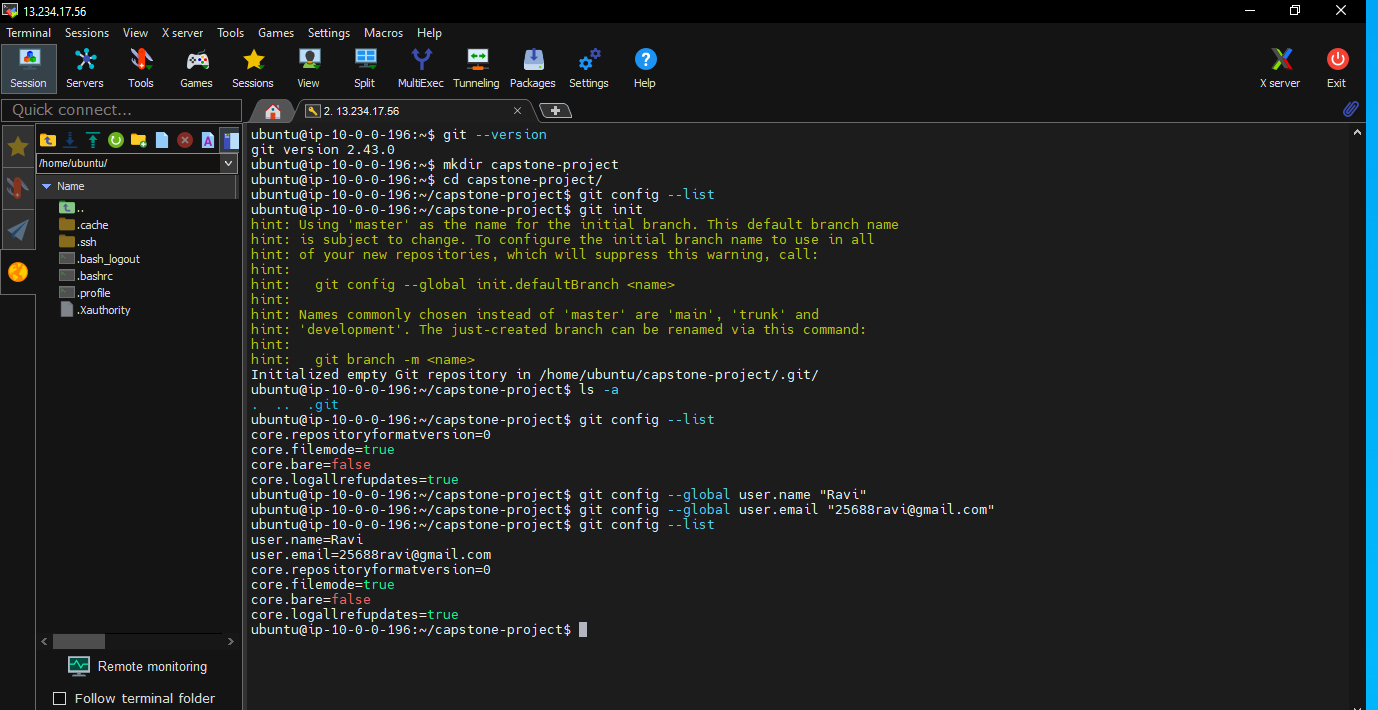
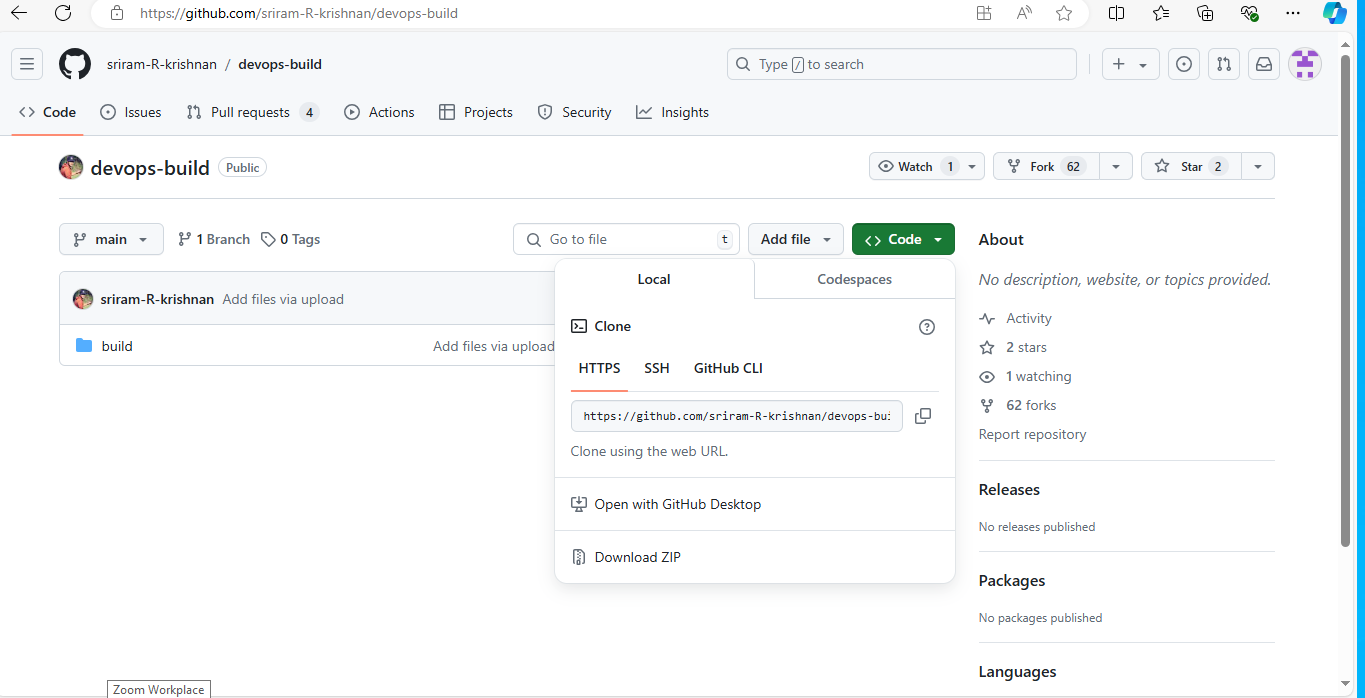
**CAPSTONE PROJECT – REACT APPLIACTION DEPLOYMENT**

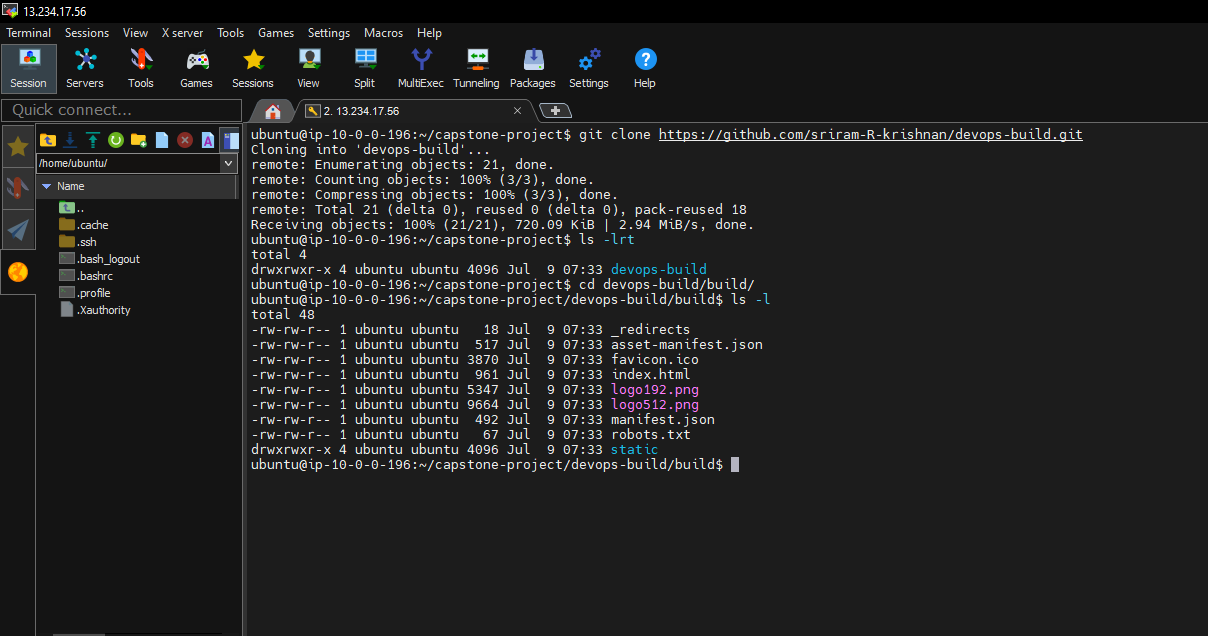
1. **APPLICATION**
2. Configuring git in Ubuntu server



1. Copying the github repo URL <https://github.com/sriram-R-krishnan/devops-build>

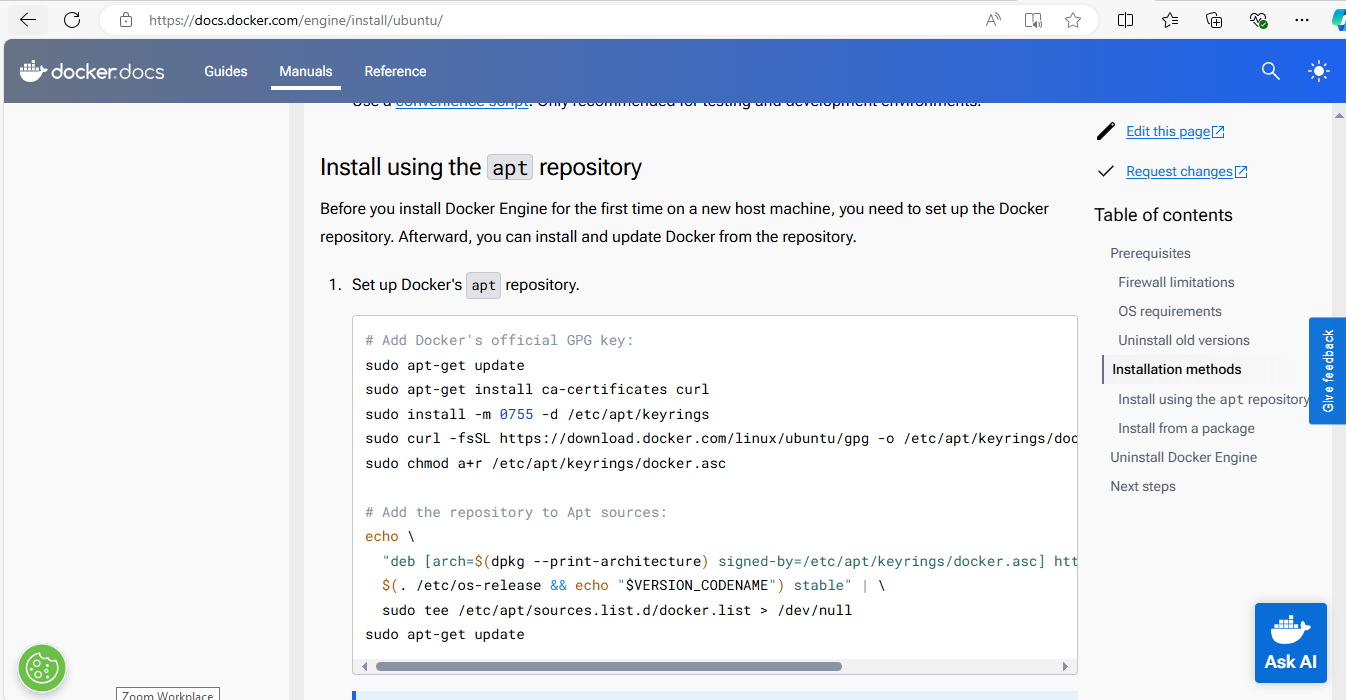


1. Cloning the github Repo URL

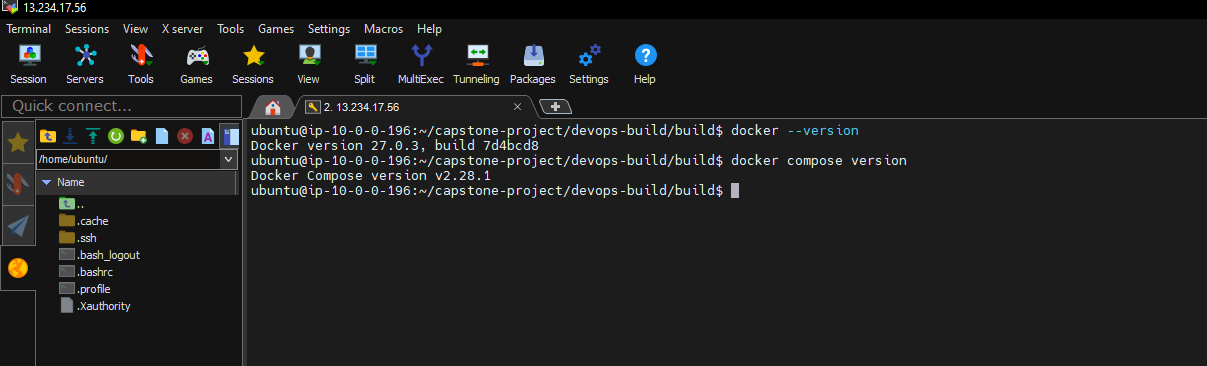


1. **Docker:**

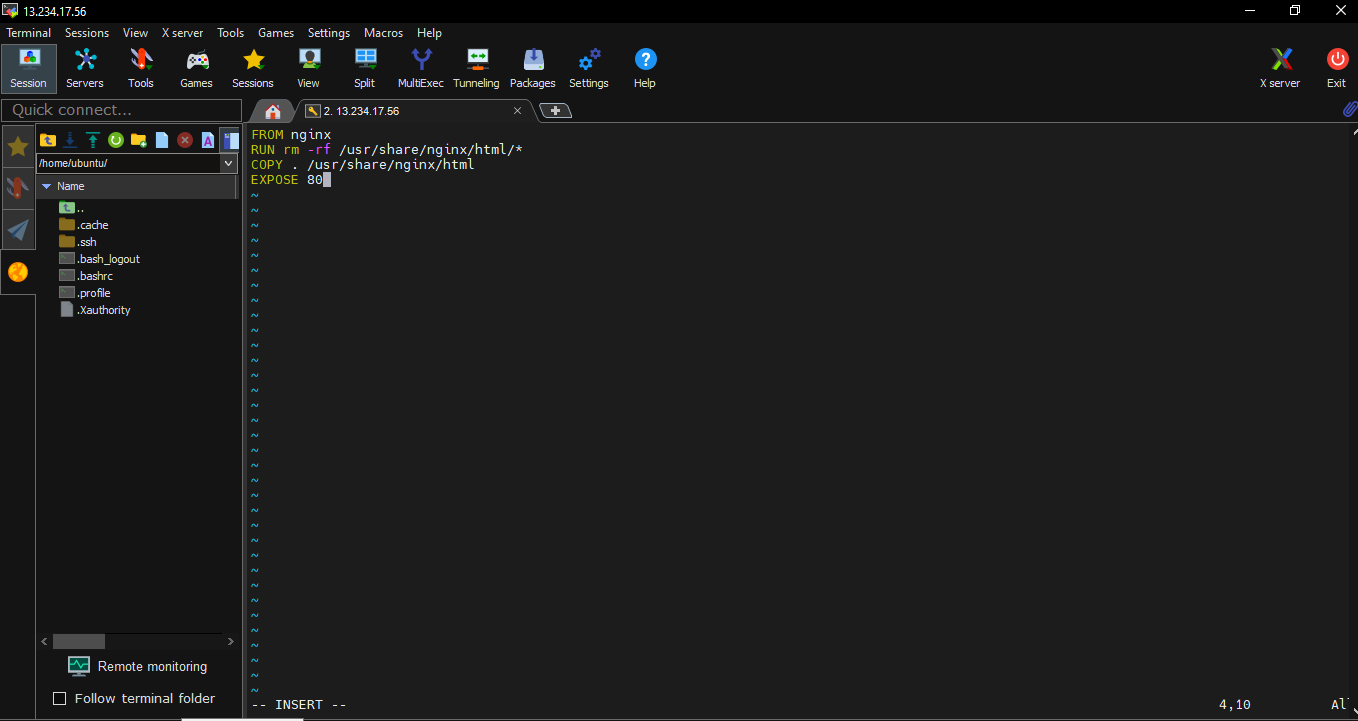
1. Installing docker in Ubuntu server by docker official documentation

