Step 1: Cloned the repo

Step 2: Created a Docker file (Dockerfile)

# Use NGINX as the base image

FROM nginx:alpine

# Set the working directory

WORKDIR /usr/share/nginx/html

# Copy the static files and index.html into the container

COPY index.html .

COPY static/ ./static/

# Expose the default HTTP port

EXPOSE 80

# Run NGINX in the foreground

CMD ["nginx", "-g", "daemon off;"]

Step 3: Created a docker compose file (docker-compose.yaml) by using the docker image

services:

  web:

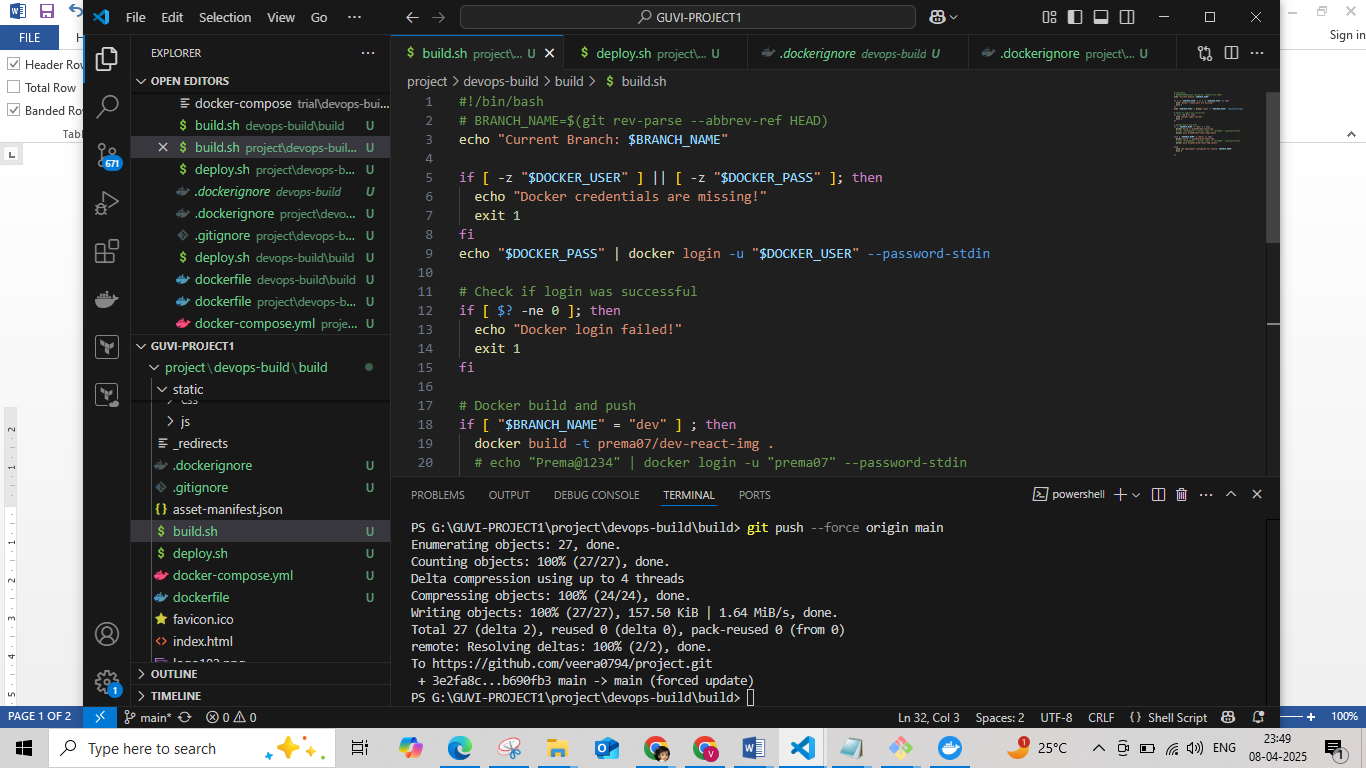
    image: react-app

    ports:

      - "8000:80"

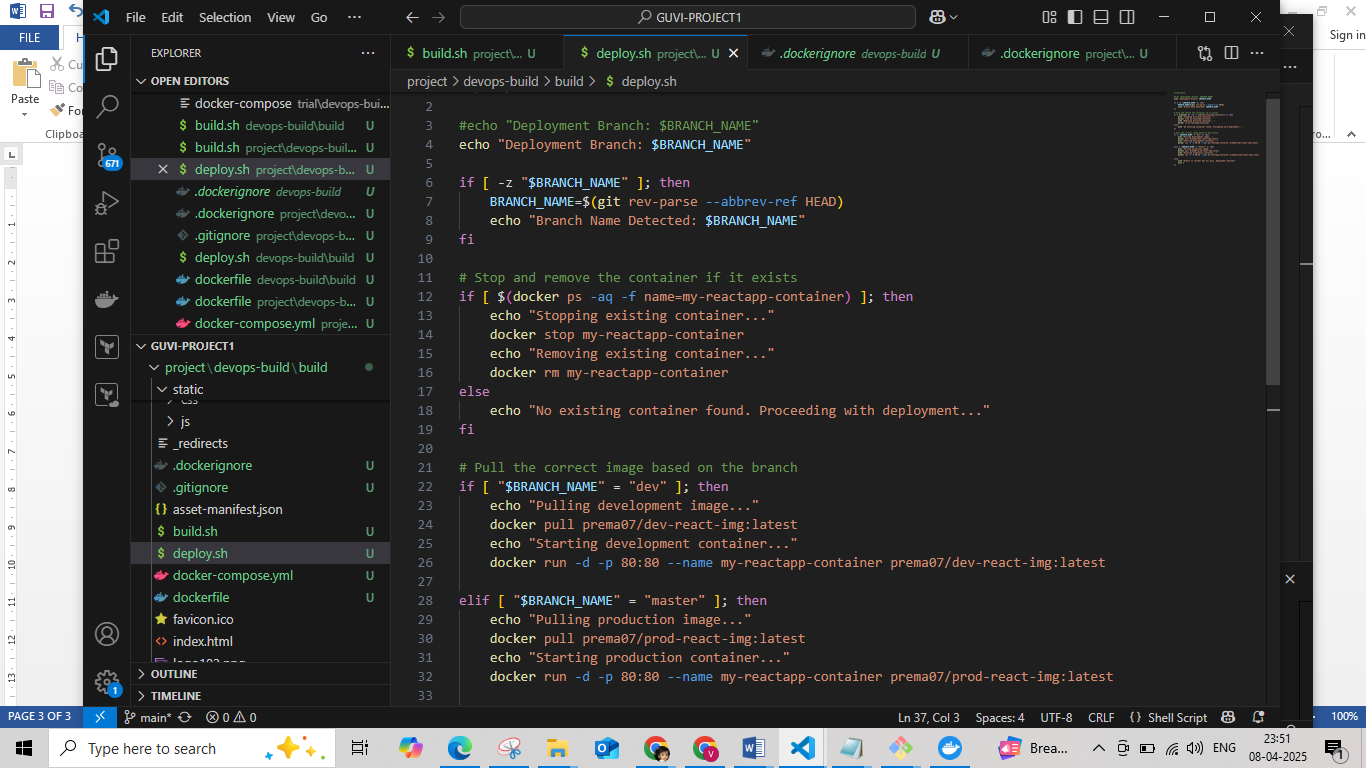
Step 4 : Created a build.sh file for building the docker image

