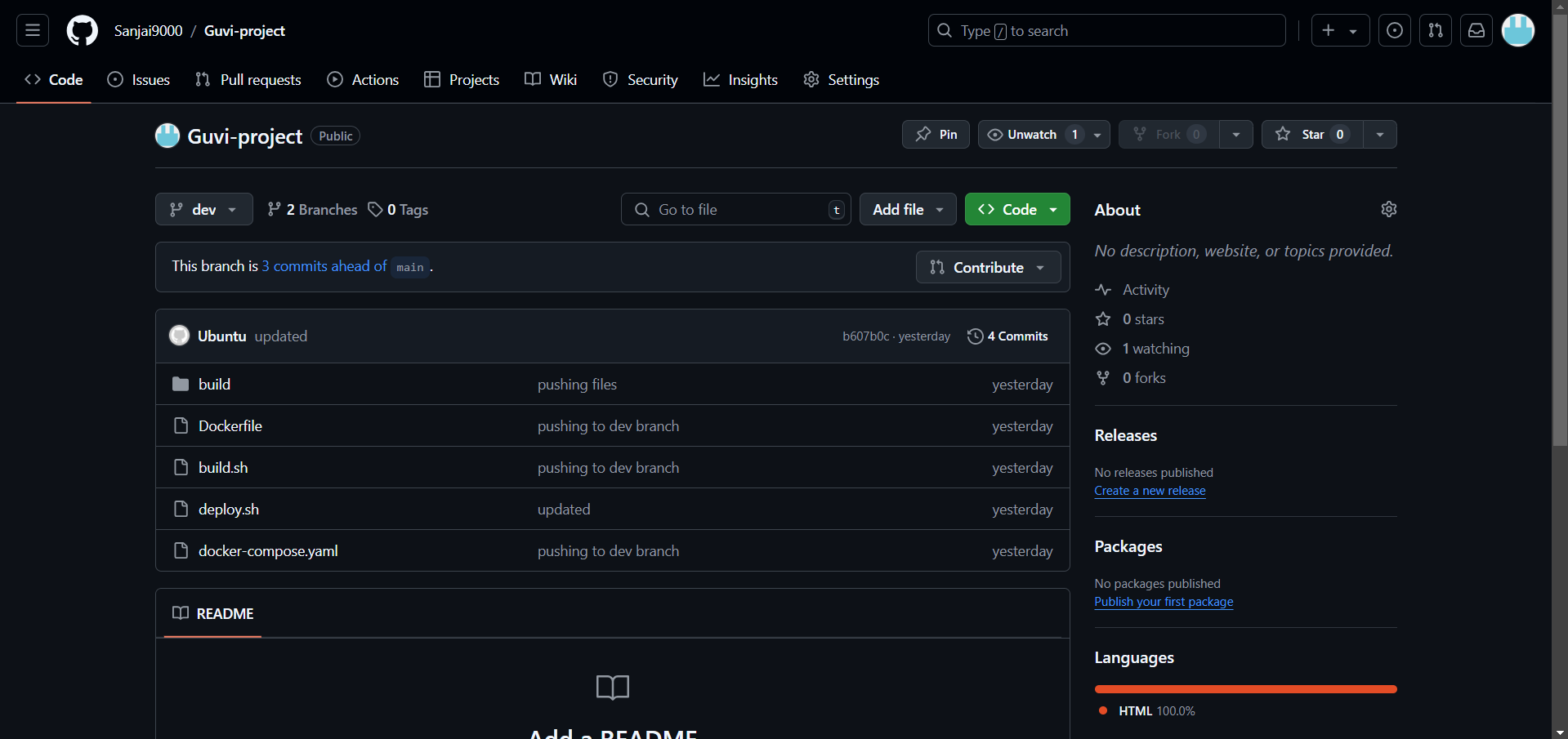
|  |
| --- |
| sudo cat /var/lib/jenkins/secrets/initialAdminPassword |