****

1. Docker and docker compose installation done successfully

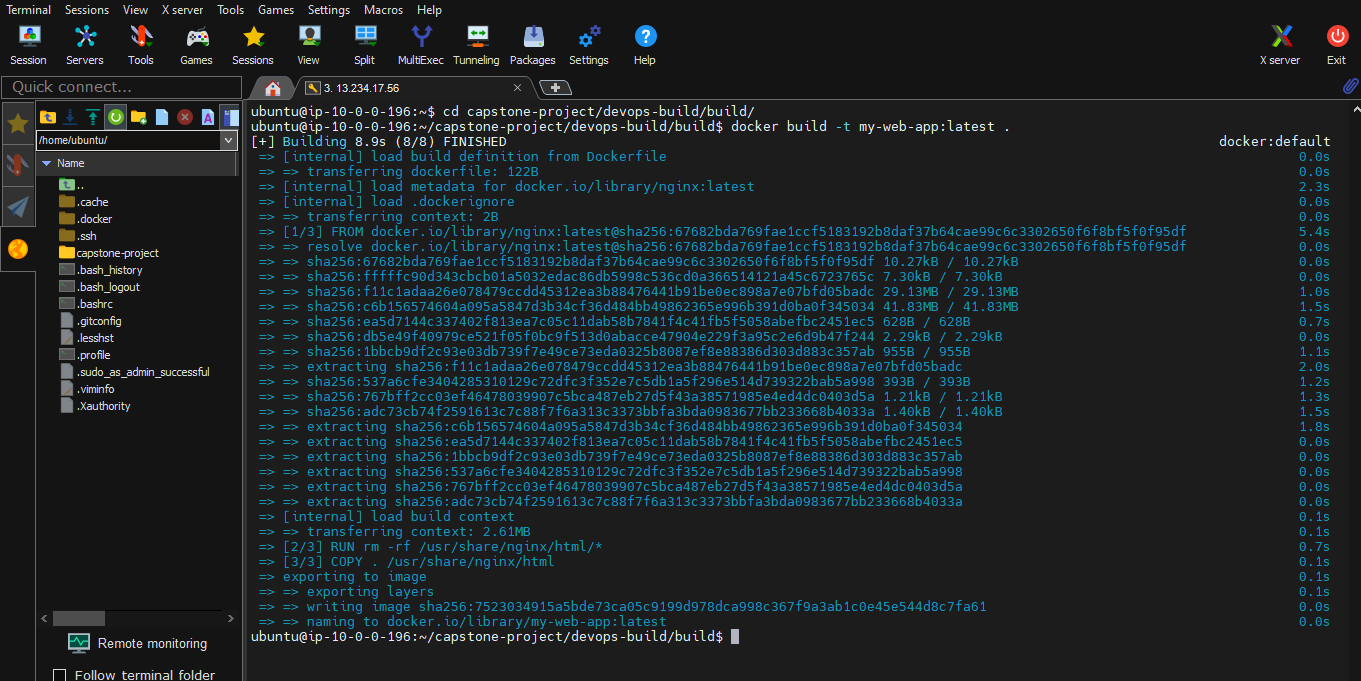
****

1. Creating a Dockerfile by the command vi Dockerfile

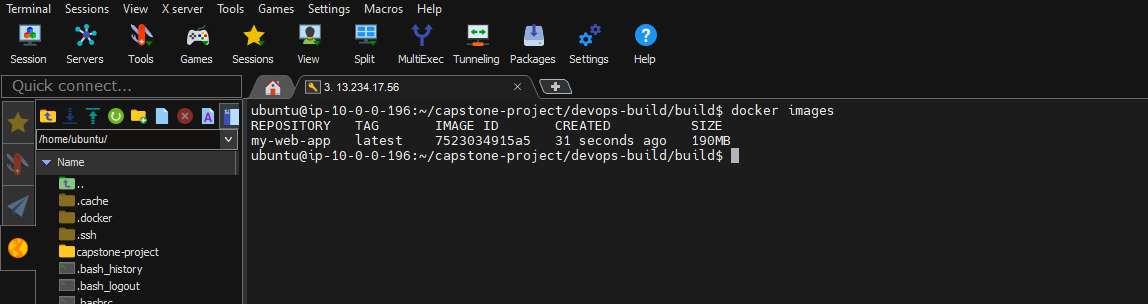
****

1. Building a docker image from the dockerfile by the command

docker build –t <imagname>:tagname

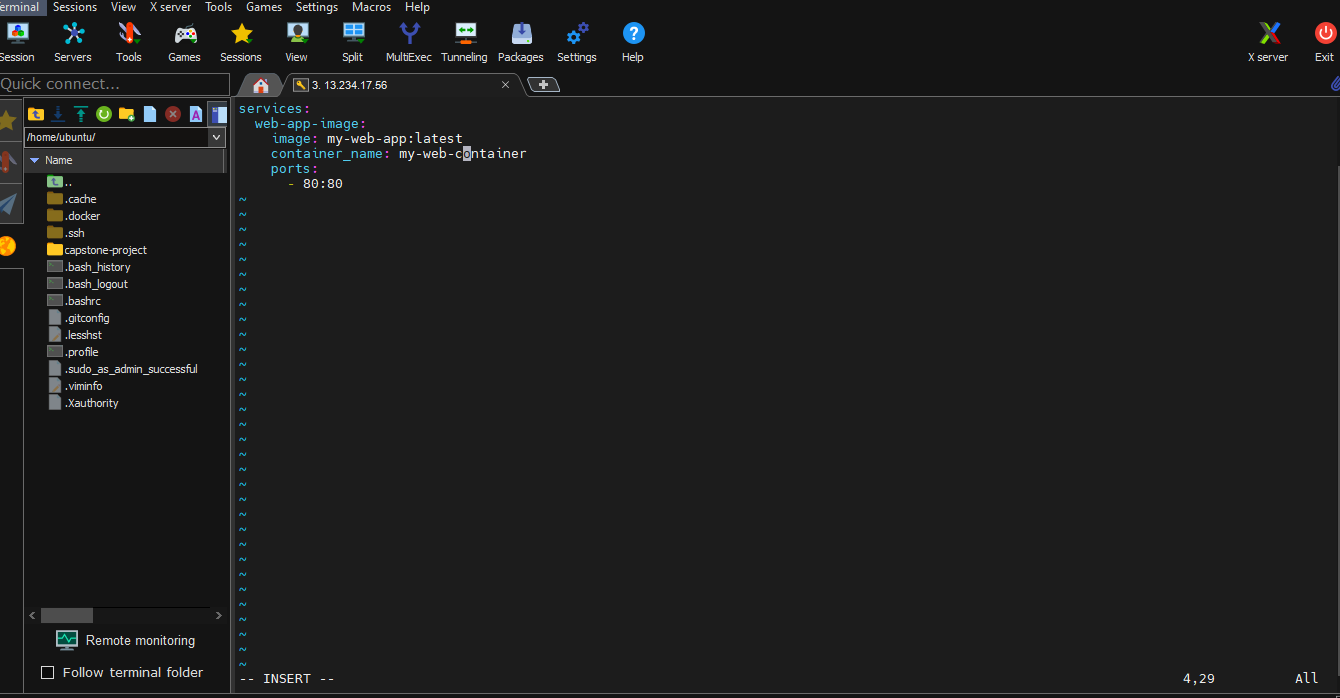


1. Docker image has been created

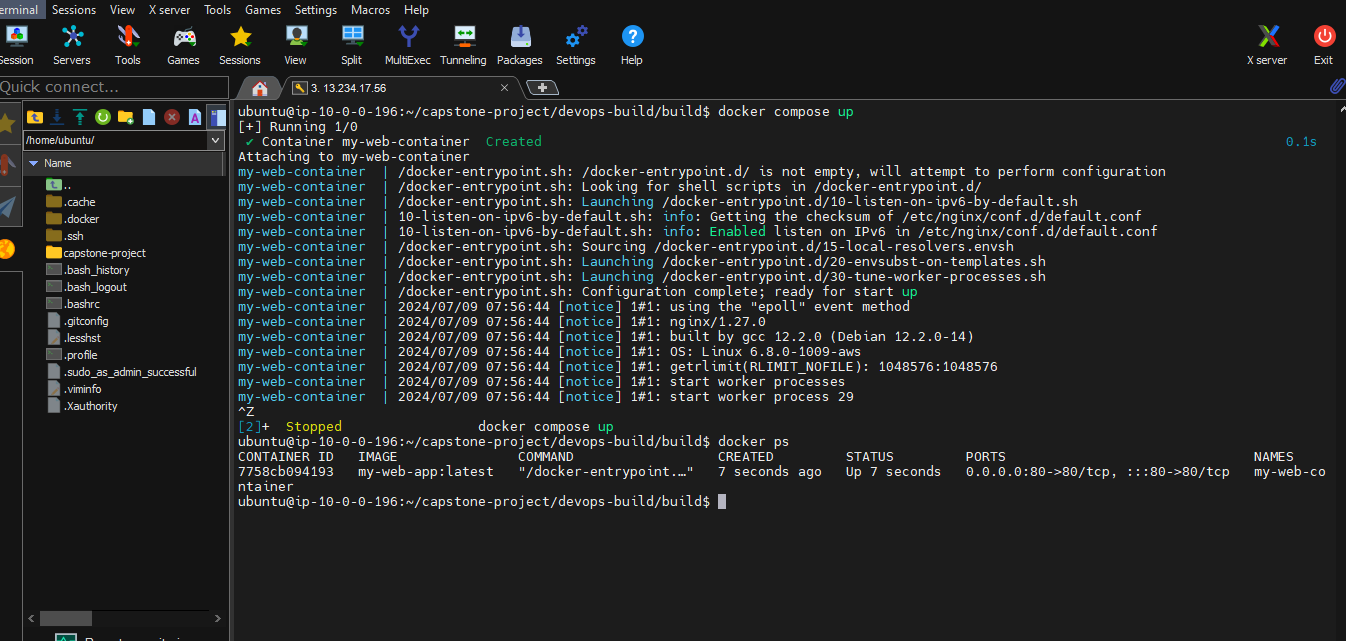


1. Creating a docker compose file by the command

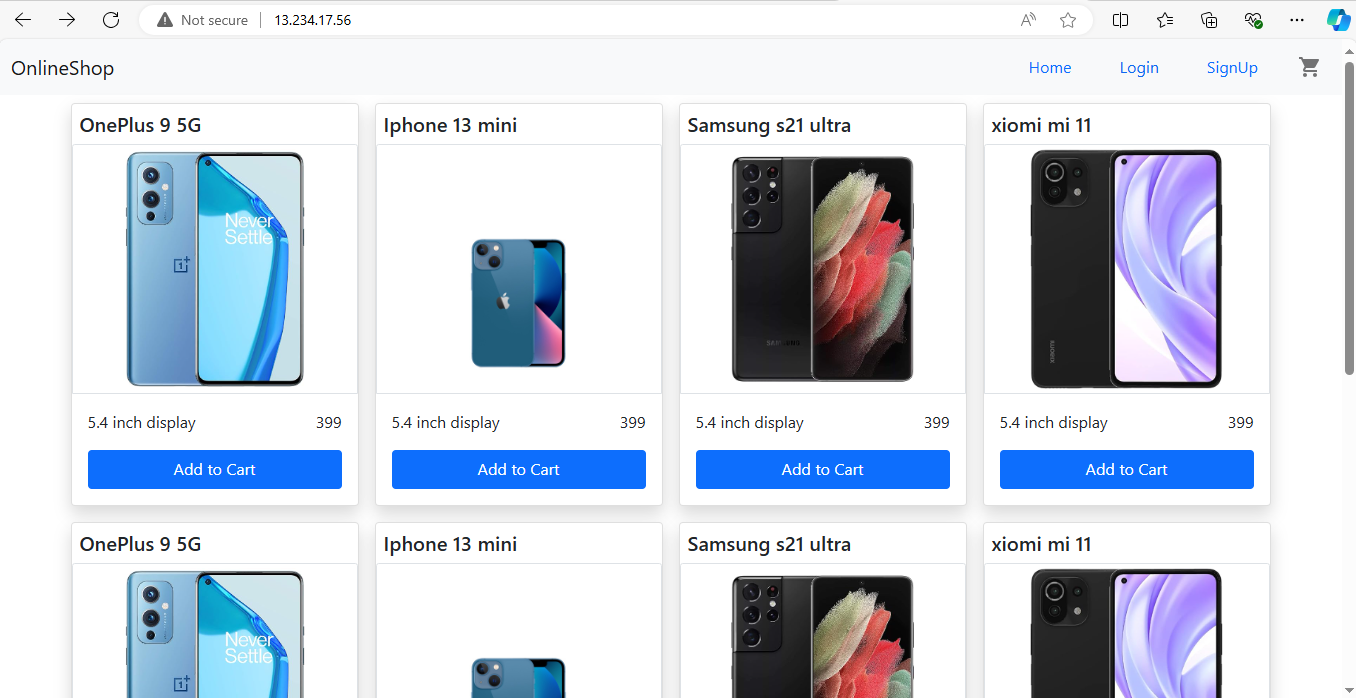
vi docker-compose.yml



1. Running the docker compose file



1. Successfully dockercompose is working and web URL page is opend

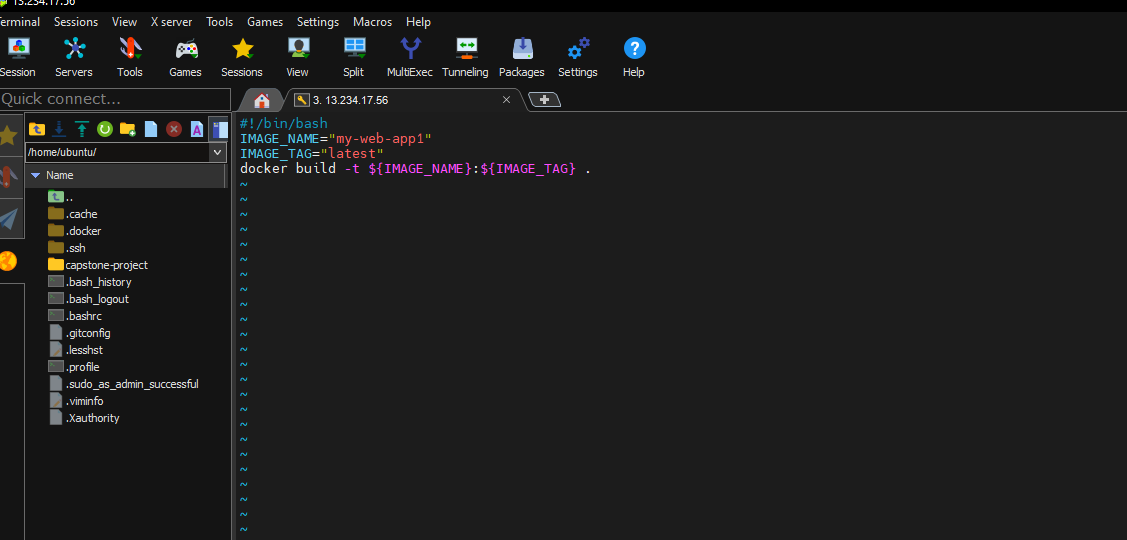


1. **Bash Scripting**

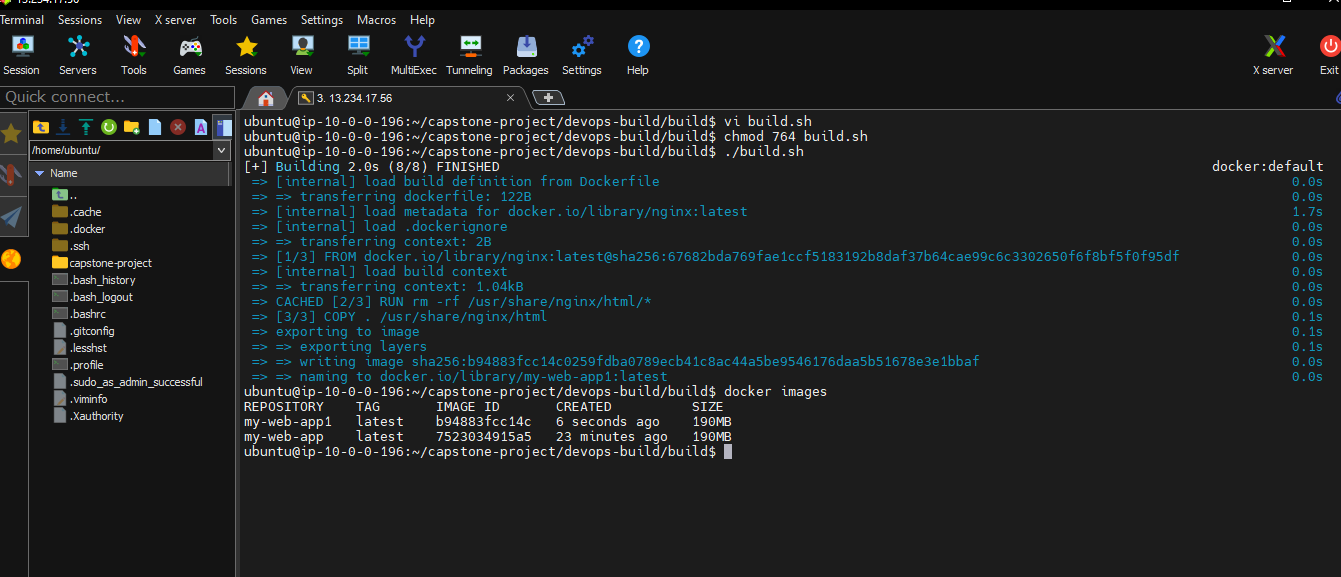
Writing 2 script files build.sh for building docker images and deploy.sh to deploy the dockerimage to server.

1. Writing build.sh script file

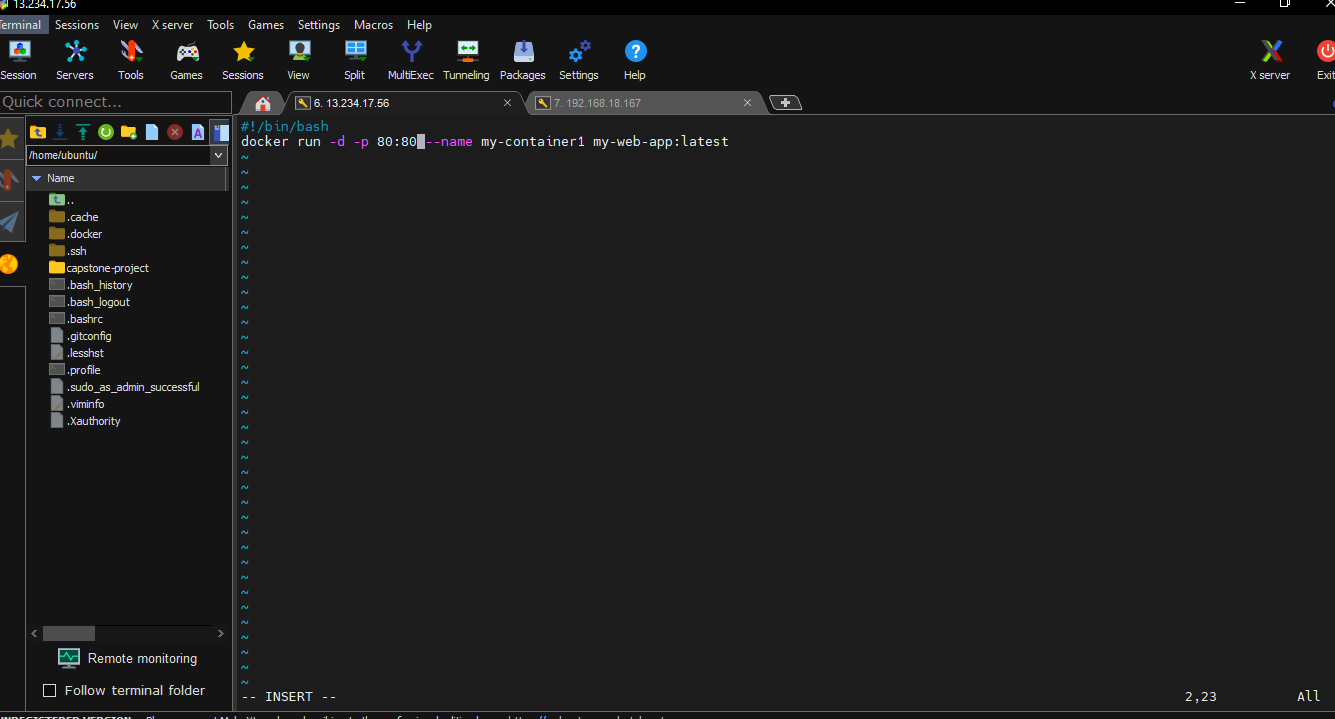
Starting by the command <vi build.sh>



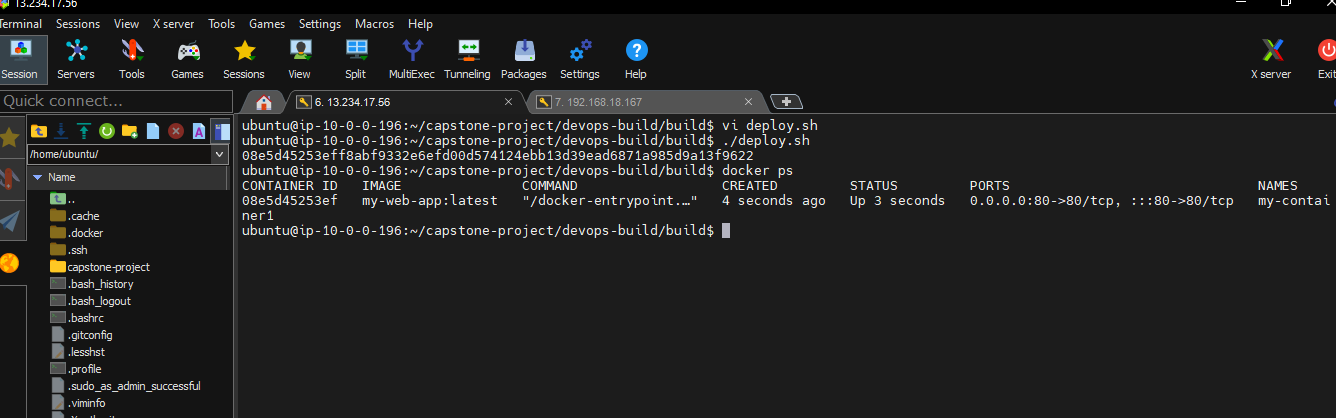
And changed the script file permission to execute the script file and executed script file and display by the command docker images