|  |
| --- |
| #!/bin/bash  # BRANCH\_NAME=$(git rev-parse --abbrev-ref HEAD)  echo "Current Branch: $BRANCH\_NAME"  if [ -z "$DOCKER\_USER" ] || [ -z "$DOCKER\_PASS" ]; then  echo "Docker credentials are missing!"  exit 1  fi  echo "$DOCKER\_PASS" | docker login -u "$DOCKER\_USER" --password-stdin  # Check if login was successful  if [ $? -ne 0 ]; then  echo "Docker login failed!"  exit 1  fi  # Docker build and push  if [ "$BRANCH\_NAME" = "dev" ] ; then  docker build -t prema07/dev-react-img .  # echo "Prema@1234" | docker login -u "prema07" --password-stdin  docker push prema07/dev-react-img:latest  elif [ "$BRANCH\_NAME" = "main" ]; then  docker build -t prema07/prod-react-img .  # echo "Prema@1234" | docker login -u "prema07" --password-stdin  docker push prema07/prod-react-img:latest  else  echo "No deployment configured for branch: $BRANCH\_NAME"  exit 0  fi |



Step 5 : Created deploy.sh file for deploying the image to the server

|  |
| --- |
| #!/bin/bash  #echo "Deployment Branch: $BRANCH\_NAME"  echo "Deployment Branch: $BRANCH\_NAME"  if [ -z "$BRANCH\_NAME" ]; then  BRANCH\_NAME=$(git rev-parse --abbrev-ref HEAD)  echo "Branch Name Detected: $BRANCH\_NAME"  fi  # Stop and remove the container if it exists  if [ $(docker ps -aq -f name=my-reactapp-container) ]; then  echo "Stopping existing container..."  docker stop my-reactapp-container  echo "Removing existing container..."  docker rm my-reactapp-container  else  echo "No existing container found. Proceeding with deployment..."  fi  # Pull the correct image based on the branch  if [ "$BRANCH\_NAME" = "dev" ]; then  echo "Pulling development image..."  docker pull prema07/dev-react-img:latest  echo "Starting development container..."  docker run -d -p 80:80 --name my-reactapp-container prema07/dev-react-img:latest  elif [ "$BRANCH\_NAME" = "master" ]; then  echo "Pulling production image..."  docker pull prema07/prod-react-img:latest  echo "Starting production container..."  docker run -d -p 80:80 --name my-reactapp-container prema07/prod-react-img:latest  else  echo "Branch is neither dev nor main. Deployment aborted."  exit 1  fi |