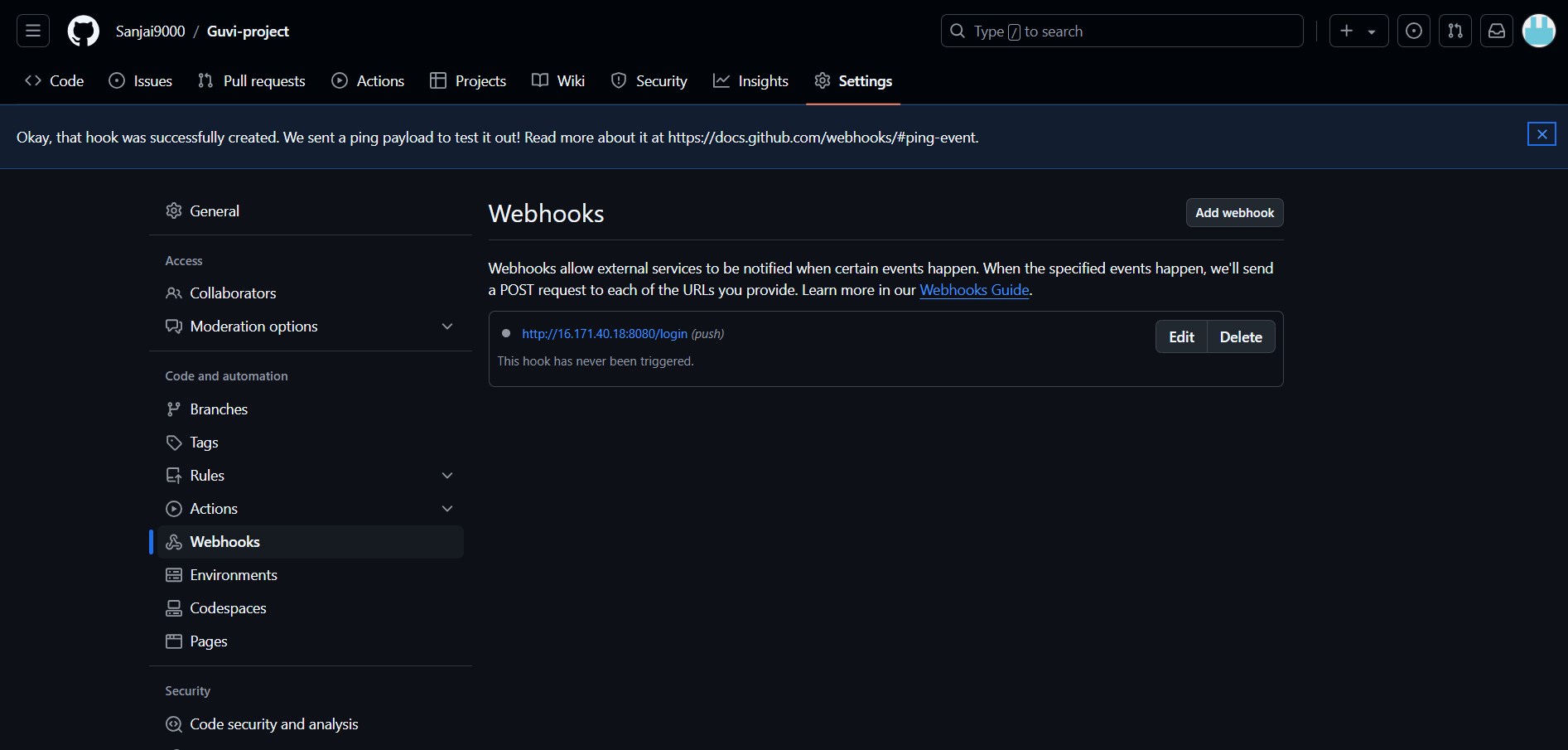


* Now checked and installed the required plugins for Github, Docker and Dockerhub. And also provided the required credentials in Jenkins. Jenkins dashboard->manage Jenkins->security->credentials->global
* To connect Jenkins to the Github repo with auto build trigger from both “dev” and master branch we have to create a new job in the Jenkins. Navigate to the Jenkins dashboard on left ->select ‘New Item’ and give a name for the pipeline->select item type as Pipeline-> OK to save the new pipeline.
* After clicking ‘Ok’ we will be lead to new page called configuration where we ca see all the settings under->General.