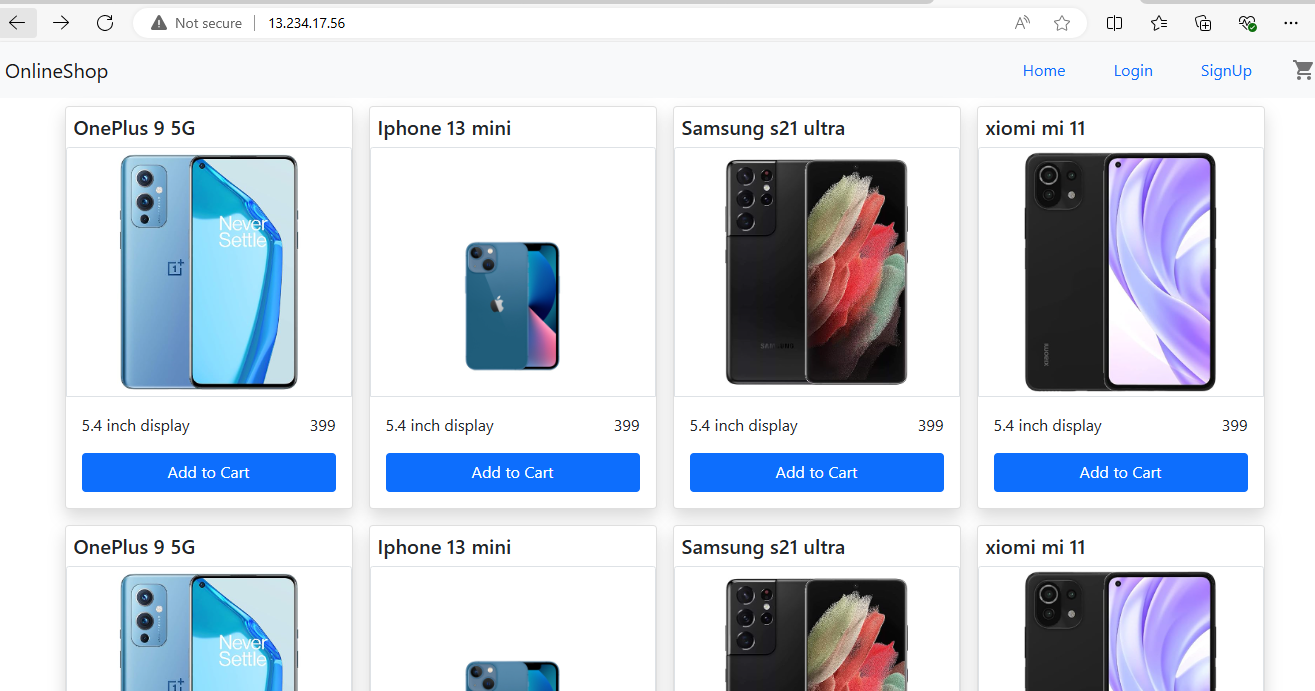
1. Writing the deploy.sh script by the command vi deploy.sh



And changed the permission and deploy the script file written for deployment



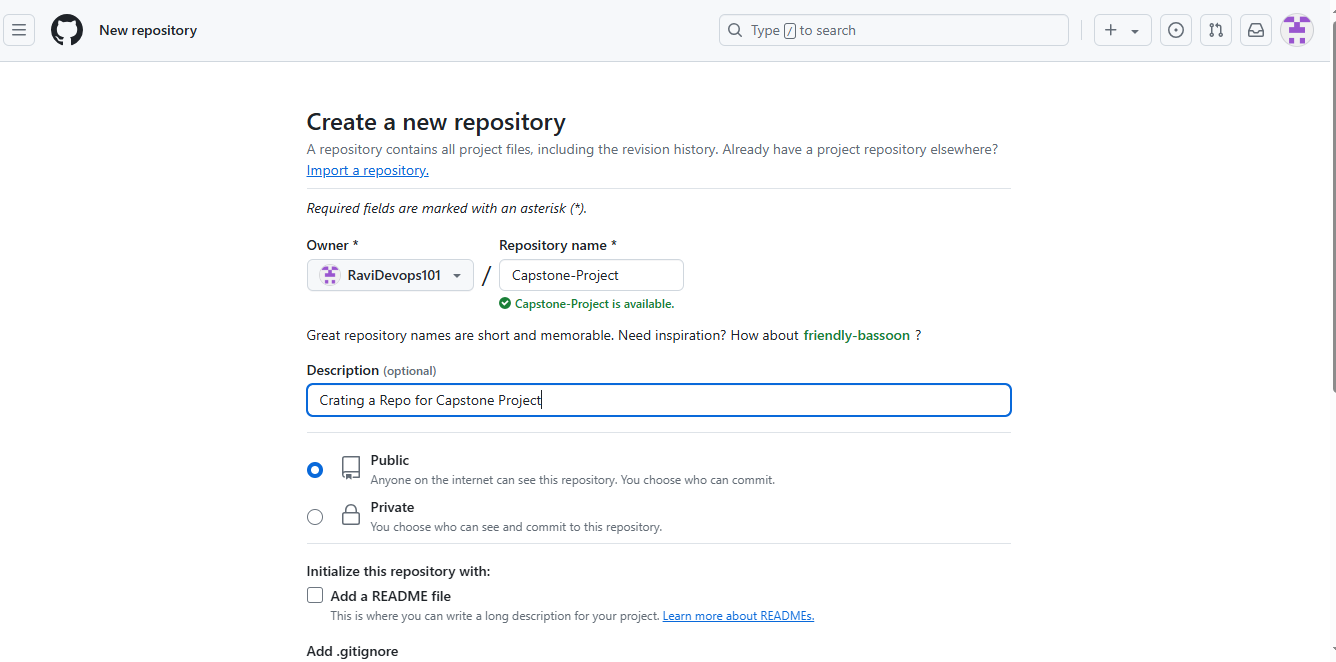
Result of deploy script and Deployed URL



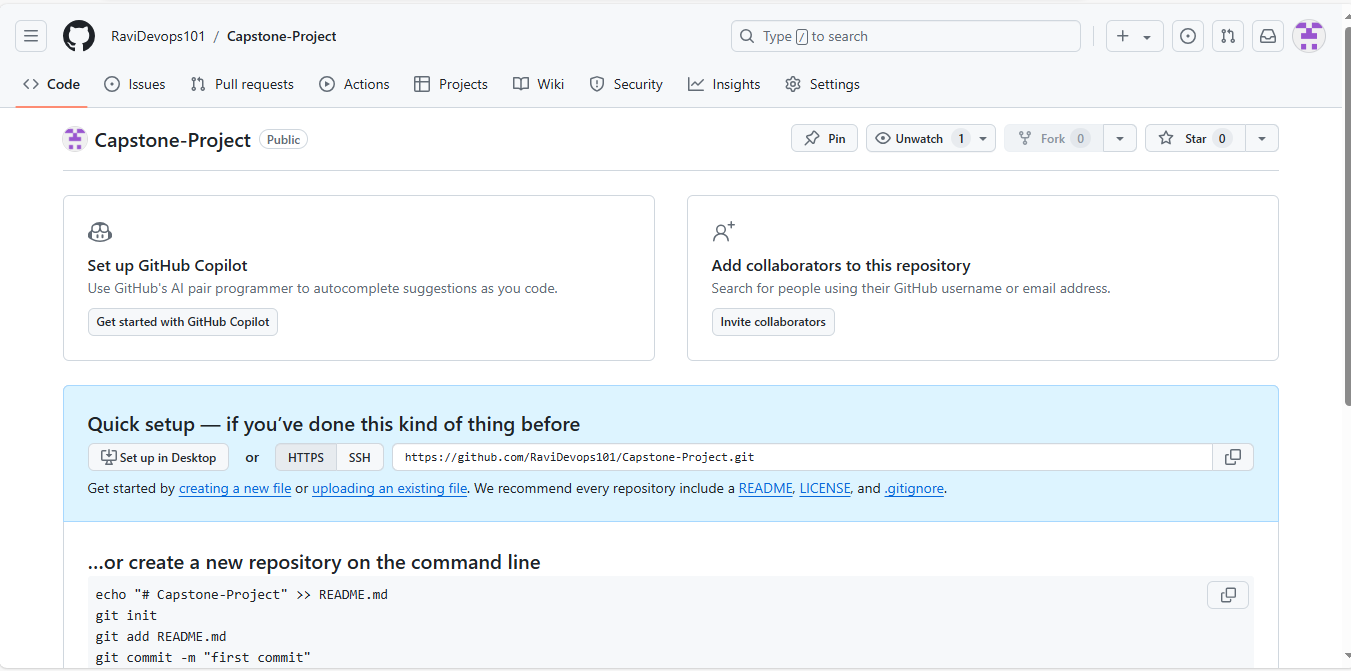
1. **Version Control:**

Push the code to github to dev branch and using gitignore file

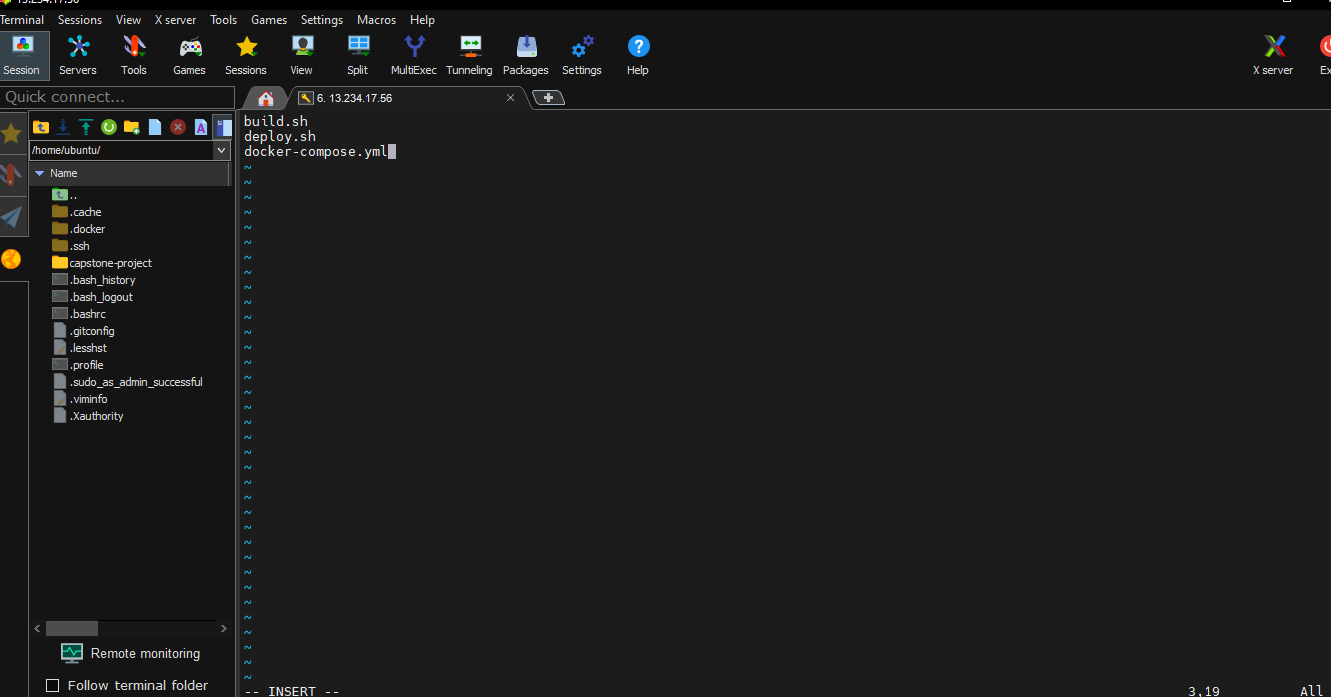
1. Creating a Repository in git hub



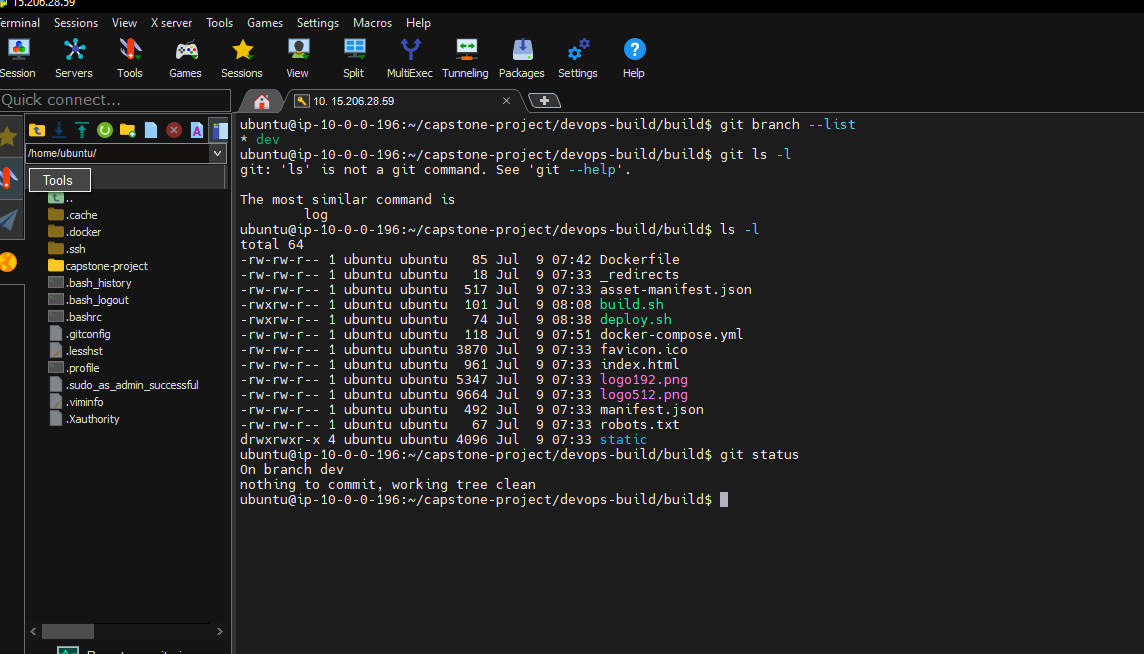
1. Repository Created



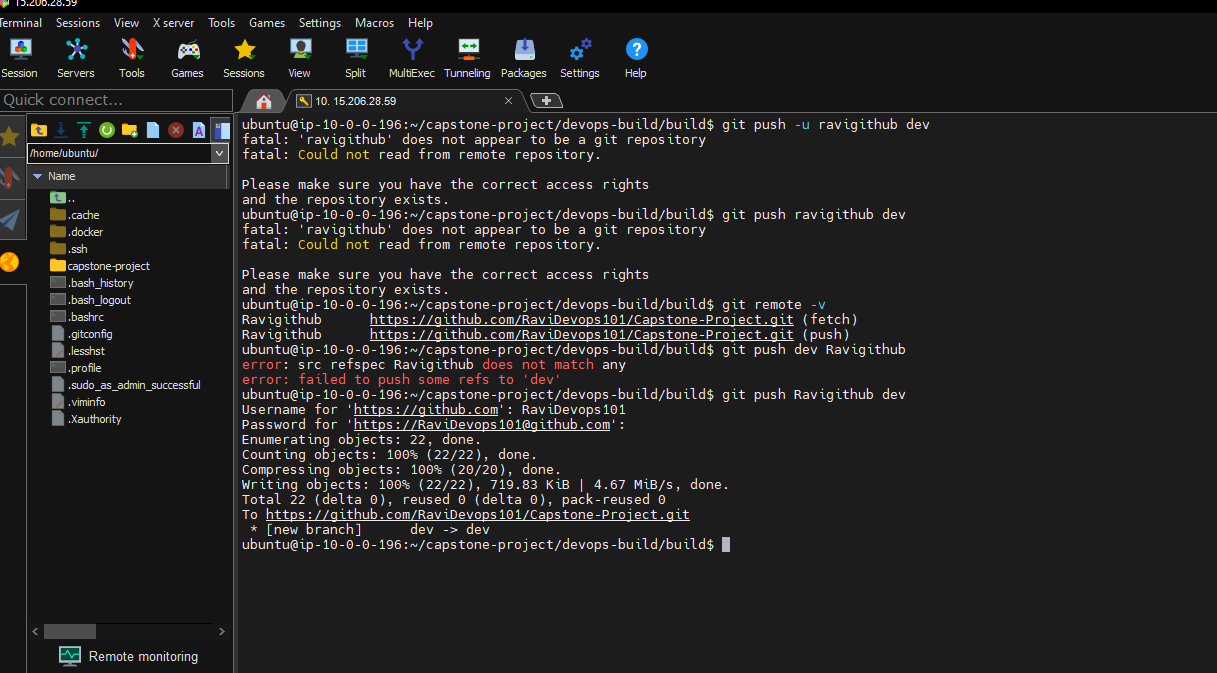
1. Creating a gitignore file by the command vi .gitignore



1. Creating a branch dev in git repository by command git checkout –b dev



1. Pushuing the code to github URL



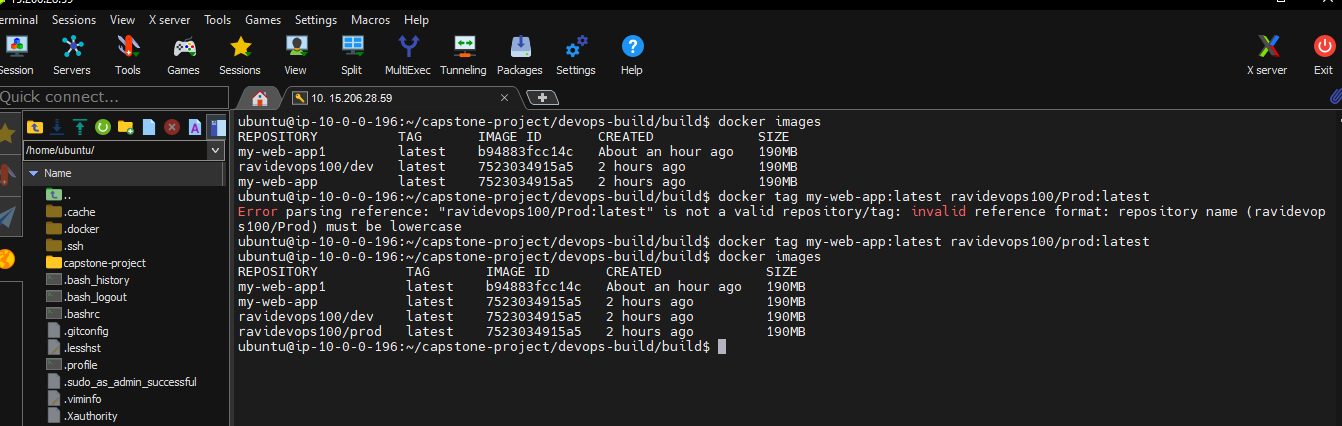
1. Finally code pushed to the github URL



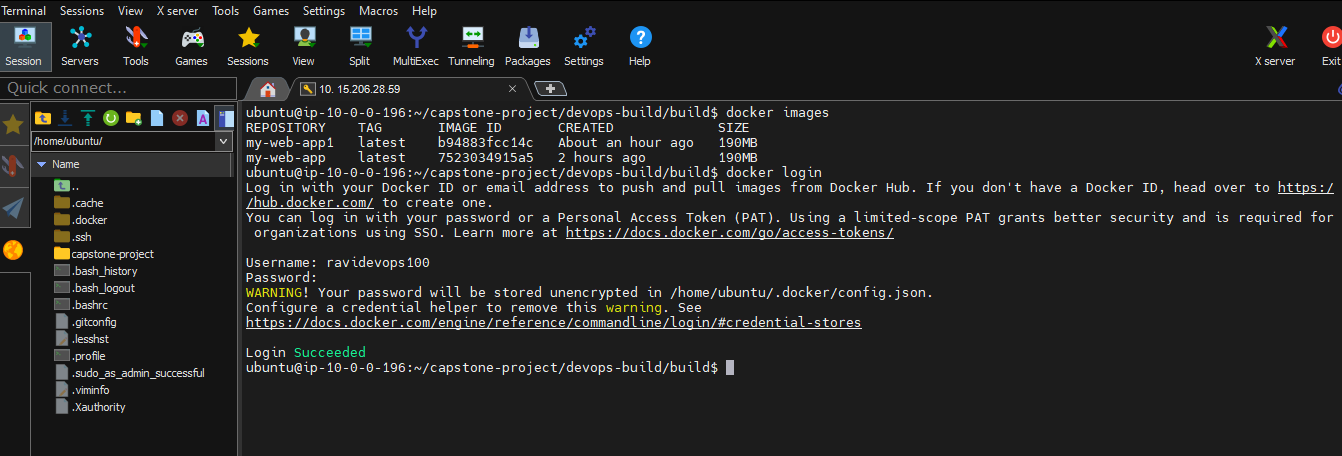
1. **Docker hub:**

Creating 2 Repos dev and prod in docherhub by pushing the images

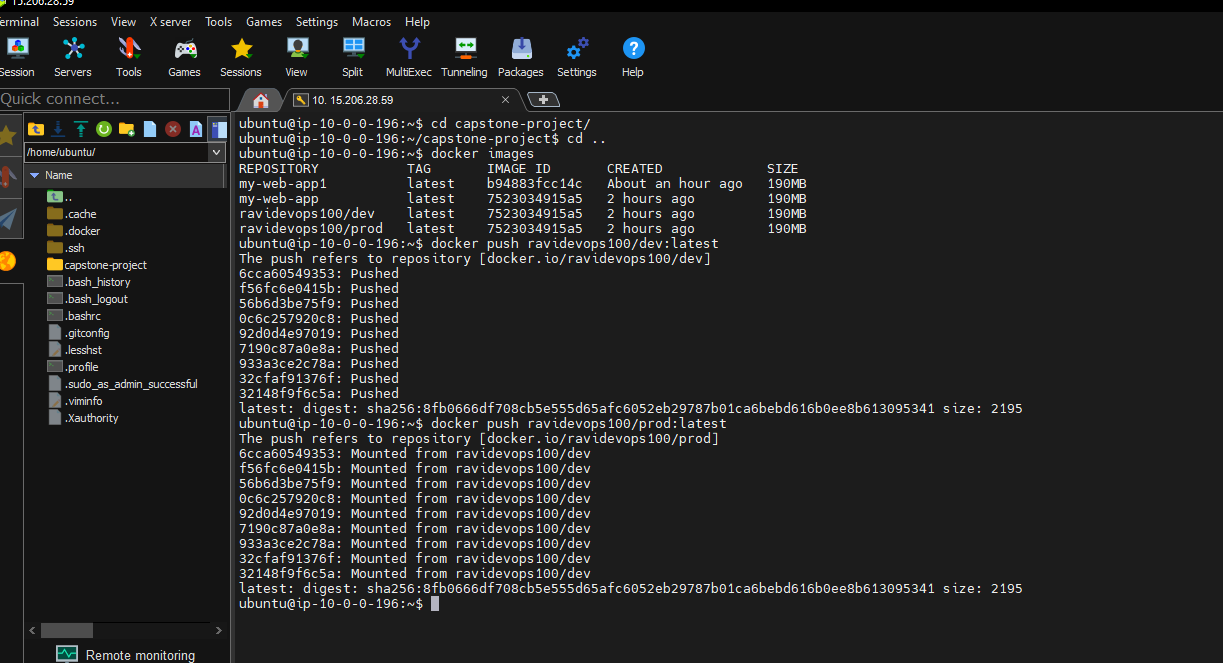
1. Docker tagging the image by command docker tag for 2 repos dev and prod