Step 6 : Created .gitignore and .dockerignore file

.gitignore file

# Node modules

node\_modules/

# Build output

build/

dist/

static/js/\*.map

static/css/\*.map

# Log files

npm-debug.log

yarn-debug.log

yarn-error.log

\*.log

# Environment files

.env

.env.local

.env.development

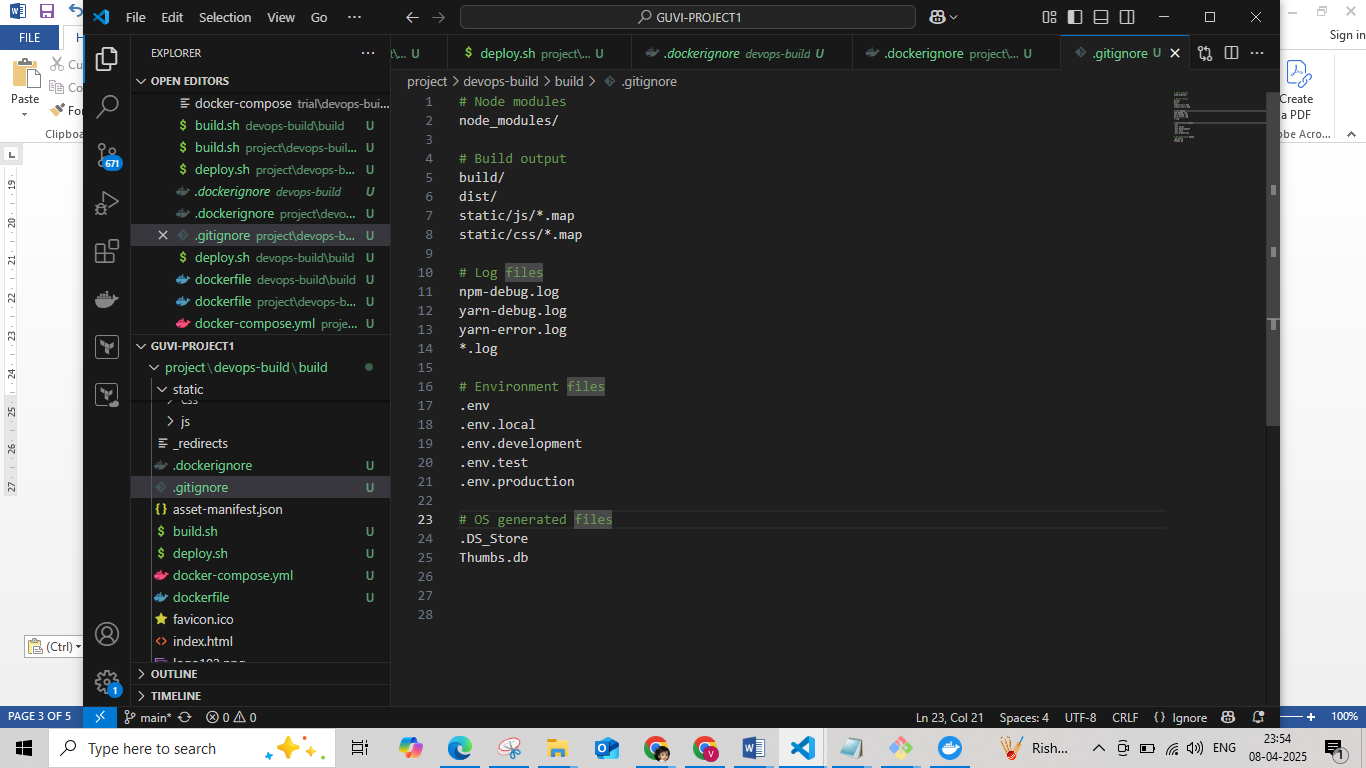
.env.test

.env.production

# OS generated files

.DS\_Store

Thumbs.db



.dockerignore file

# Ignore logs

\*.log

# Ignore Docker-related files

Dockerfile

.dockerignore

# Ignore build artifacts (if applicable)

build

dist

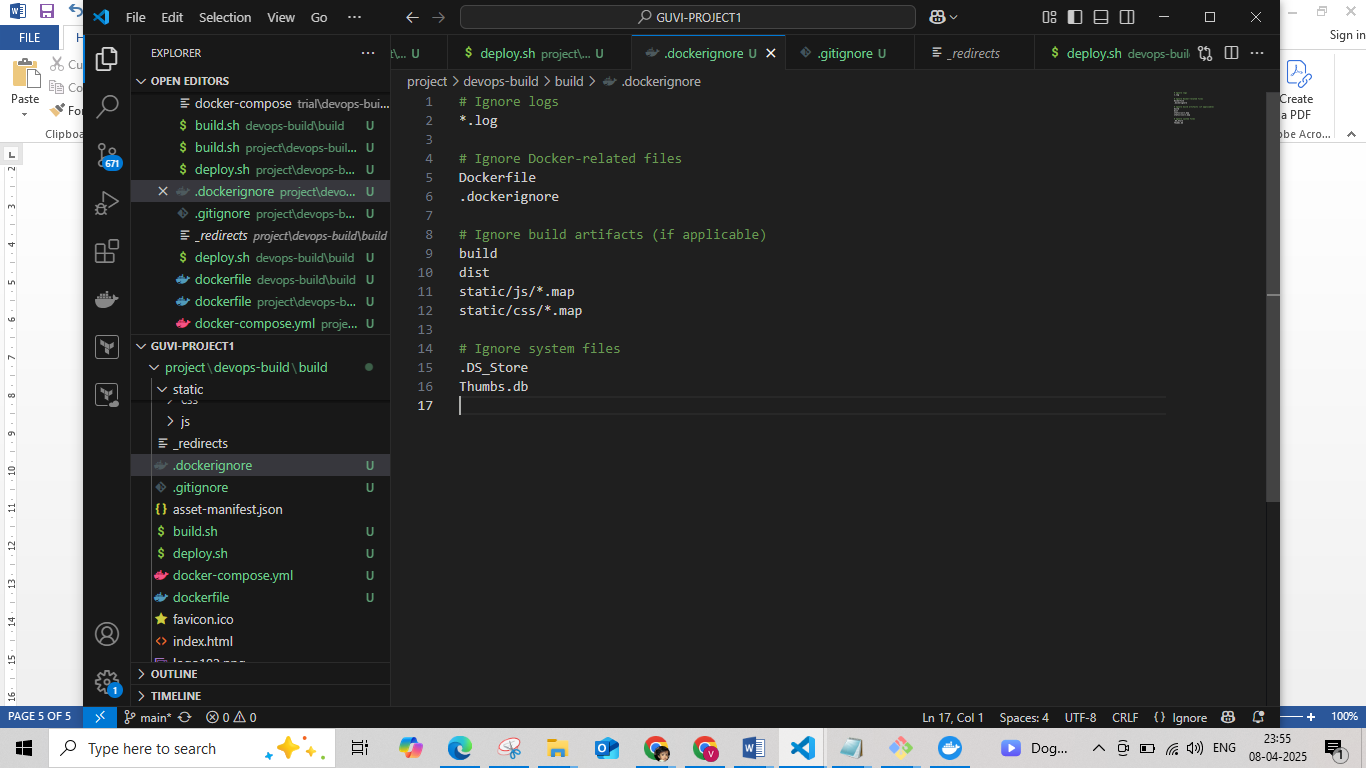
static/js/\*.map

static/css/\*.map

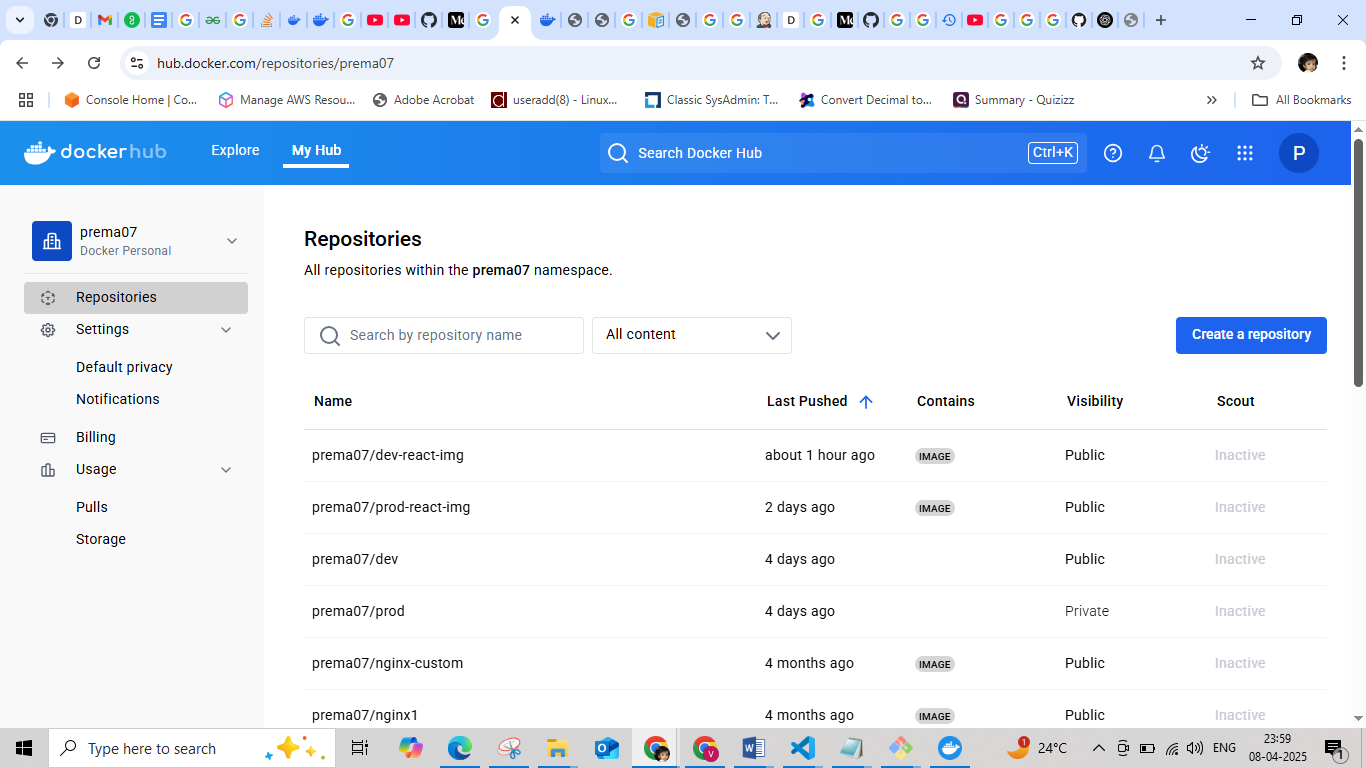
# Ignore system files

.DS\_Store

Thumbs.db



Step 7 : Created 2 repo in docker hub as dev-react-img and prod-react-img



|  |  |
| --- | --- |
| Step 8 : | 1. Push all the code to github repo - <https://github.com/veera0794/project> 2. Created two branches in project repo as dev and master. 3. Created a Jenkinsfile in every branch |

**Jenkinsfile**

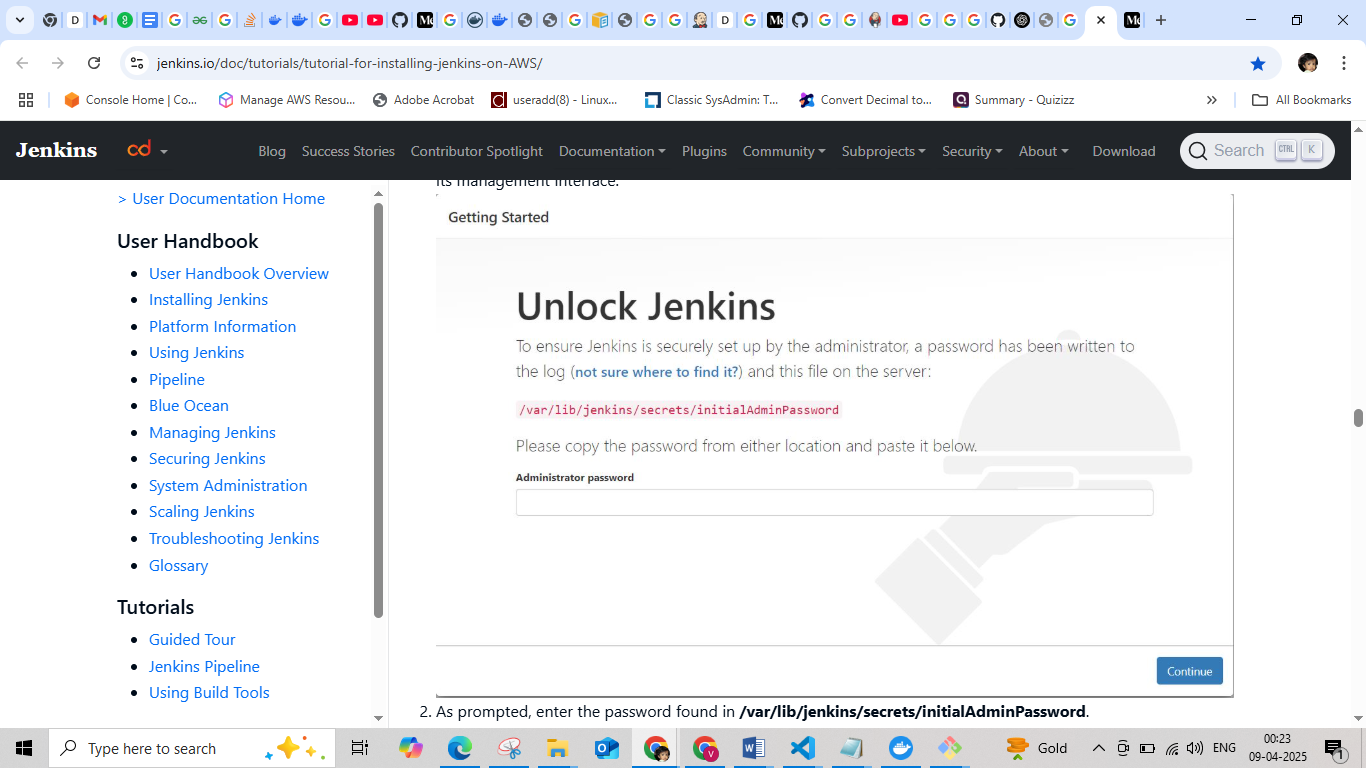
|  |
| --- |