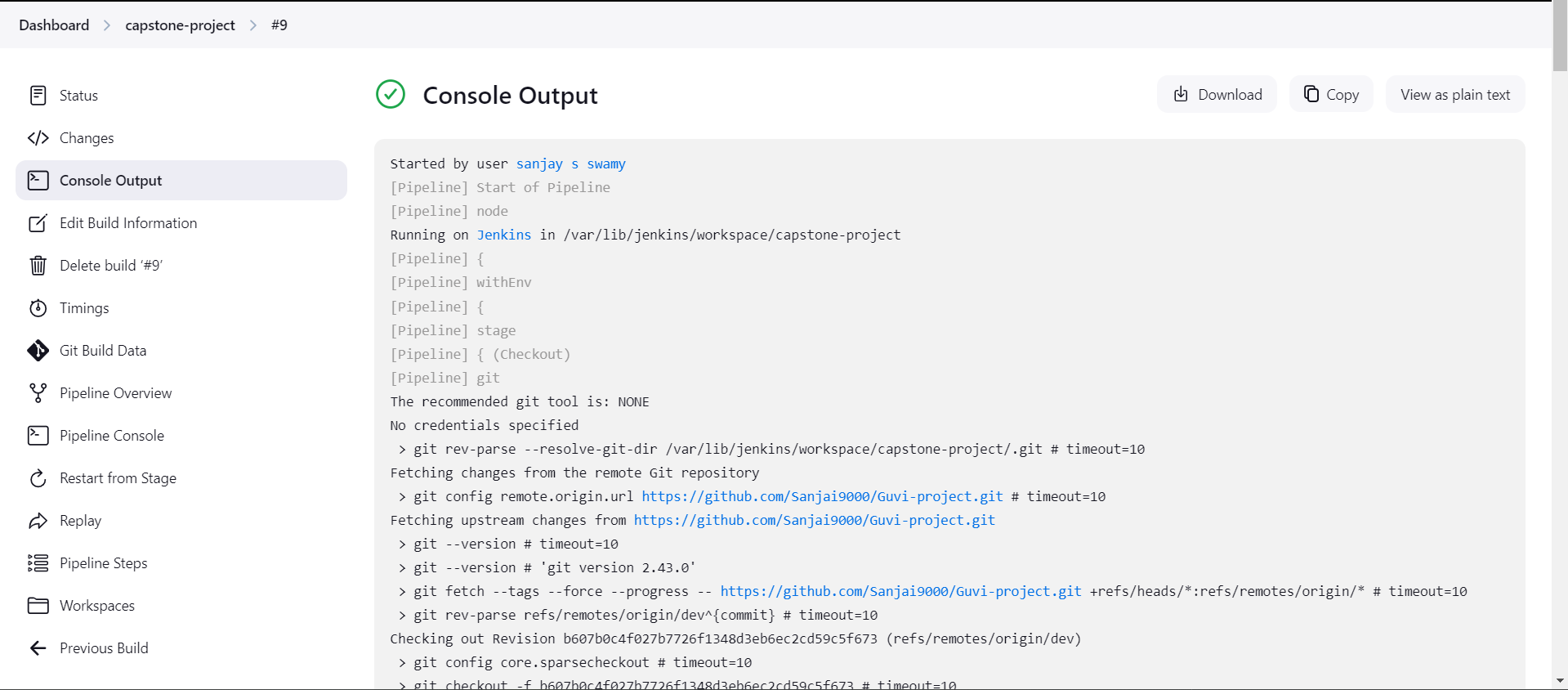


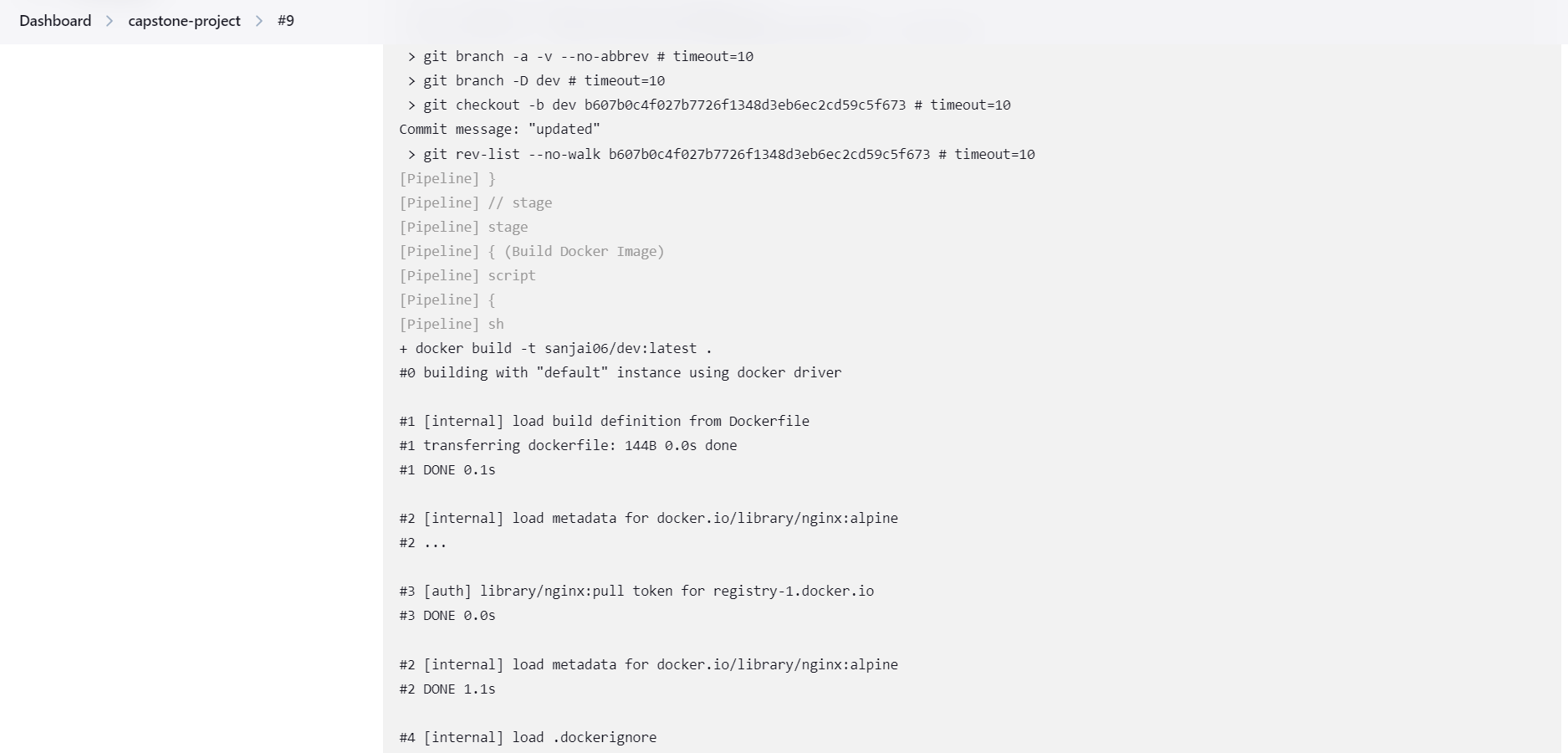