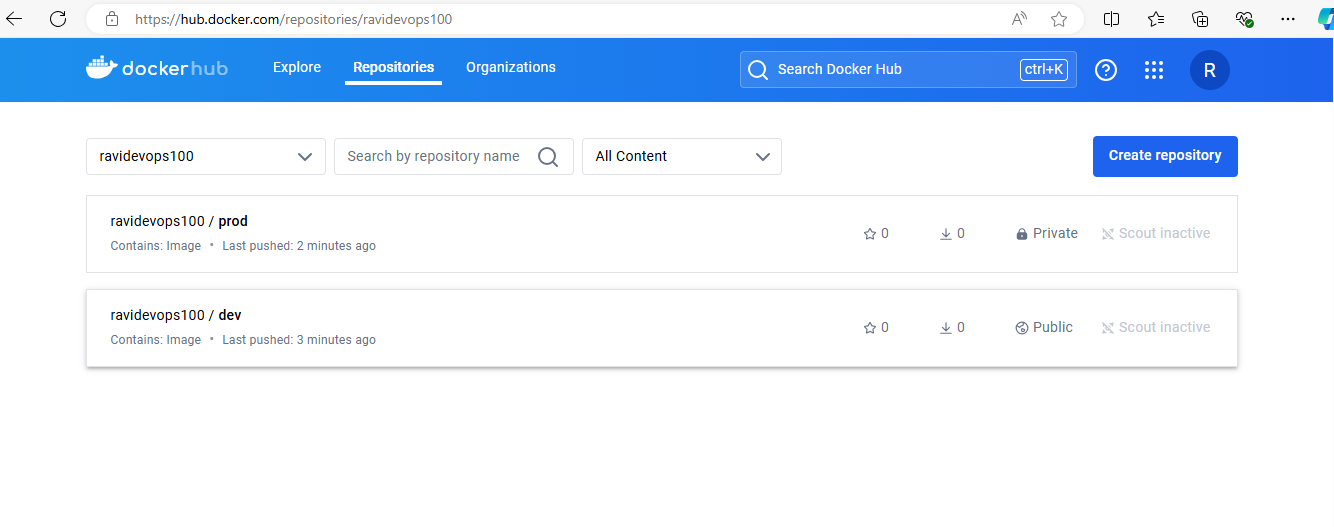
1. Using docker login command to add credentials in server

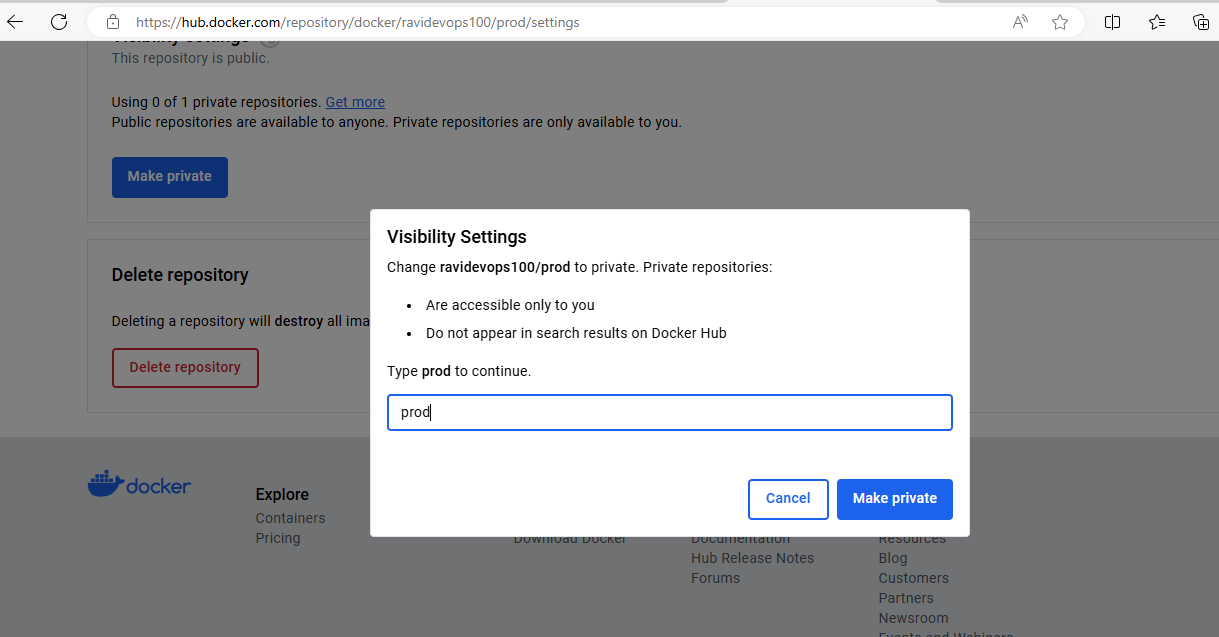


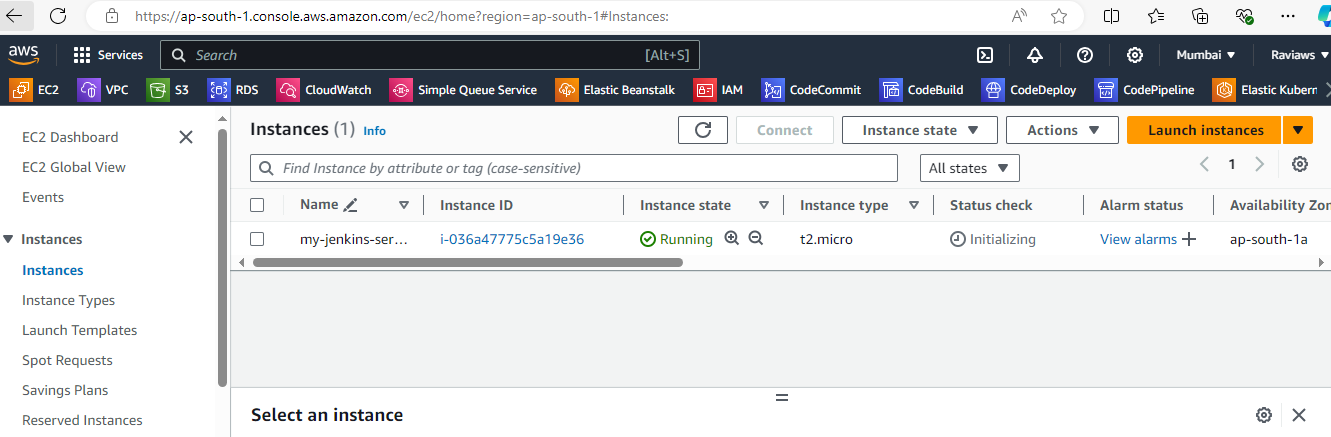
1. Pushed the tagged image to docker hub by docker push command

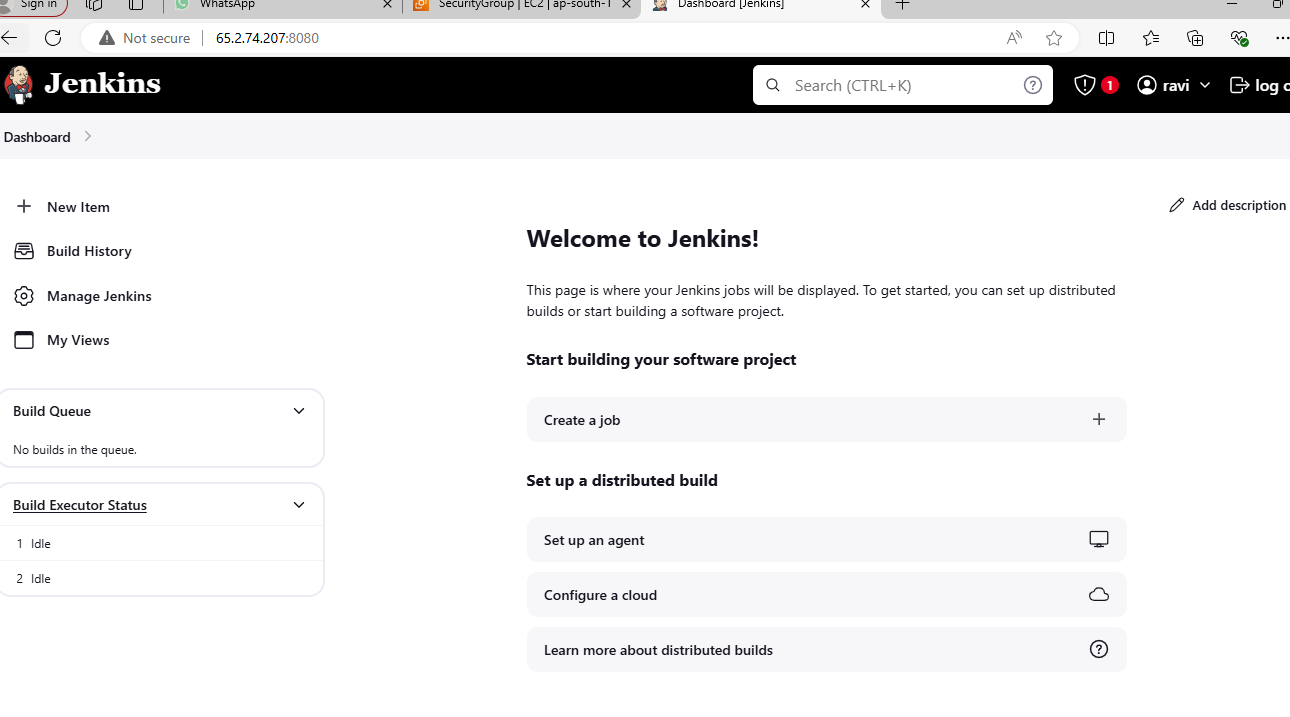
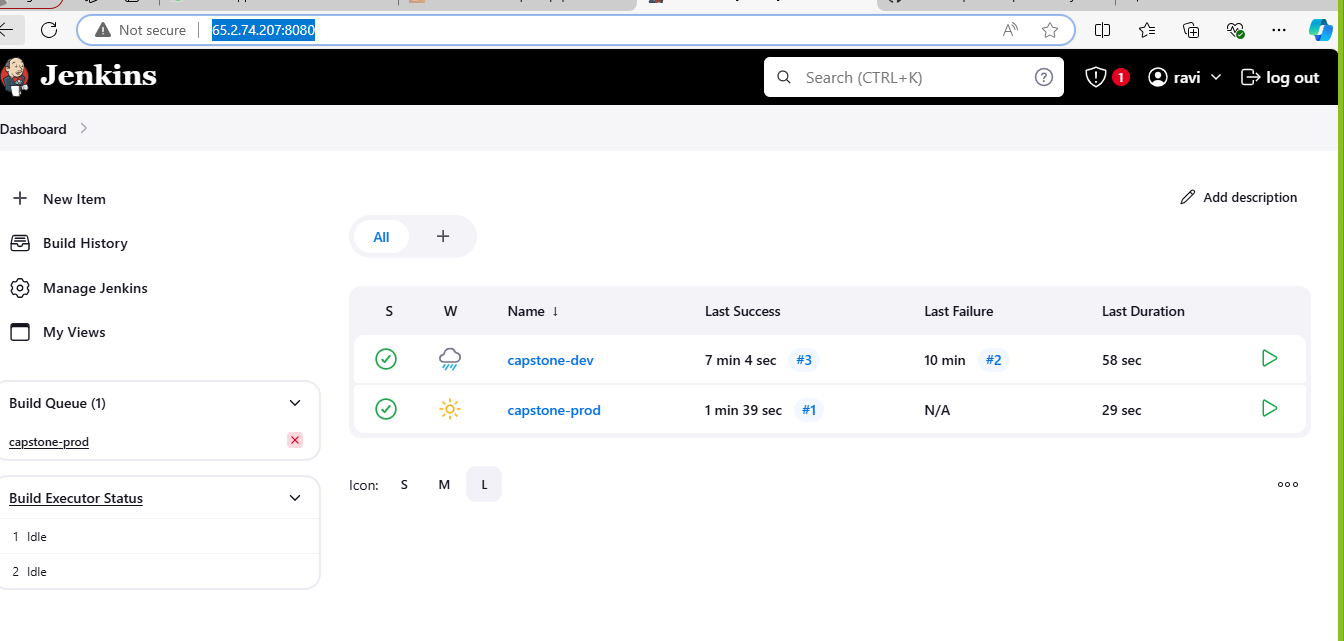


1. Docker images pushed to dockrhub successfully.

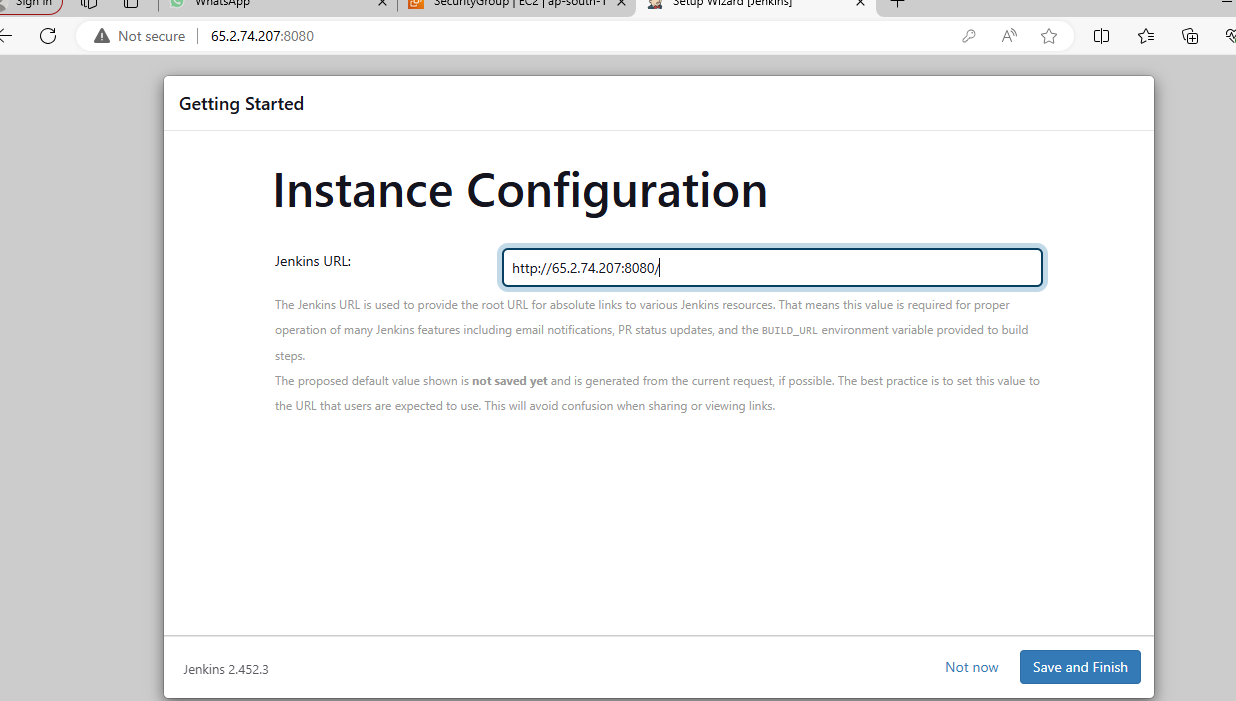


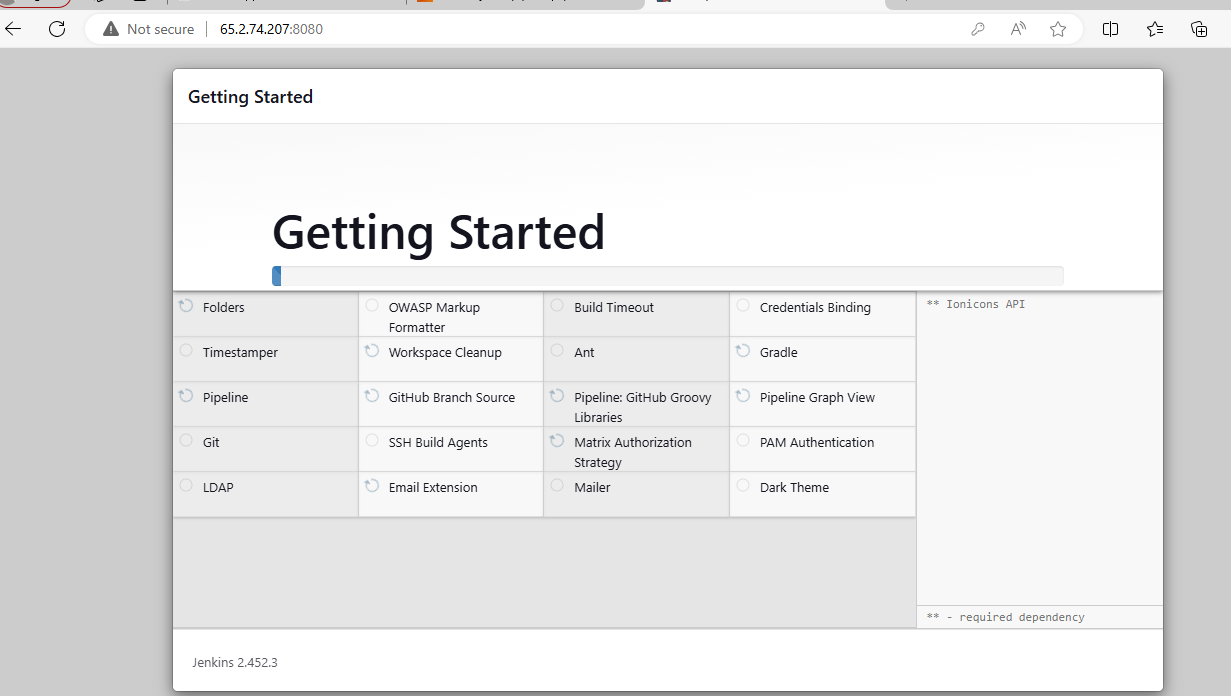
1. Made the prod repo into private
2. **Jenkins**
3. I have created a new ec2 instance for Jenkins inatllation in AWS but after installed and configuration of Jenkins for some times it works and getting very slow due to free tier limit exceeds and so not able to create pipeline project and listed those screen shots below