| pipeline{  agent any  environment {  BRANCH\_NAME = "${env.BRANCH\_NAME}"  DOCKER\_CREDENTIALS\_ID = 'docker-hub-credential'  DEPLOY\_SERVER = '3.82.107.138' // Deployment server  DEPLOY\_PATH = '/var/www/app' // Deployment path  }  stages{  stage('Checkout'){  steps{  checkout scm  }  }  stage('Execution permission to scripts'){  steps{  sh '''  chmod +x build.sh  chmod +x deploy.sh  '''  }  }  stage('Build and Push Docker Image'){  steps{  withCredentials([usernamePassword(credentialsId: DOCKER\_CREDENTIALS\_ID, usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {  sh '''  export DOCKER\_USER=$DOCKER\_USER  export DOCKER\_PASS=$DOCKER\_PASS  ./build.sh  ./deploy.sh  '''  }  }  }  stage('Pull the pushed image and Deploy to EC2') {  steps{    withCredentials([usernamePassword(credentialsId: DOCKER\_CREDENTIALS\_ID, usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]){  sh """  echo "Deploying with $USER"  scp -o StrictHostKeyChecking=no -i /var/lib/jenkins/.ssh/grafana-key.pem deploy.sh ec2-user@3.82.107.138:/home/ec2-user/  #ssh -o StrictHostKeyChecking=no -i /var/lib/jenkins/.ssh/grafana-key.pem ec2-user@3.82.107.138 'bash /home/ec2-user/deploy.sh'  """  }  }  }  }  } |