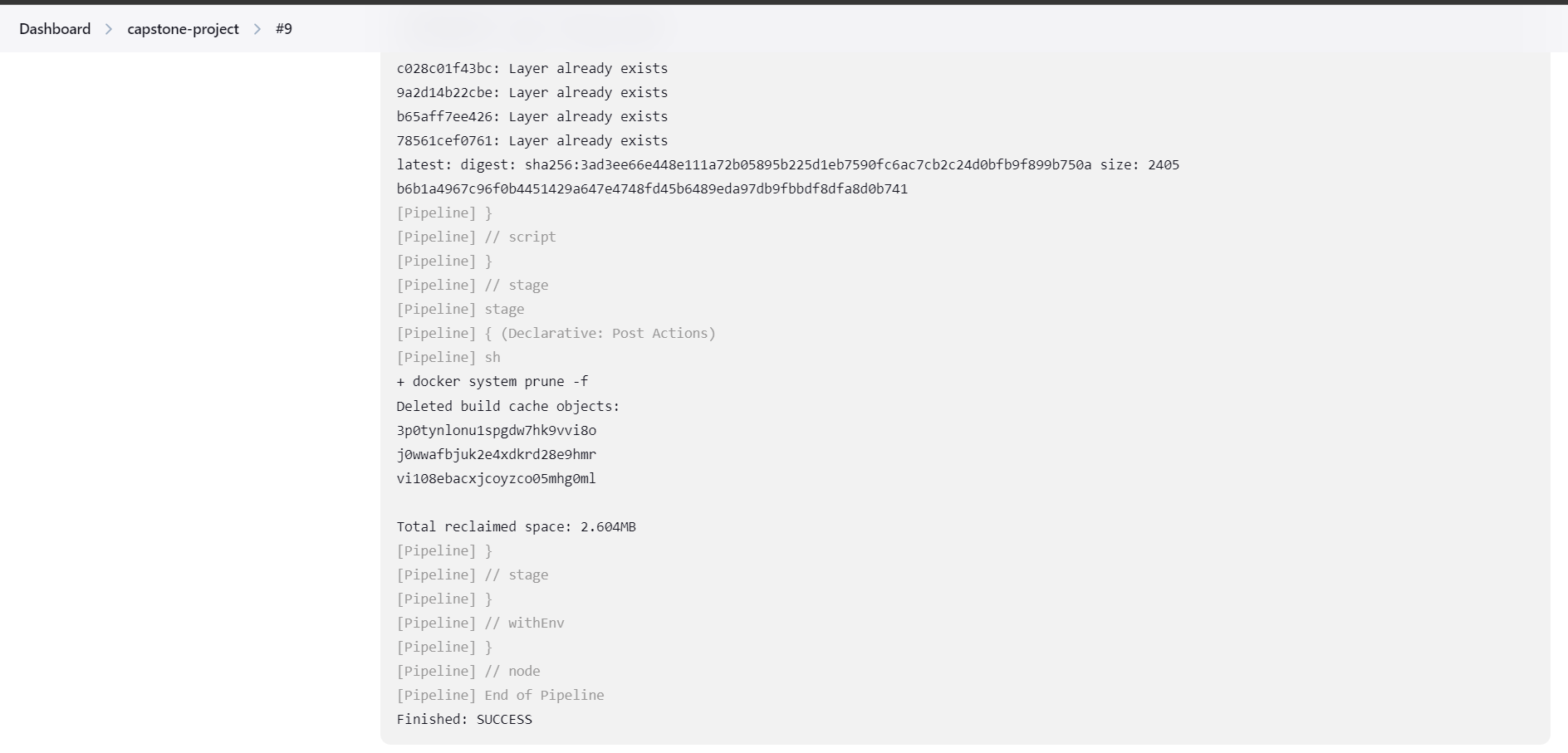