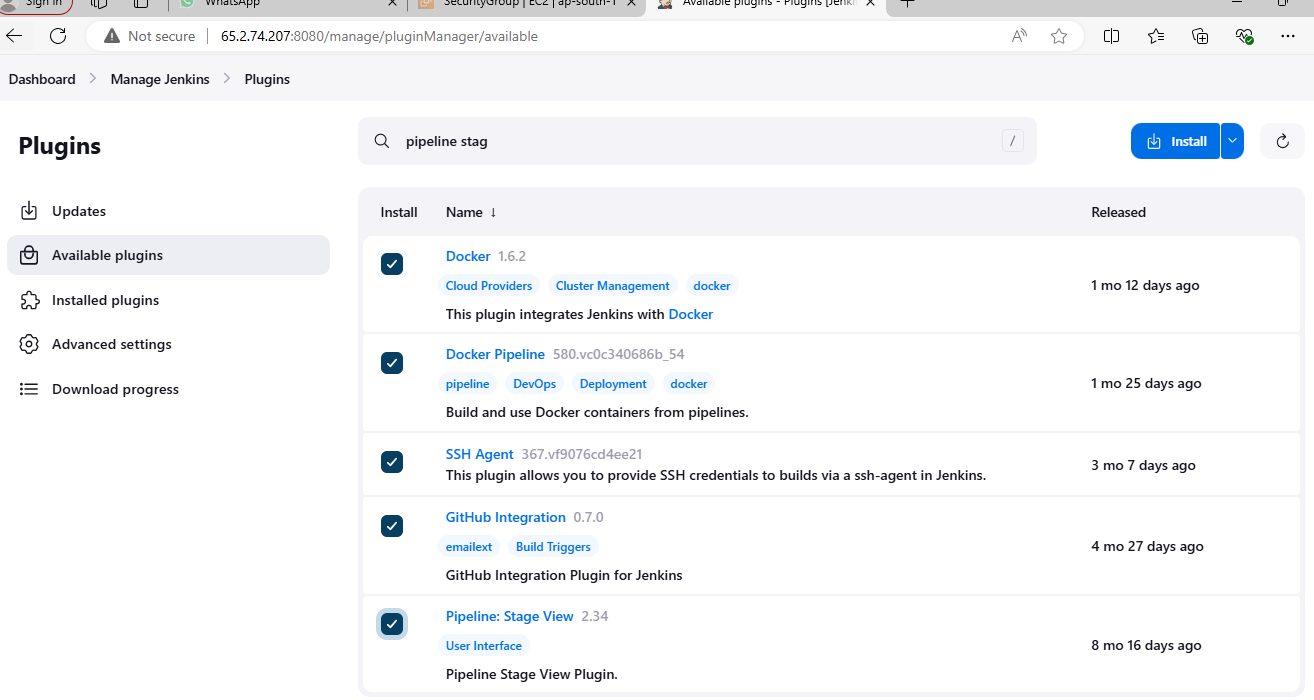






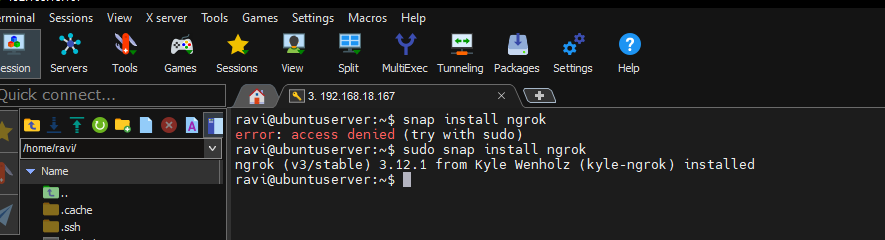




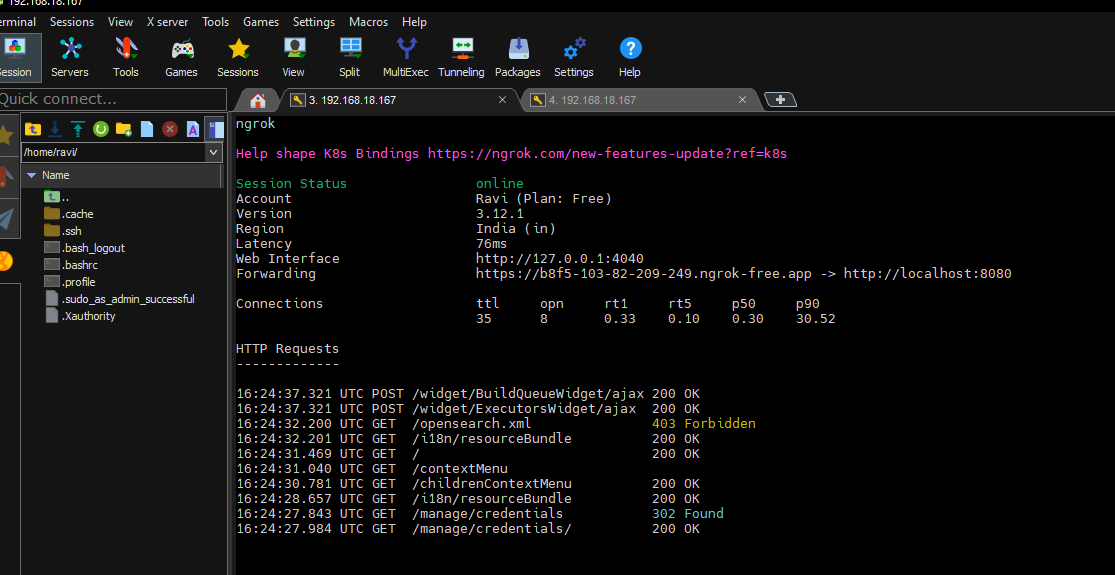


So Finally I Setup a local Ubuntu server and installed Jenkins and made as public using ngrock application and the steps as follows .

Installed ngrock application to make local server to public url

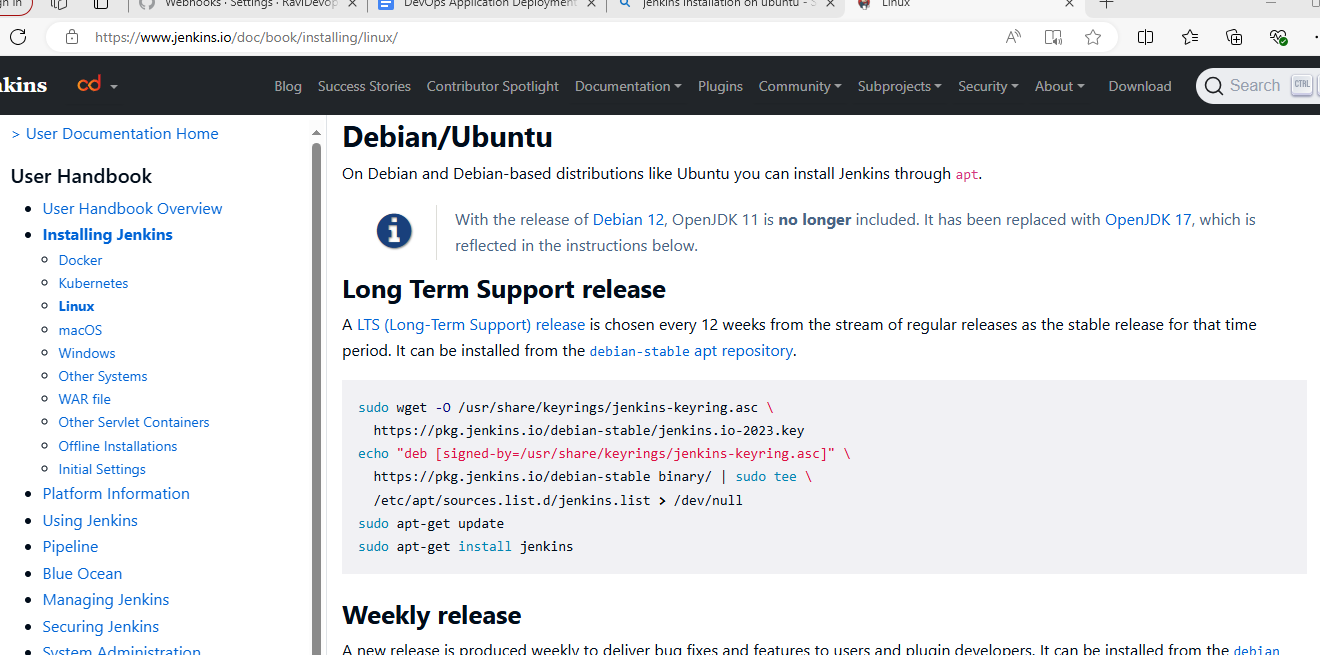


Ngrock public url



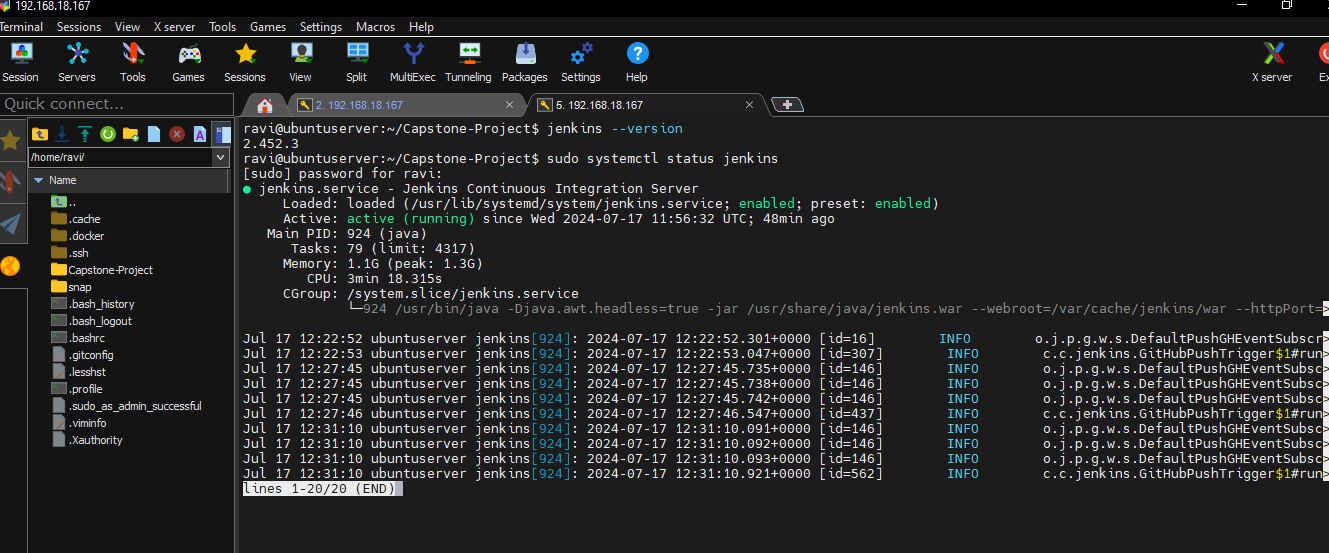
Starting Work with Jenkins.