Step 9: Configure Jenkins setup

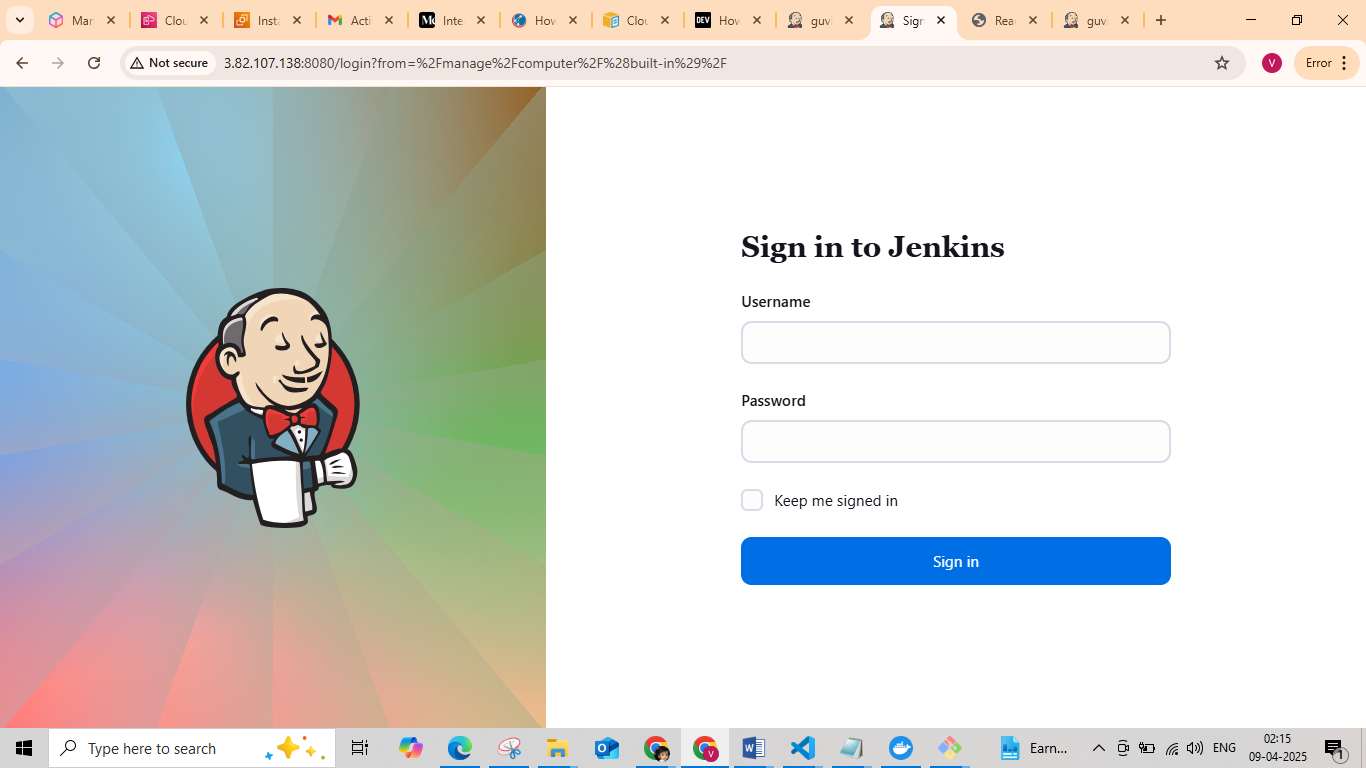
1. Create an EC2 to instance connect to it and run the following commands to install the Jenkin to instance

|  |
| --- |
| sudo yum update –y  sudo wget -O /etc/yum.repos.d/jenkins.repo \  <https://pkg.jenkins.io/redhat-stable/jenkins.repo>  sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key  sudo yum upgrade  sudo yum install java-17-amazon-corretto -y  sudo yum install jenkins -y  sudo systemctl enable jenkins  sudo systemctl start jenkins  sudo systemctl status jenkins |

1. Login to the Jenkin browser by <http://ec2_public_ip:8080>



1. Use this command to display the password **“sudo cat /var/lib/jenkins/secrets/initialAdminPassword “** for first time and create user name and password for login page.