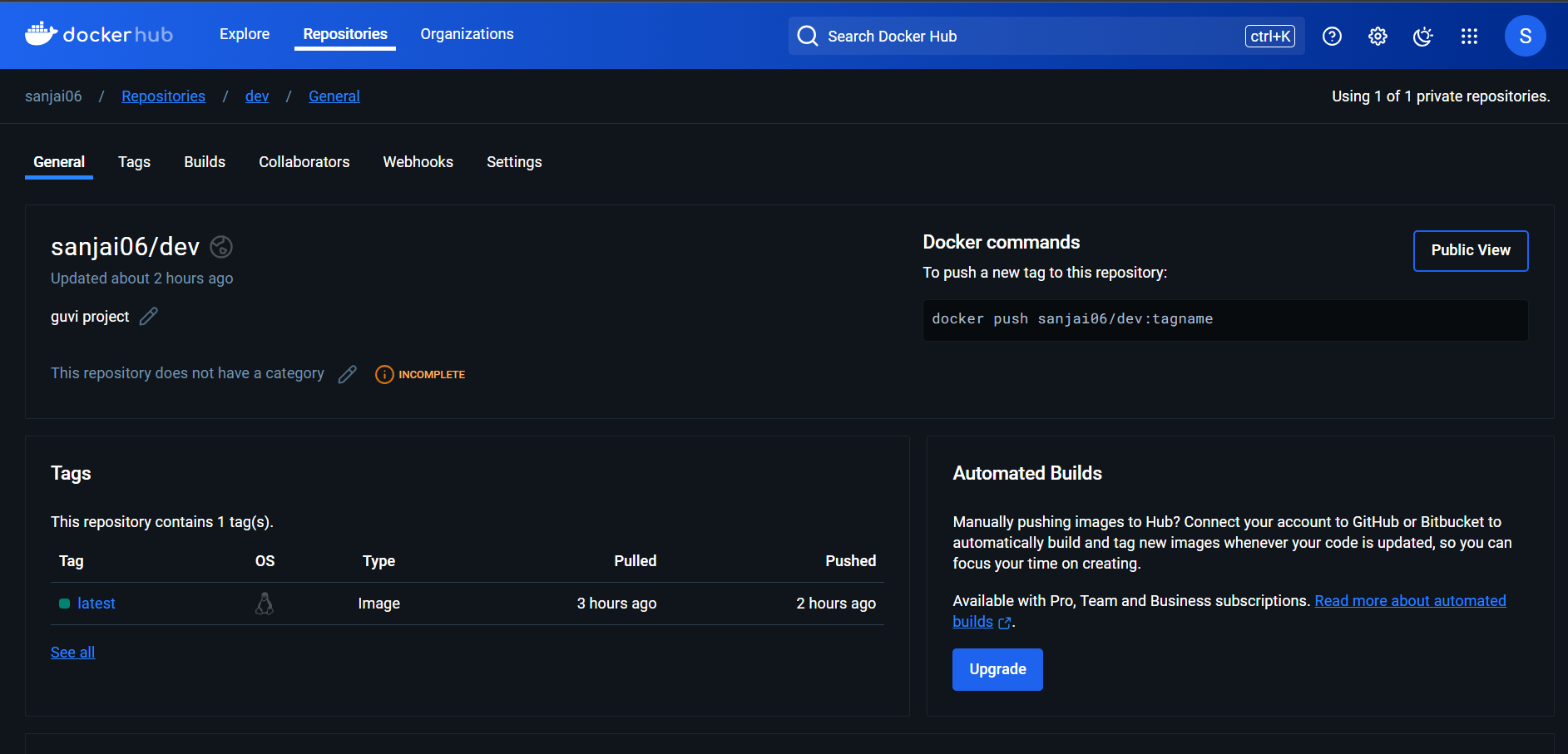


* Once code pushed to “dev” branch, docker image must build and pushed to dev repo in docker hub