1. Installed Jenkins with the commands in official documentation

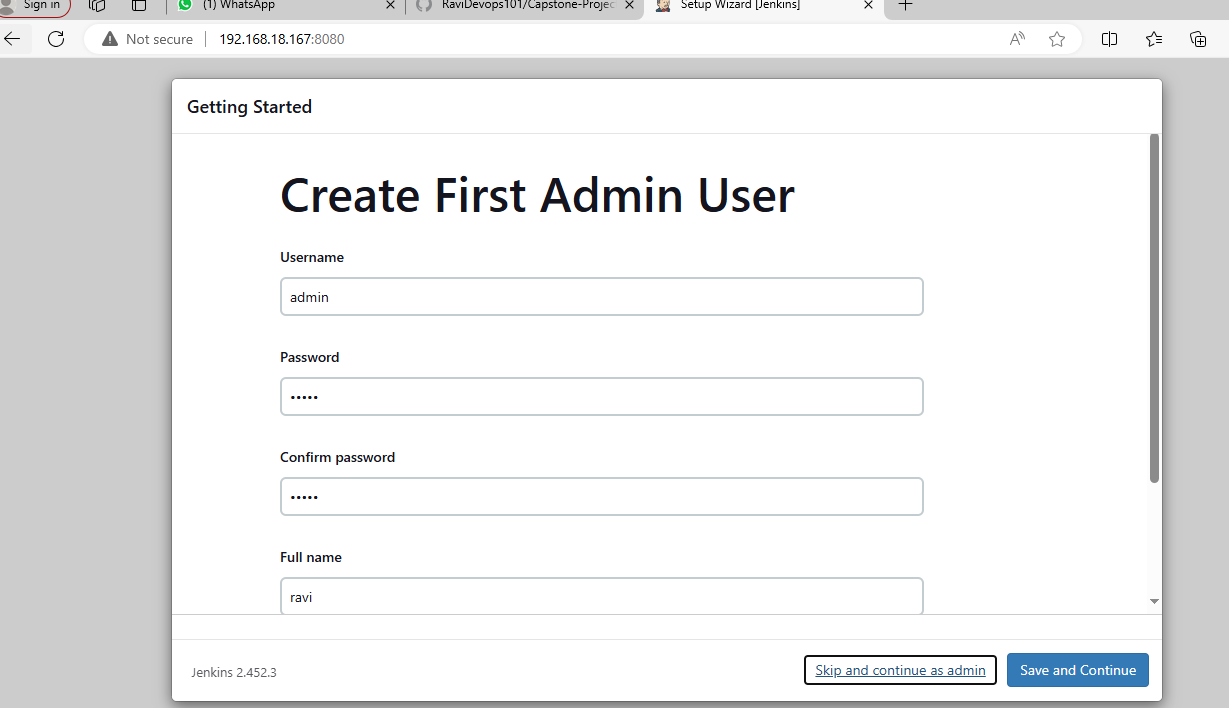


1. Checked Jenkins installation with command Jenkins –version and

Sudo systemctl status Jenkins

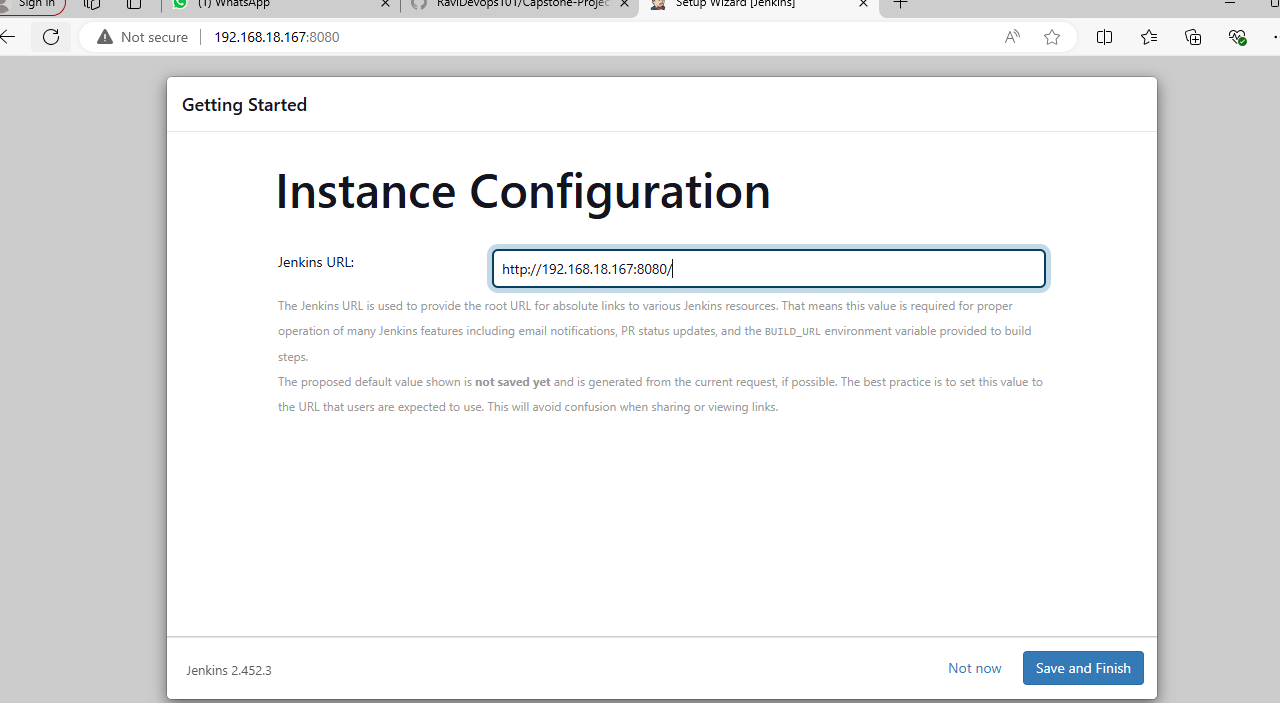


1. Opened Jenkins page in local

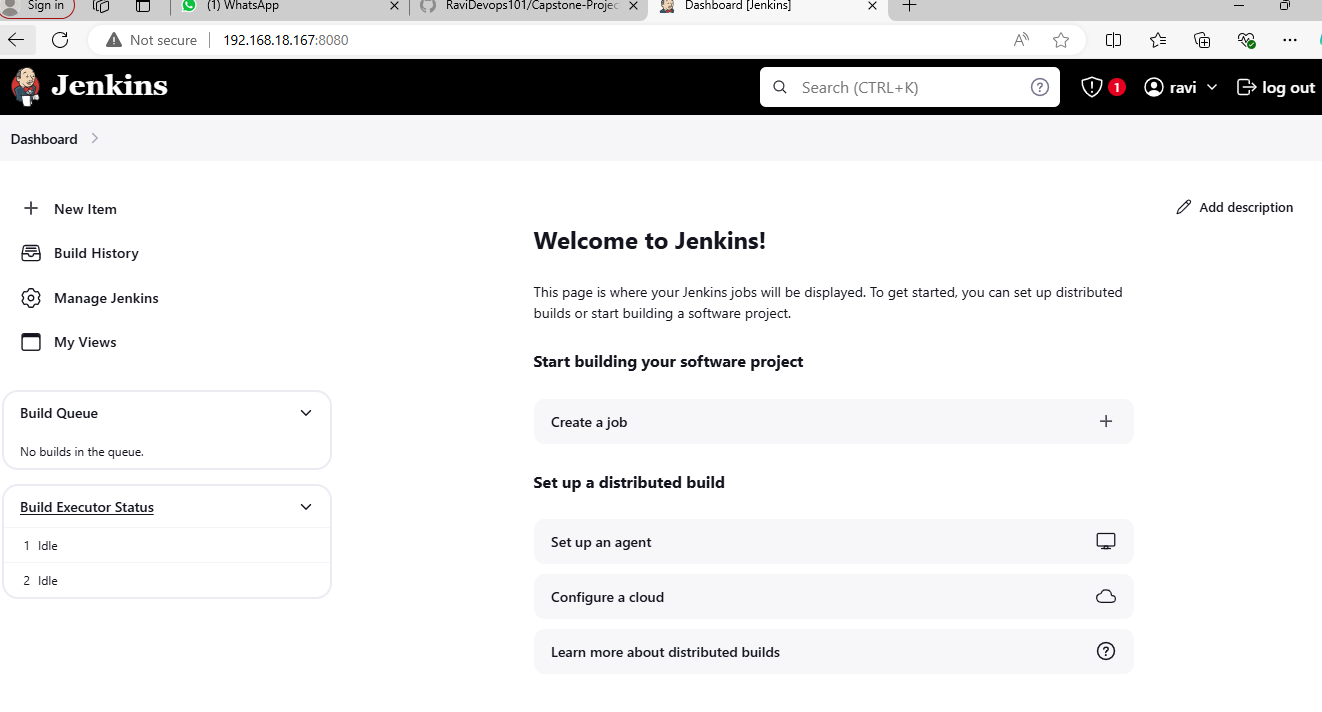




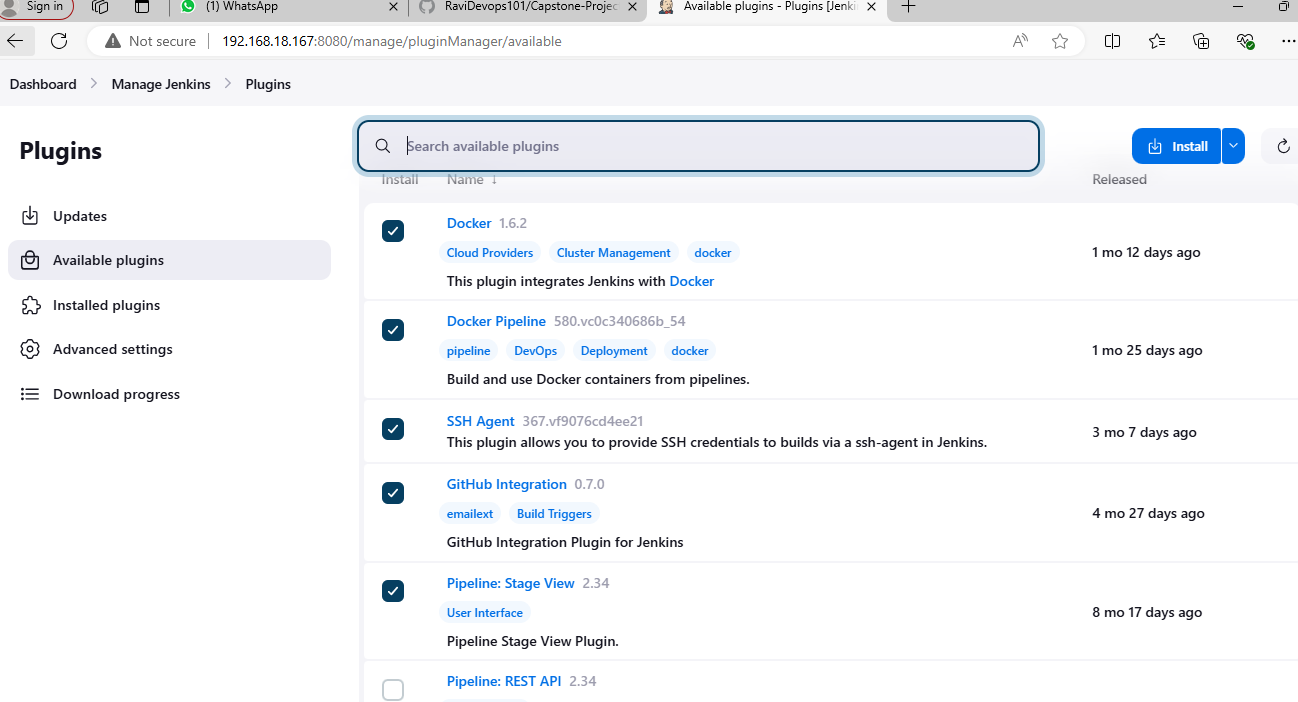




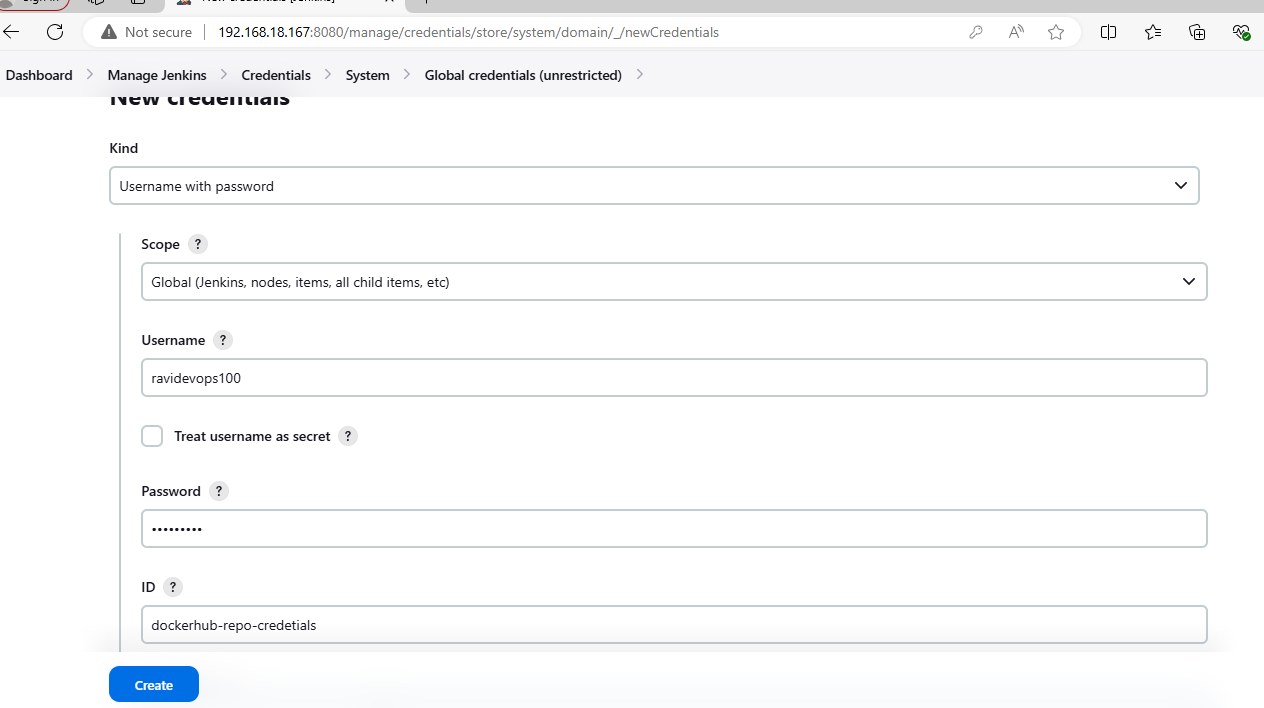
1. Jenkins home page



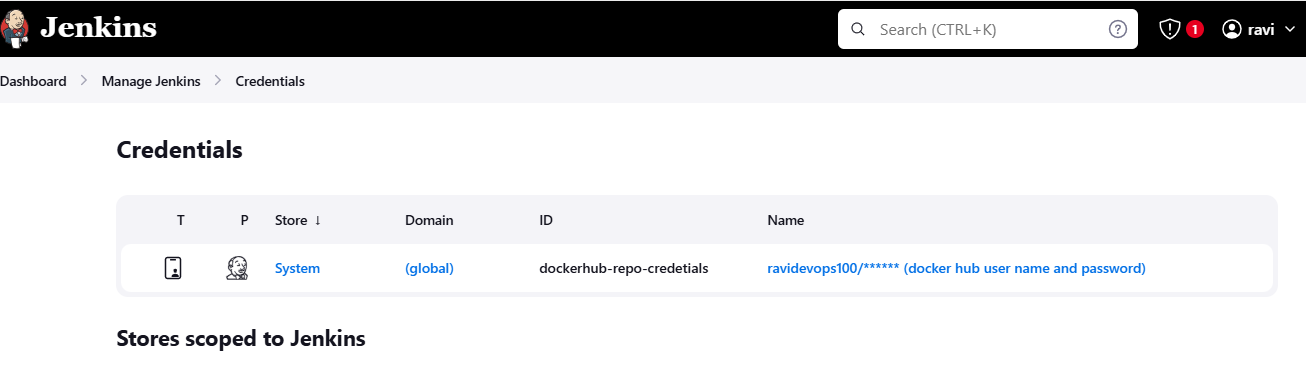
1. Installed plugins for docker github and pipeline in jenkins



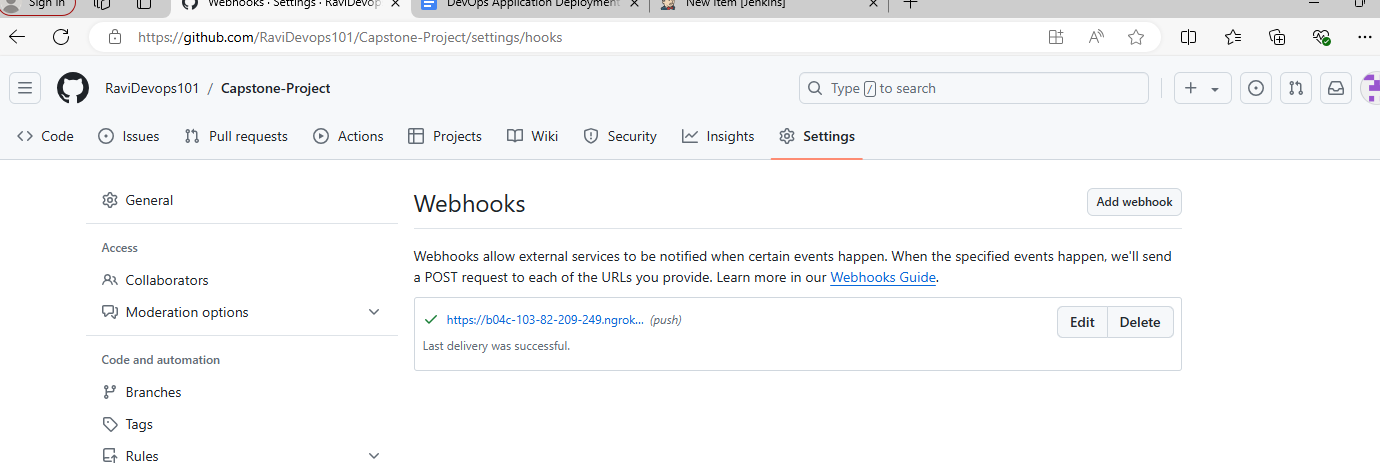
1. Adding docker hub Credntials in Jenkins



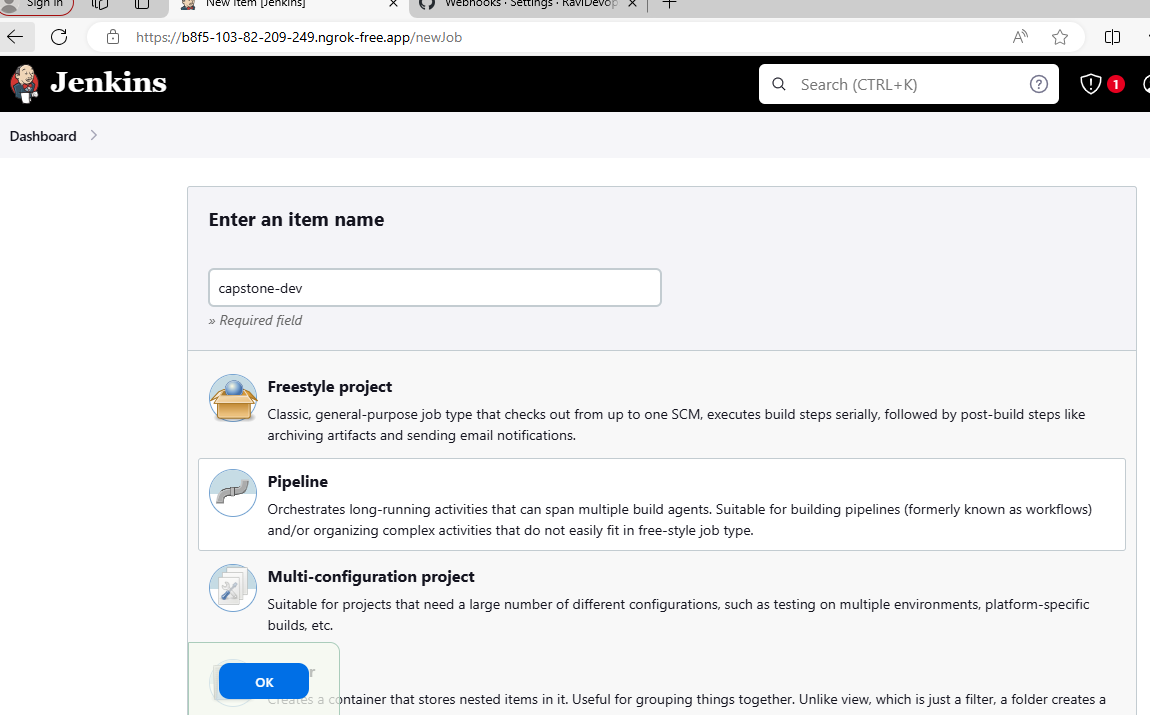
Docker hub credential added

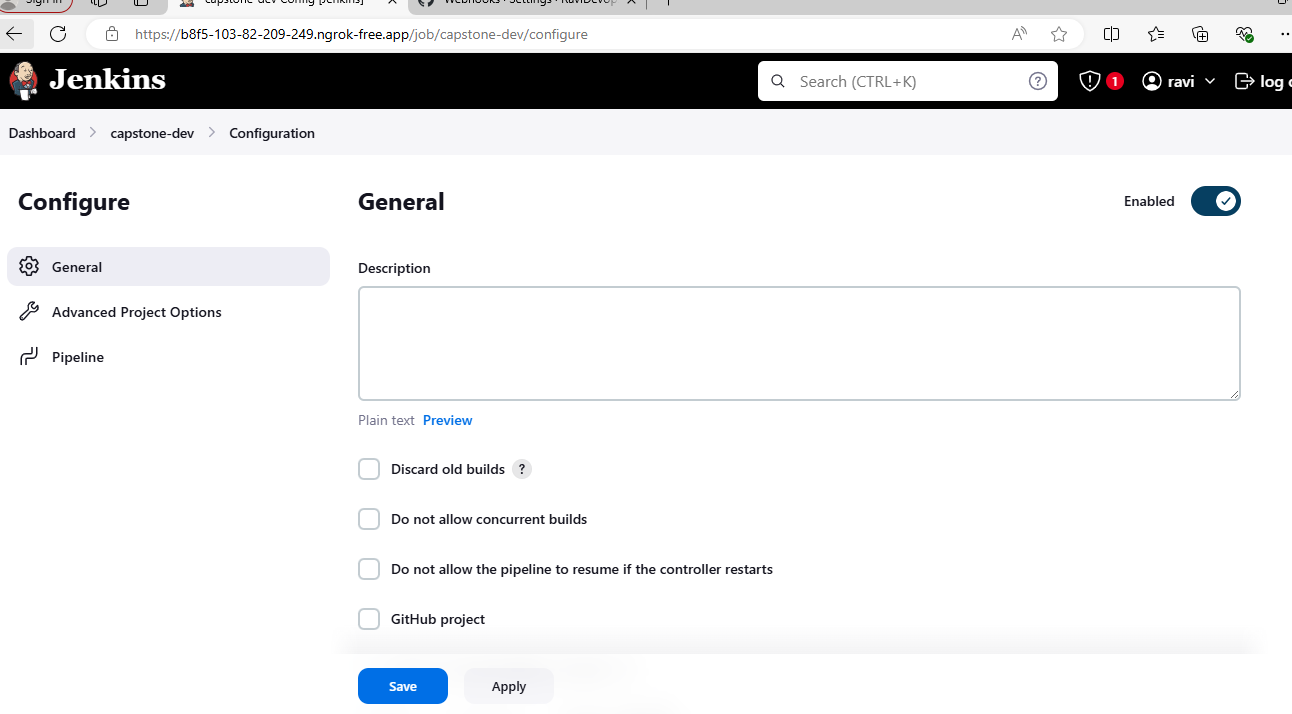


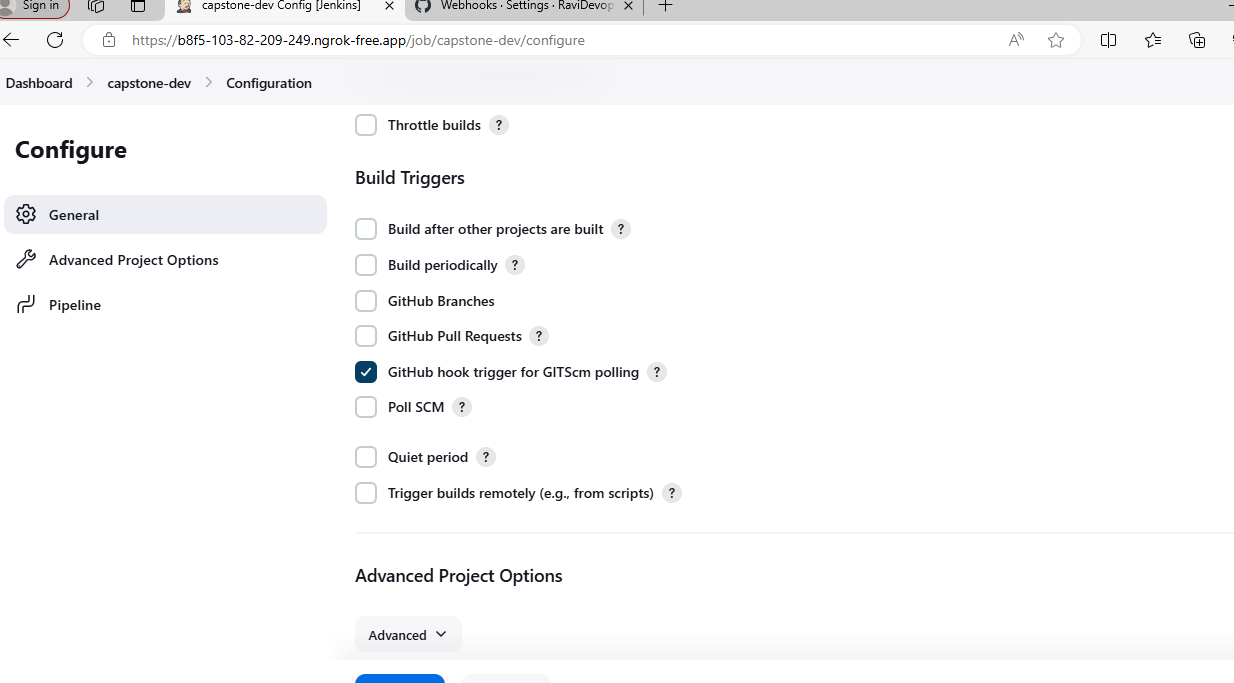
1. Configuring wenhook git hub repo to build auto trigger



1. Creating a new pipline for dev repo and configuring







The pipeline script is as shown below

pipeline {

agent any

environment {

DOCKER\_HUB\_CREDENTIALS = 'dockerhub-repo-credetials'

REPO\_URL = 'https://github.com/RaviDevops101/Capstone-Project.git'

BRANCH = 'dev'

DOCKER\_IMAGE = 'ravidevops100/dev'

CONTAINER\_NAME = 'test-container'

}

stages {

stage('Checkout') {

steps {

cleanWs()

git branch: "${env.BRANCH}", url: "${env.REPO\_URL}"

}

}

stage('Build and Push Docker Image') {

steps {

script {

docker.withRegistry('https://index.docker.io/v1/', "${env.DOCKER\_HUB\_CREDENTIALS}") {

def dockerImage = docker.build("${env.DOCKER\_IMAGE}")

dockerImage.push()

}

}

}

}

stage('Deploy Container') {

steps {

script {

// Stop and remove the container if it exists

sh "docker ps -aqf name=${env.CONTAINER\_NAME} | xargs -r docker stop || true"

sh "docker ps -aqf name=${env.CONTAINER\_NAME} | xargs -r docker rm || true"

// Run the new container

sh "docker run -d -p 80:80 --name ${env.CONTAINER\_NAME} ${env.DOCKER\_IMAGE}"

}

}

}

}

post {

always {

script {

// Clean up stopped containers

sh 'docker container prune -f || true'

// Clean up unused images

sh 'docker image prune -f || true'

// Clean up dangling volumes

sh 'docker volume prune -f || true'