1. Install the necessary plugin

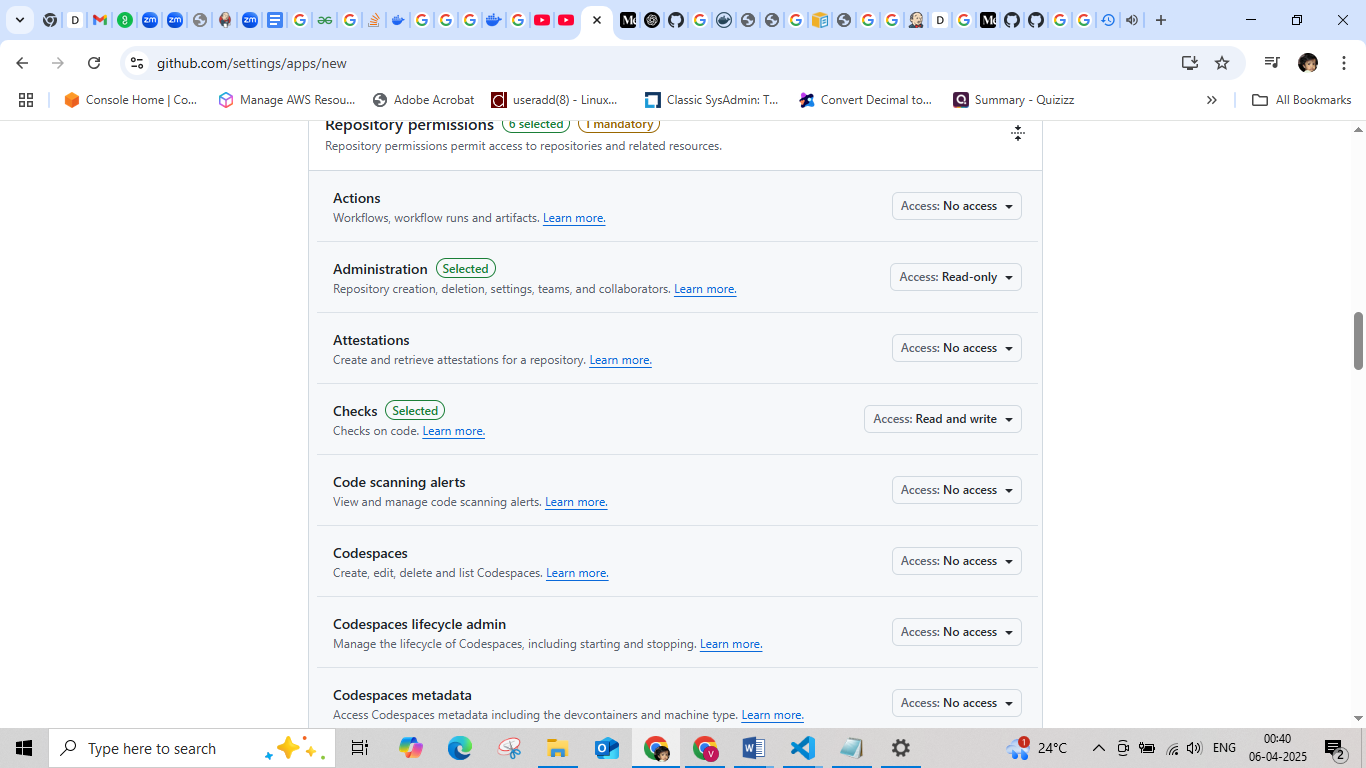
Jenkin setup is over next create a project.

Step 10 : Set-up the Git-hub App

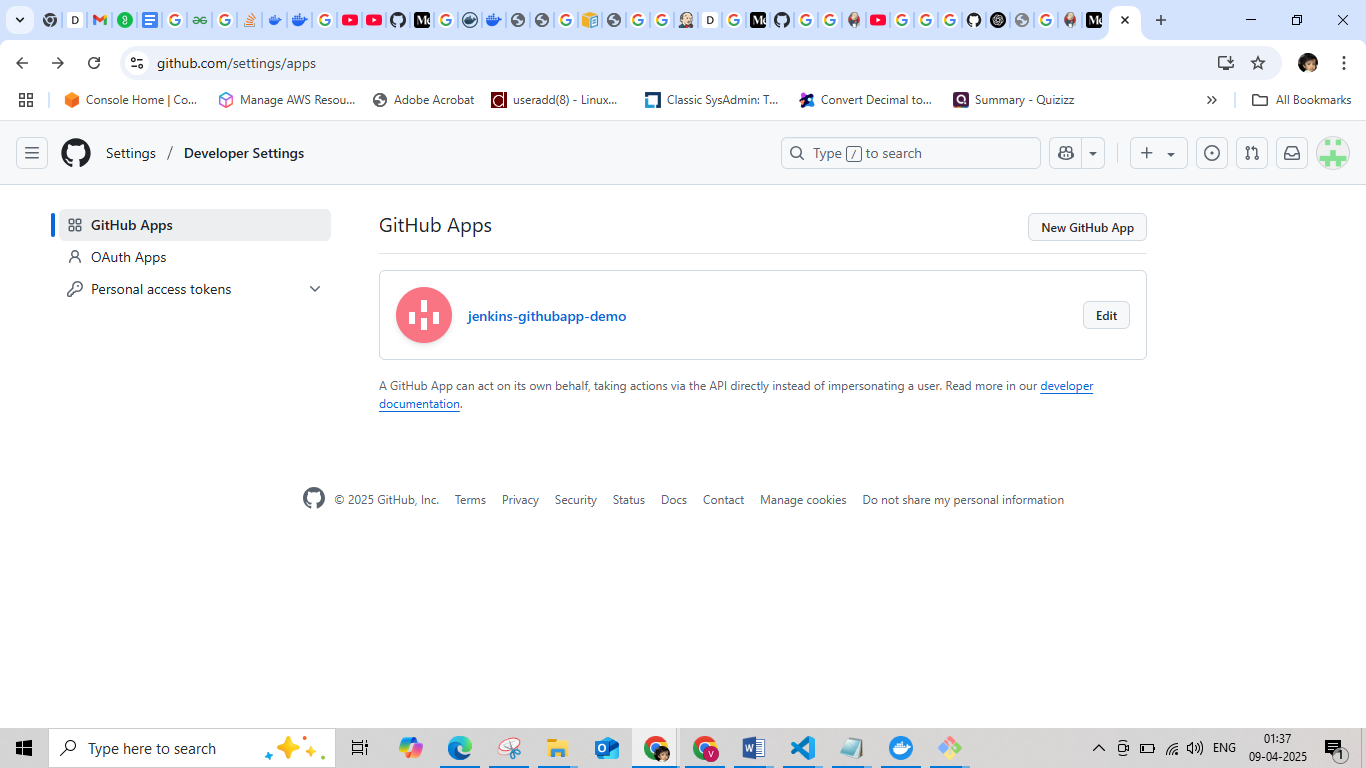
Settings 🡪 Developer Settings 🡪 Github Apps 🡪 New Github App