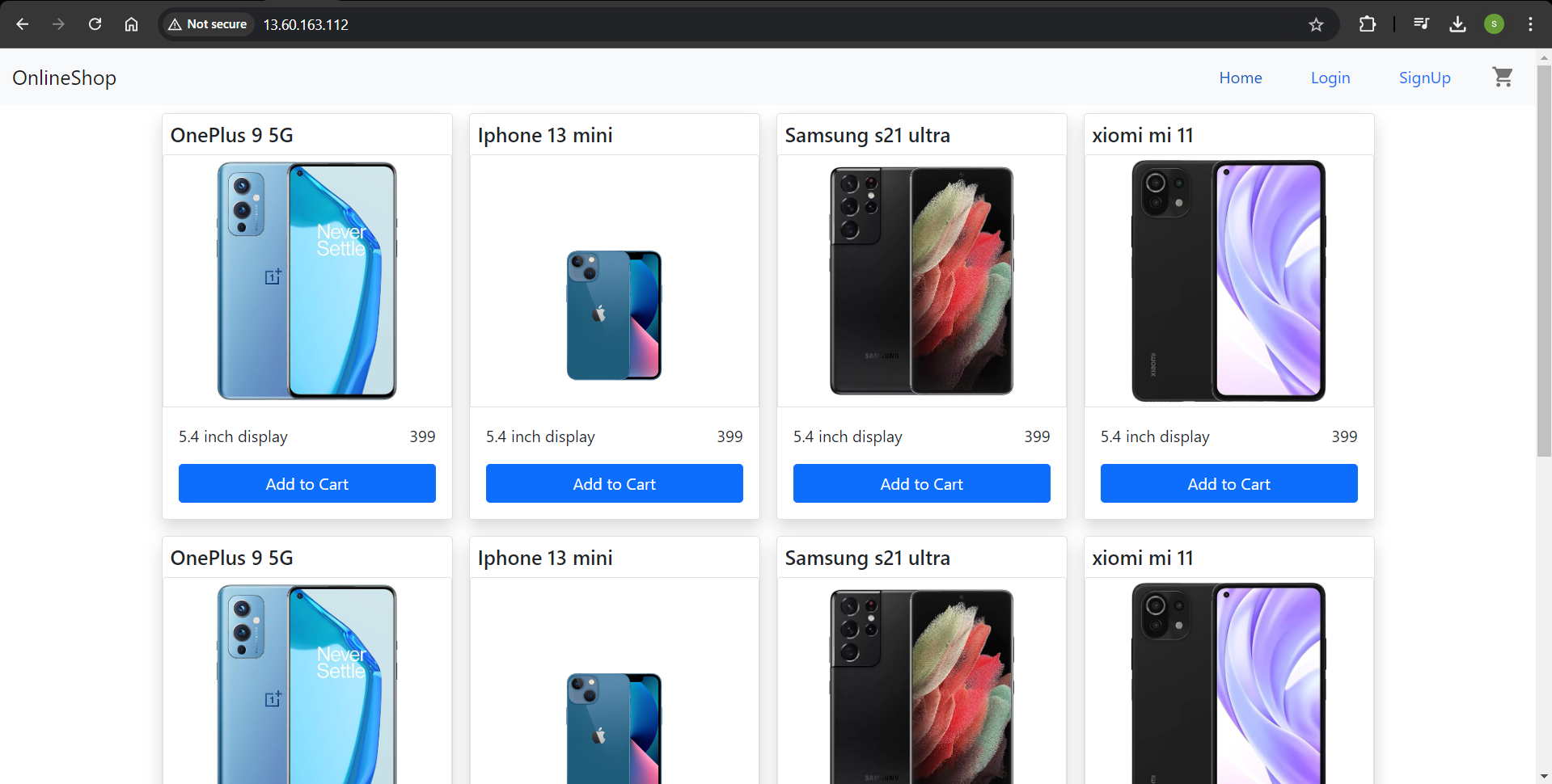








* Once the image is build and deployed we can see that the image is pushed to docker hub, and to see the site deployed edit the inbound rules and add port 80.
* In the browser using the Public IP along with the port 80 we can see the application running-> [React App](http://18.175.132.71/) ->13.60.163.112:80



**Monitoring:**

* For setting up a monitoring system to check the health status of the application I have used Prometheus. Firstly, ensure your system is up to date:

|  |
| --- |
| sudo apt-get update |

* Prometheus can be installed by downloading the binary files or using Docker. Here, we have download the binary:

|  |
| --- |
| cd /opt/ sudo wget <https://github.com/prometheus/prometheus/releases/download/v2.44.0/prometheus-2.44.0.linux-amd64.tar.gz> sudo tar -xvf prometheus-2.44.0.linux-amd64.tar.gz sudo mv prometheus-2.44.0.linux-amd64 prometheus cd Prometheus |

A screenshot of a computer program

Description automatically generated

* To access Prometheus we have to enable port number 9090 and as we have not assigned any target machine yet it will monitor the same running Monitor machine for now.

A screenshot of a computer

Description automatically generated

* Once Prometheus is installed, configured and running we have to use Node exporter. Node Exporter is used to collect system-level metrics. To install it:

|  |
| --- |
| cd /opt/ sudo wget <https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz> sudo tar -xvf node\_exporter-1.6.1.linux-amd64.tar.gz sudo mv node\_exporter-1.6.1.linux-amd64 node\_exporter cd node\_exporter |

A screen shot of a computer

Description automatically generated

* Now we can also see in the Prometheus the created Monitoring and target machine matrices are visible.

A screenshot of a computer

Description automatically generated

* Inorder to access the Node Exporter we have to enable the port number 9100. It is now collecting the default matrics of the running machine.

A screenshot of a computer

Description automatically generated

* Below are the ports that have been accessed in the main instance for the Project.

A screenshot of a computer

Description automatically generated

**Submission:**

* Github repo URL :
* Deployed site URL : [React App](http://18.175.132.71/) ->13.60.163.112:80
* Docker images name : sanjai06/dev:latest