// Clean up dangling networks

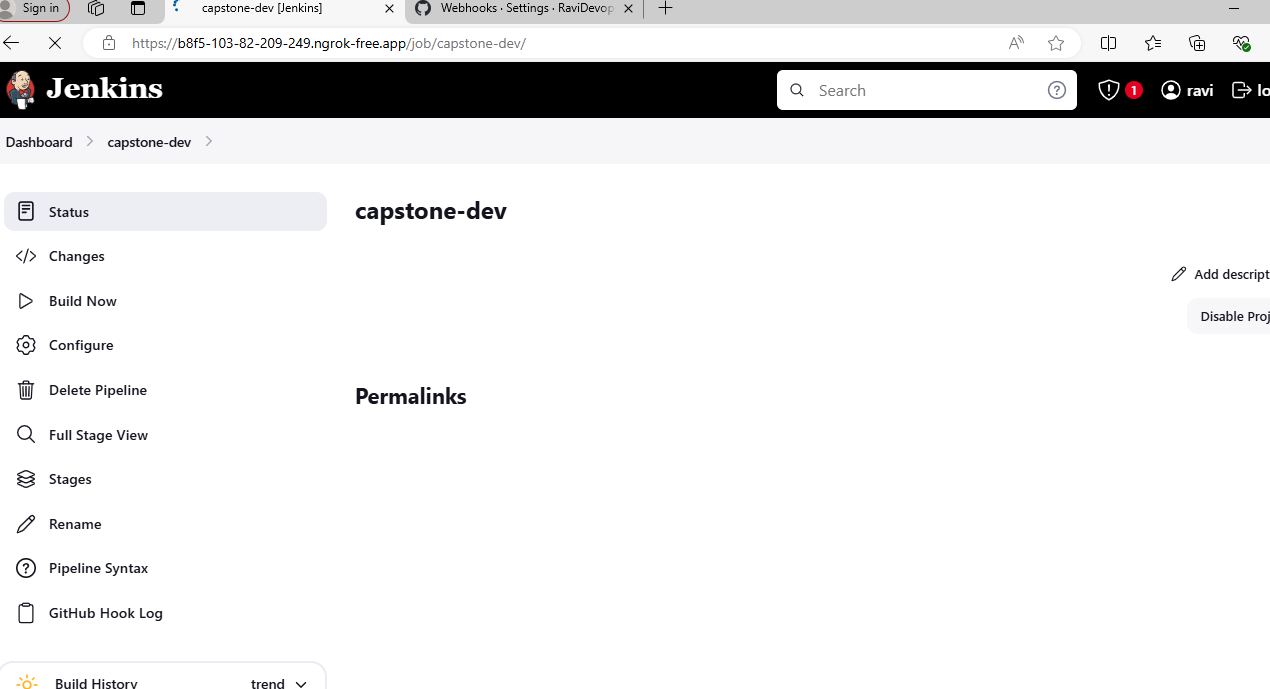
sh 'docker network prune -f || true'

}

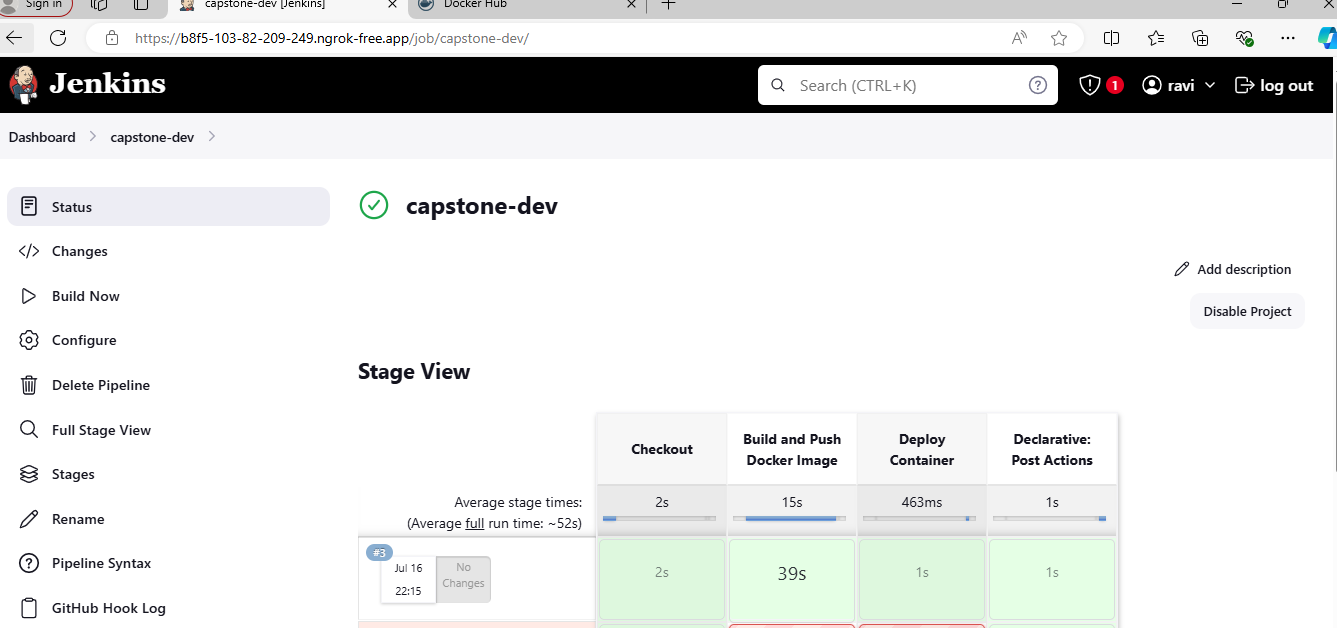
}

}

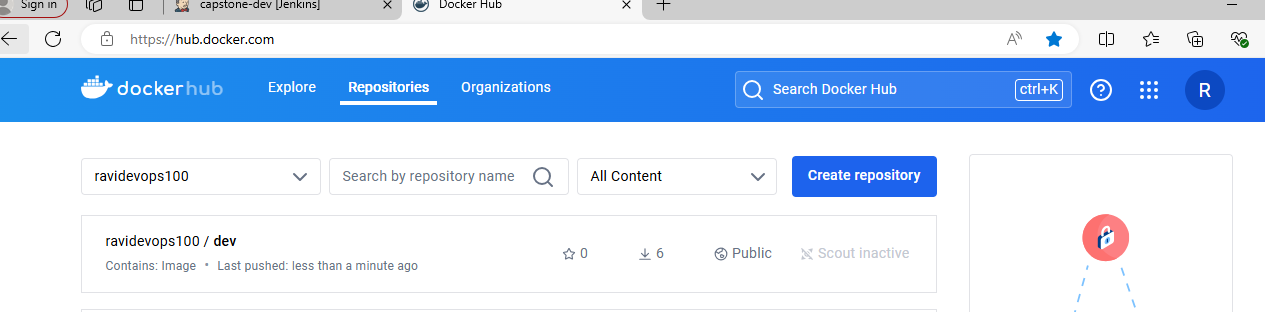
}



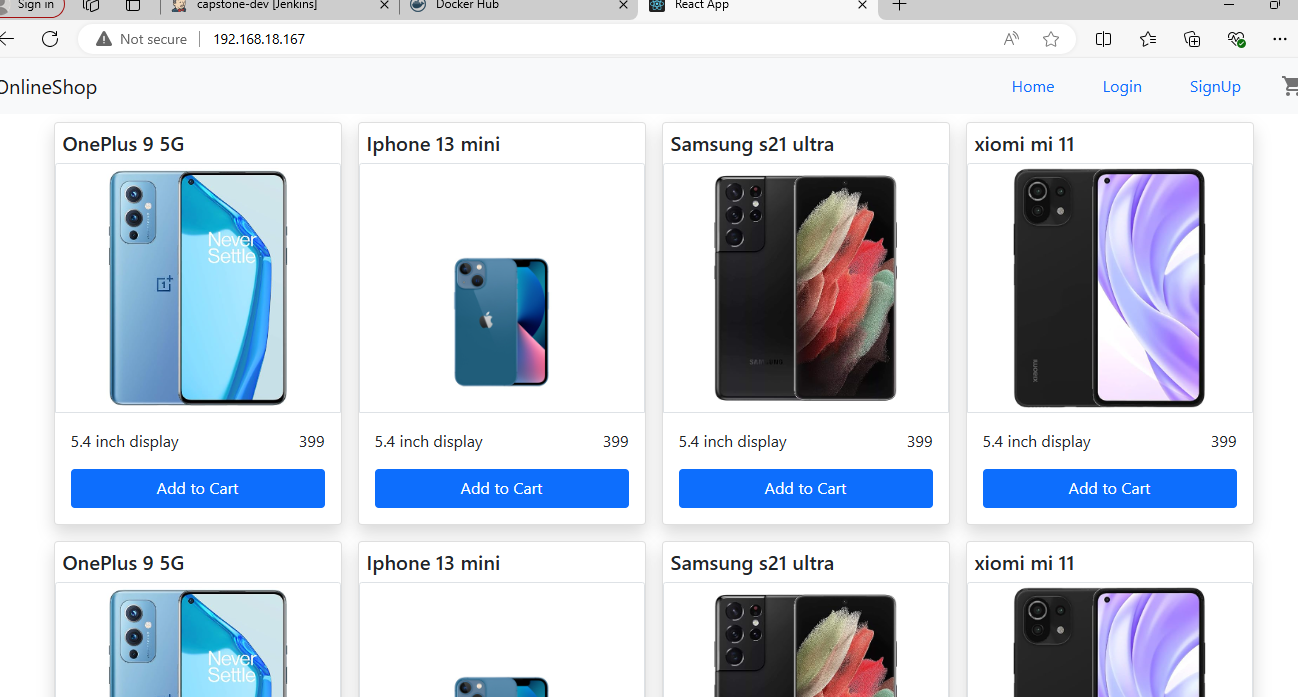
Manually build and tested dev repo

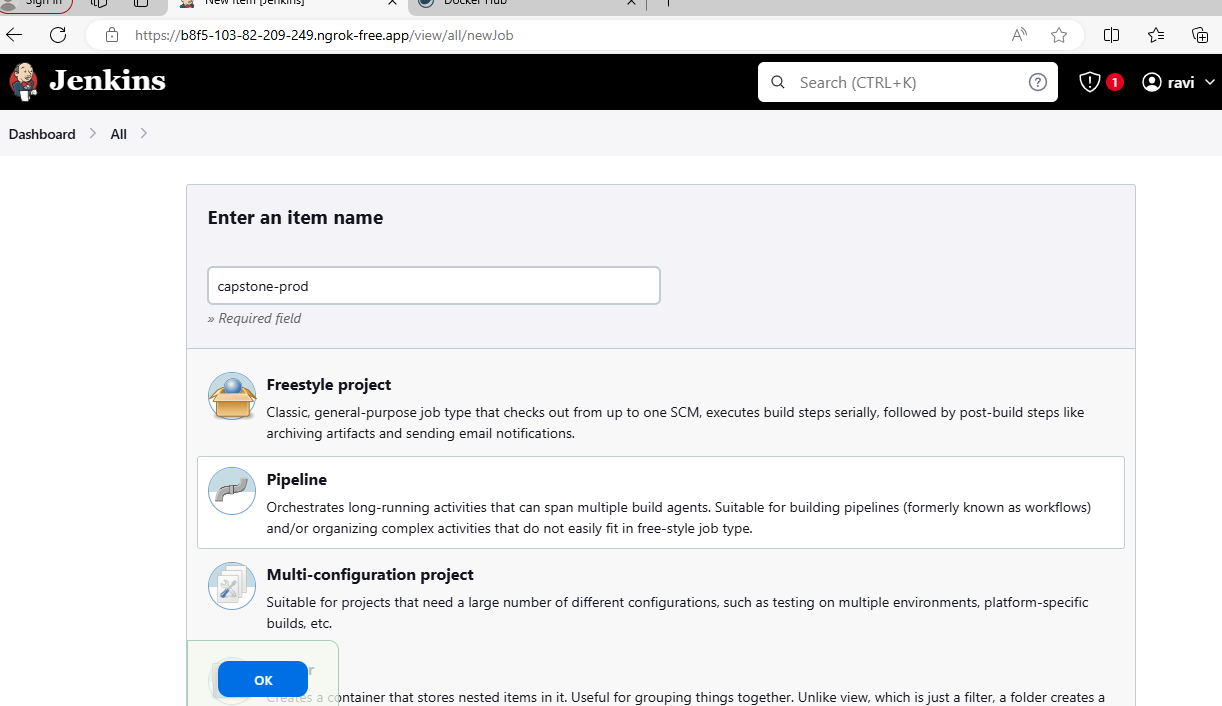


Pushed and Updated in docker hub with manual build

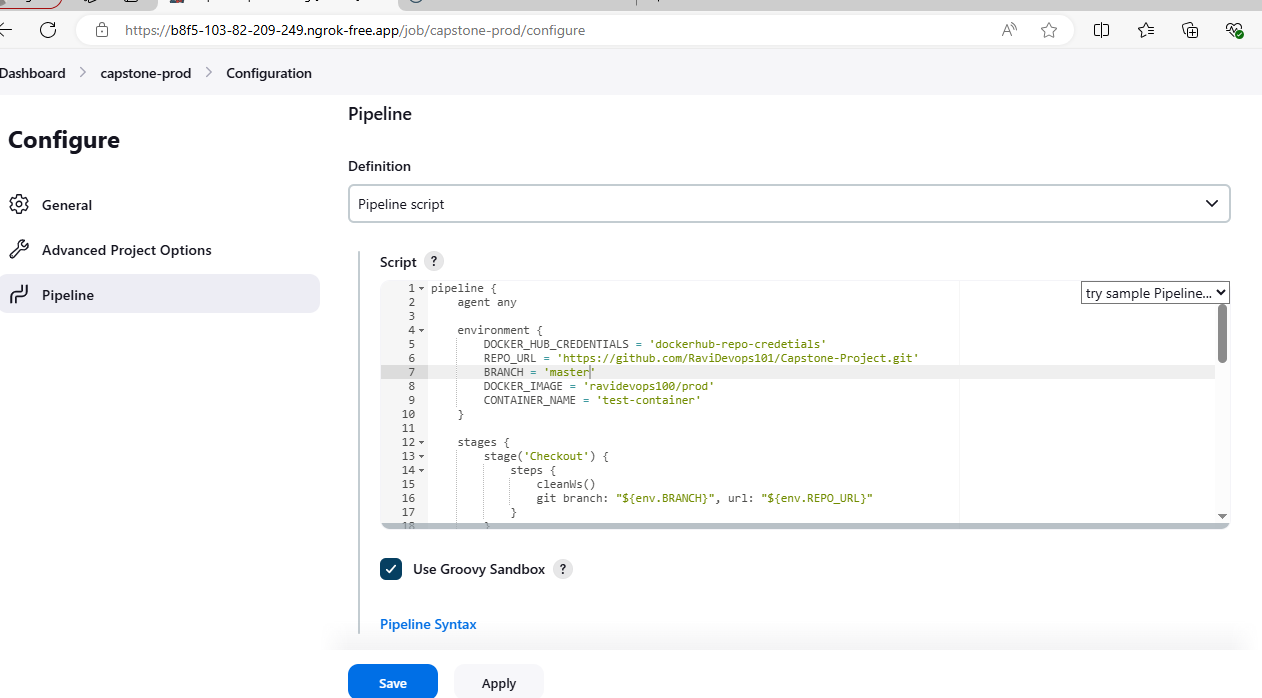


URL Output

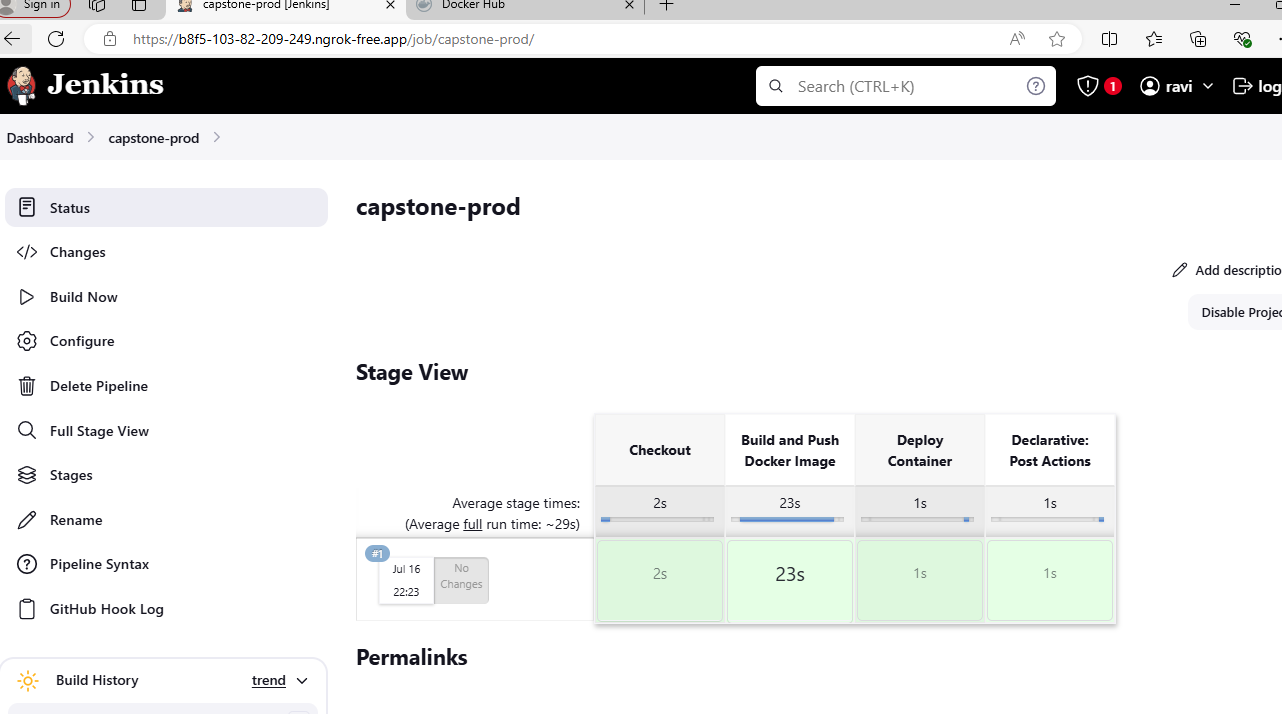


1. Creating a new pipline for prod repo and configuring 

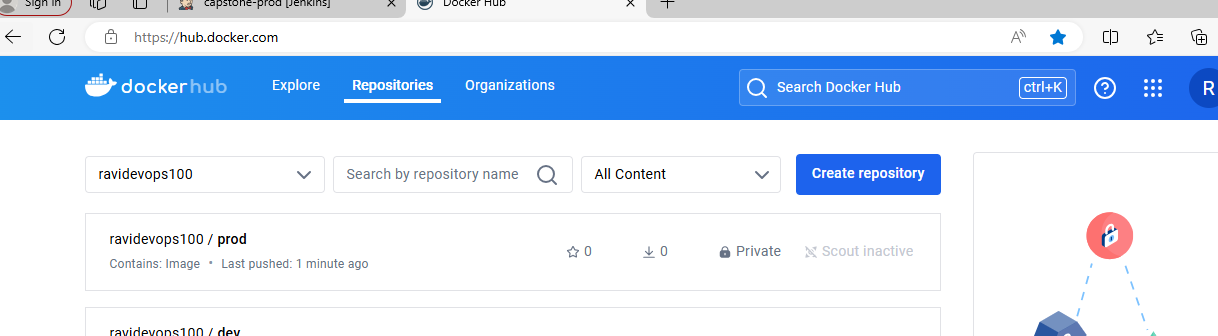
Pipline script for prod repo



Manually build and tested prod repo in Jenkins

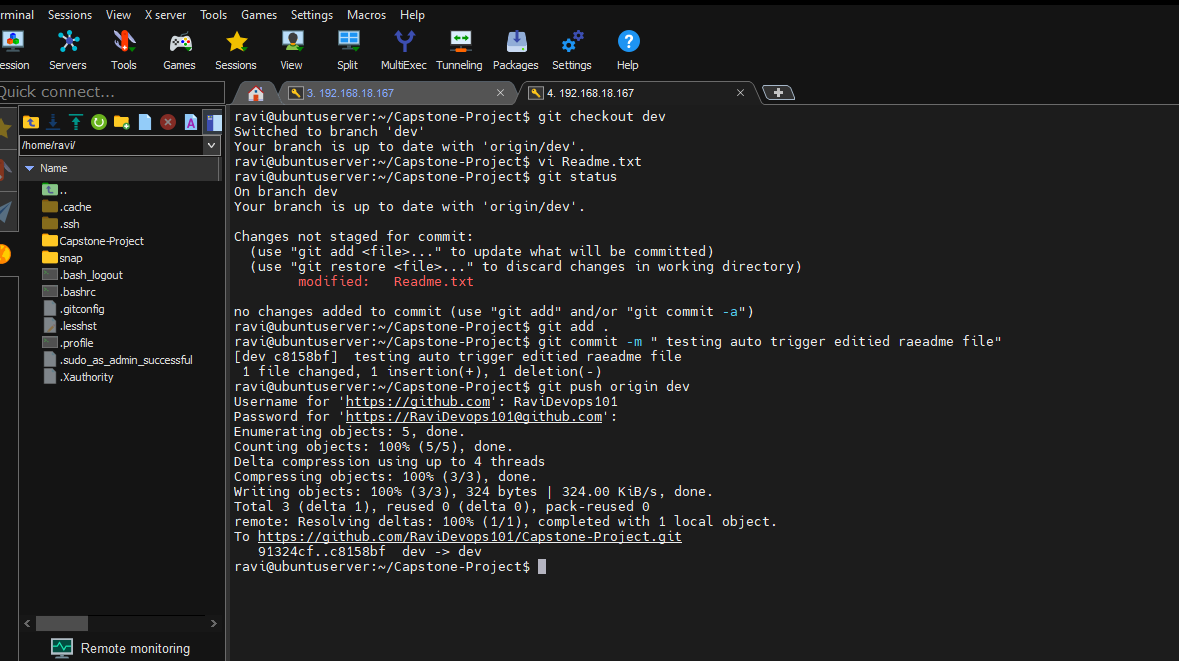


Pushed and updated prod repo in docker hub

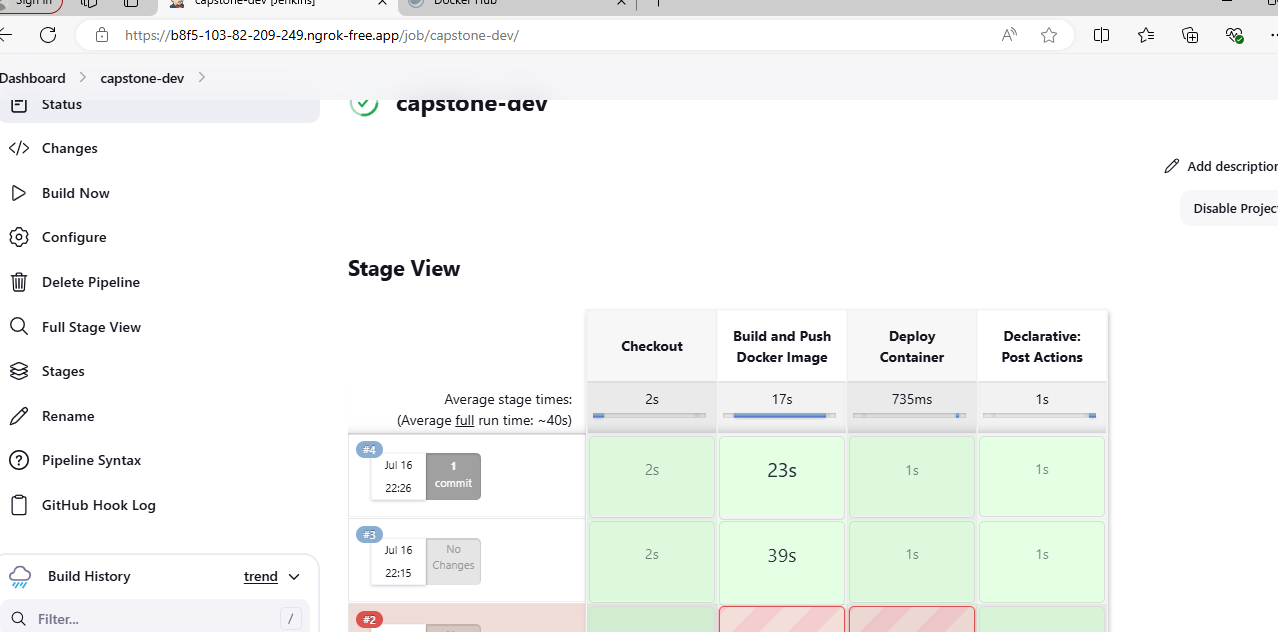


1. Testing auto trigger for dev repo

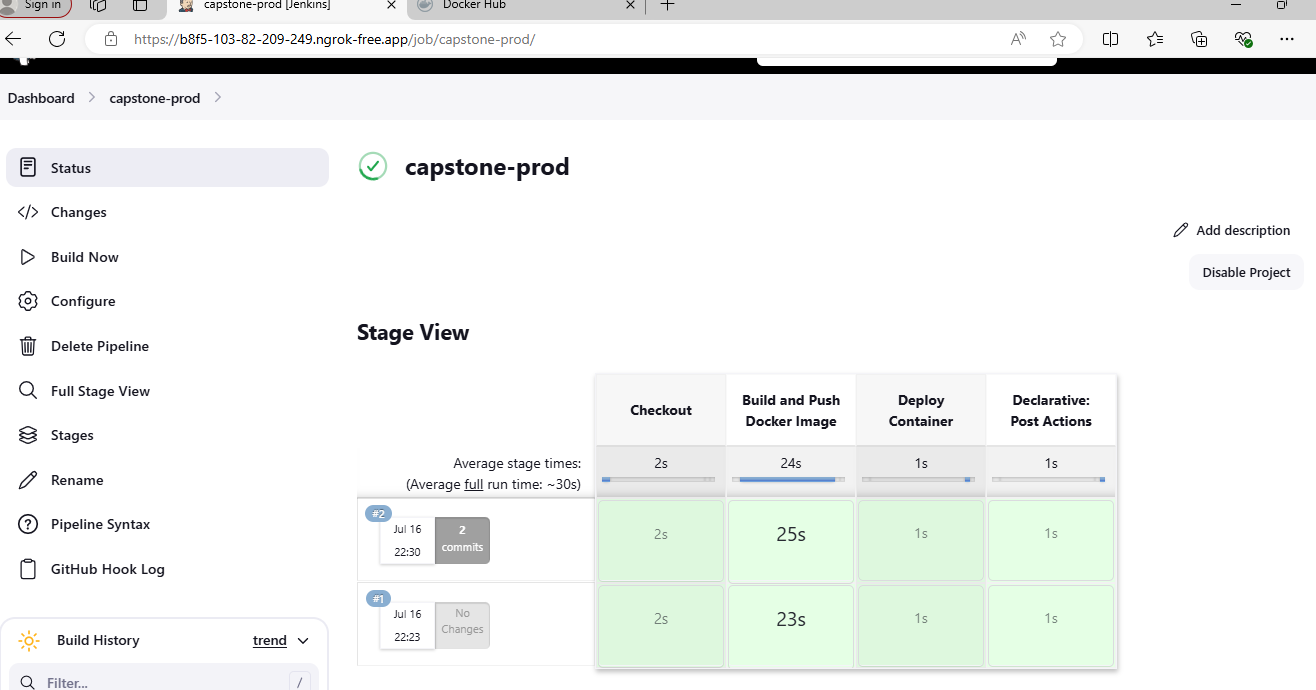
Edited readme file and pushed in git repo



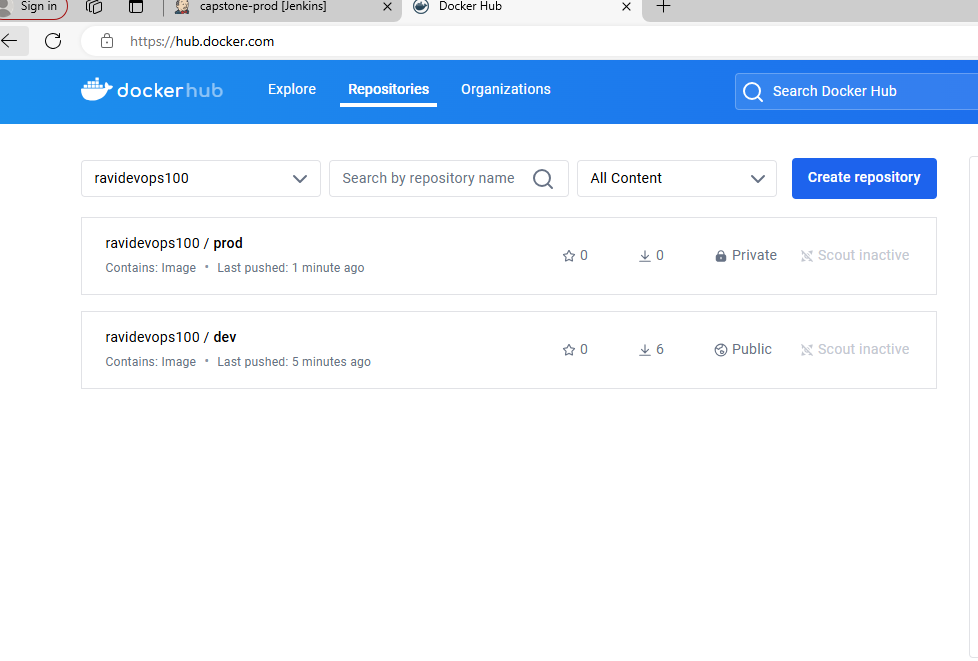
Auto trigger worked in jenkins



Auto trigger for prod repo in jenkins

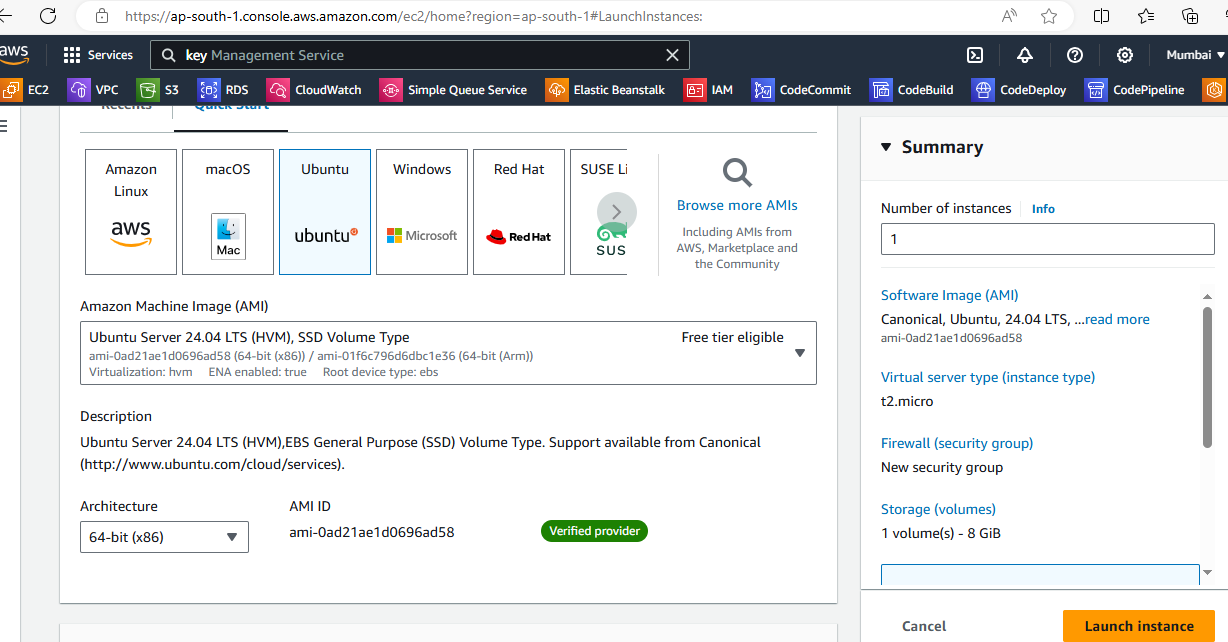


Docker hub repo after auto trigger

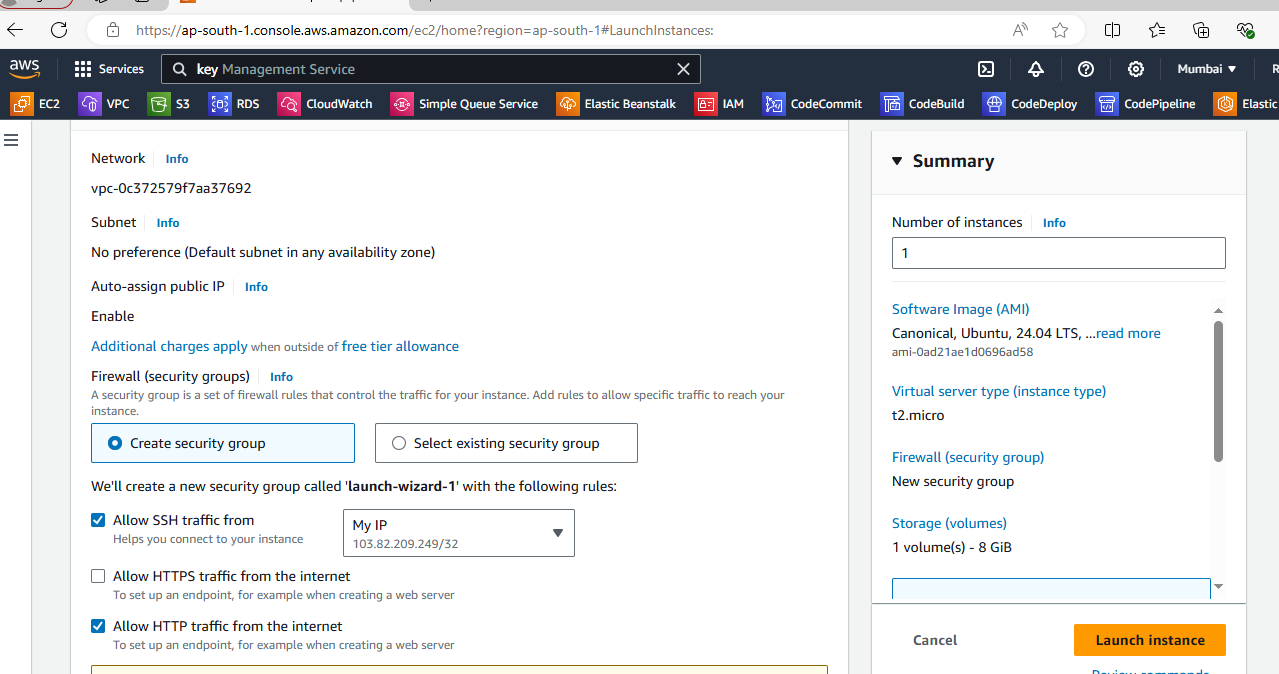


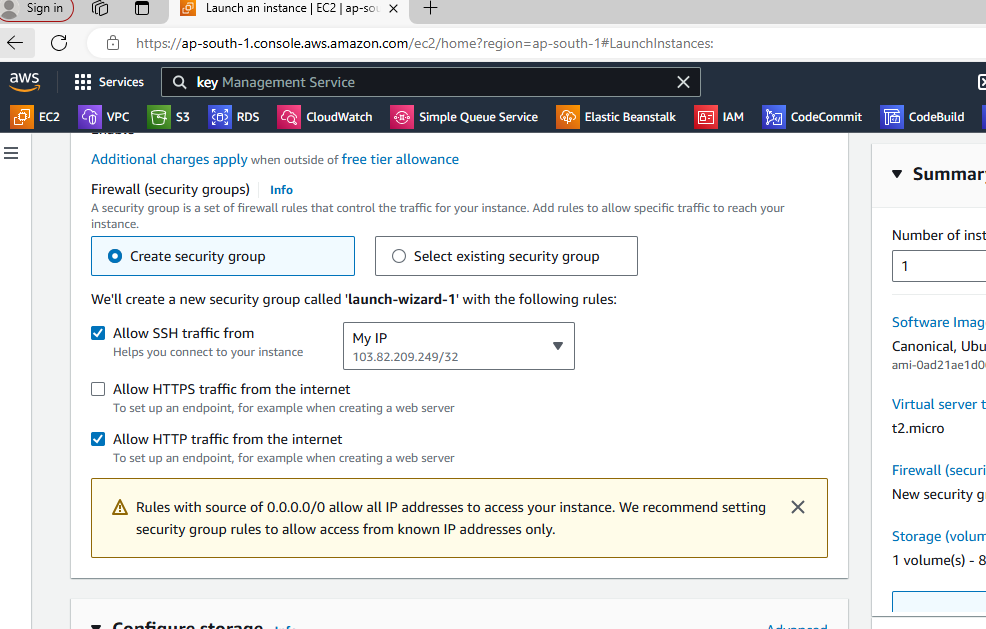
1. **AWS:**
2. Creating ec2 instance in AWS



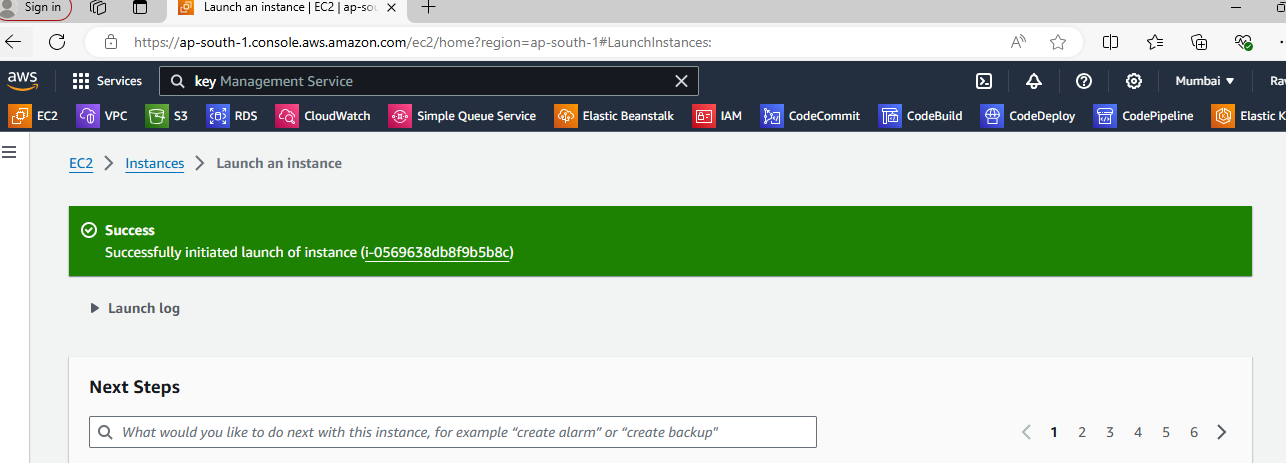


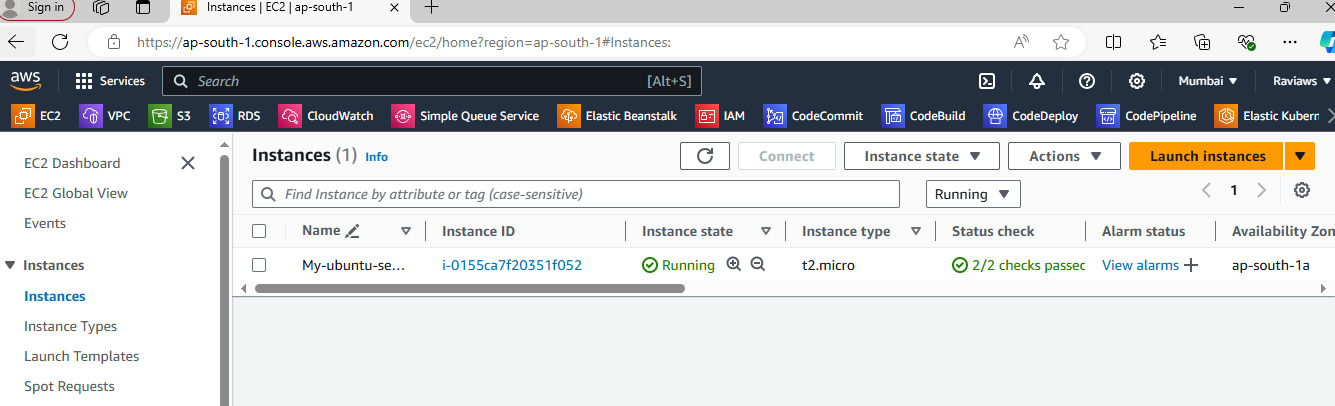
Configured Security Group





Ec2 instance created.





Security group configuration

My ip address in ssh to login to the server

http to access the application