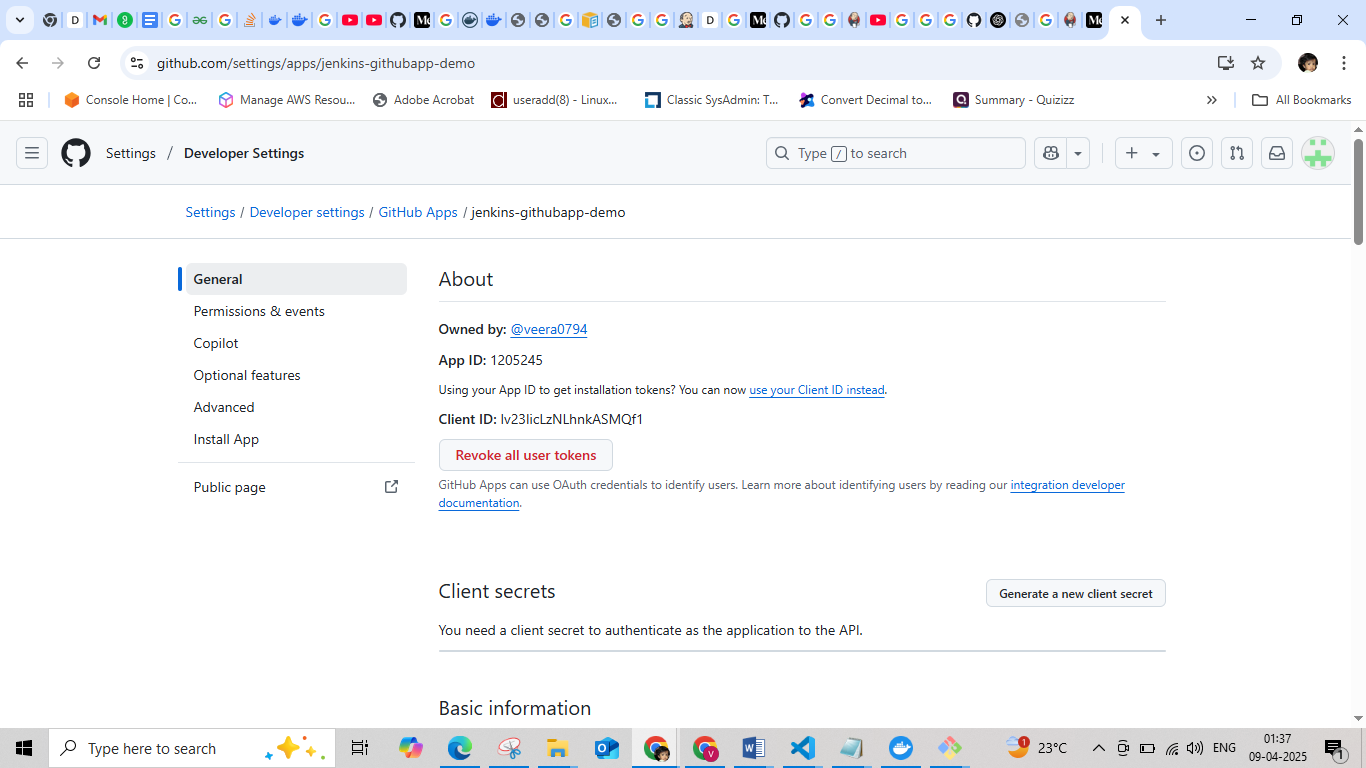
Give the following details:

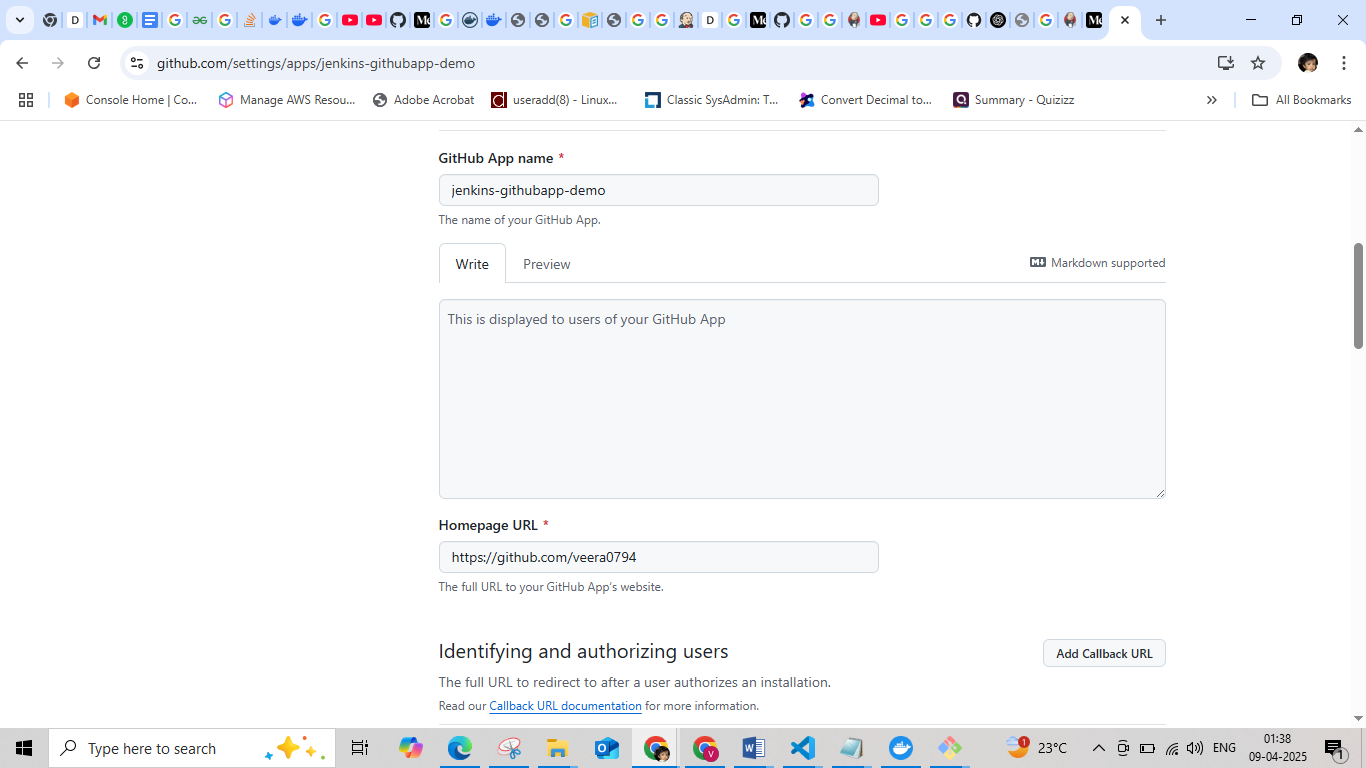
1. GitHub App name
2. Homepage URL
3. Webhook URL
4. Set the permissions

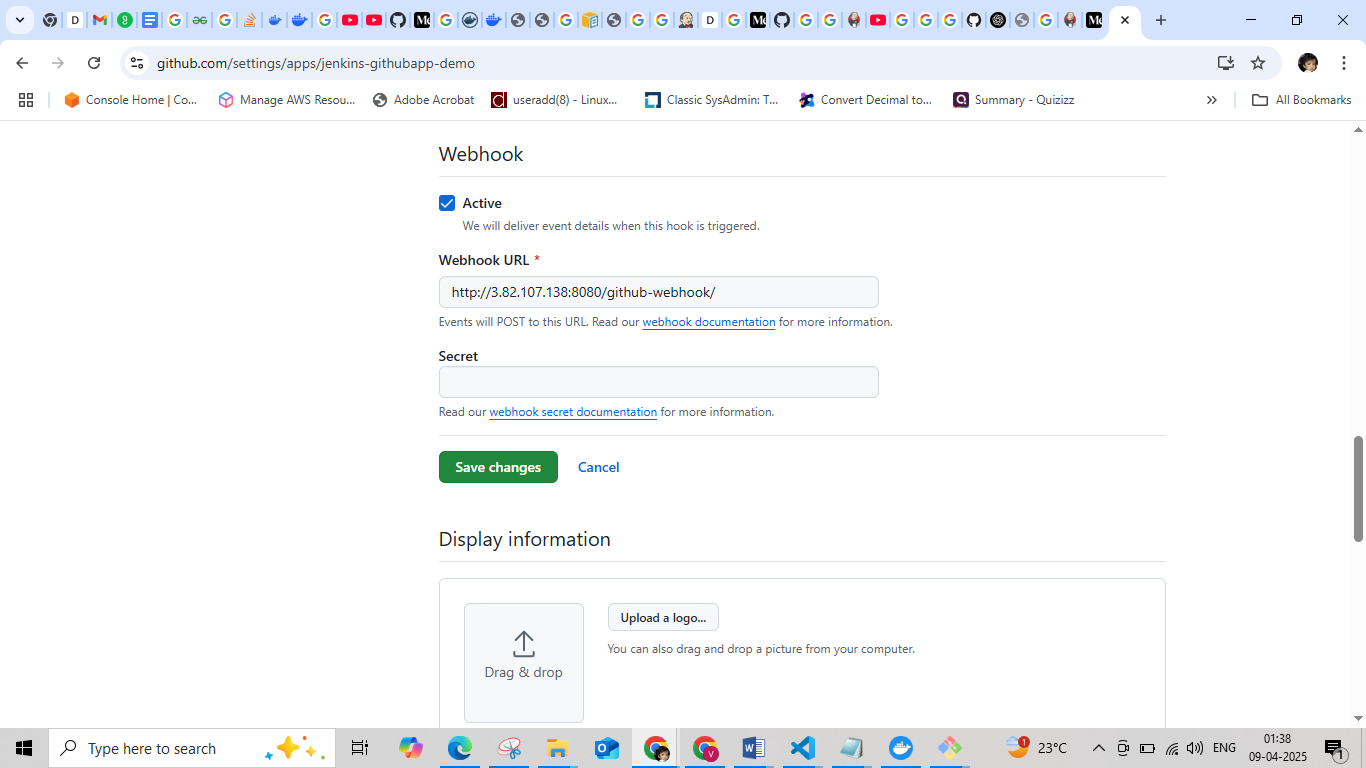


1. Create





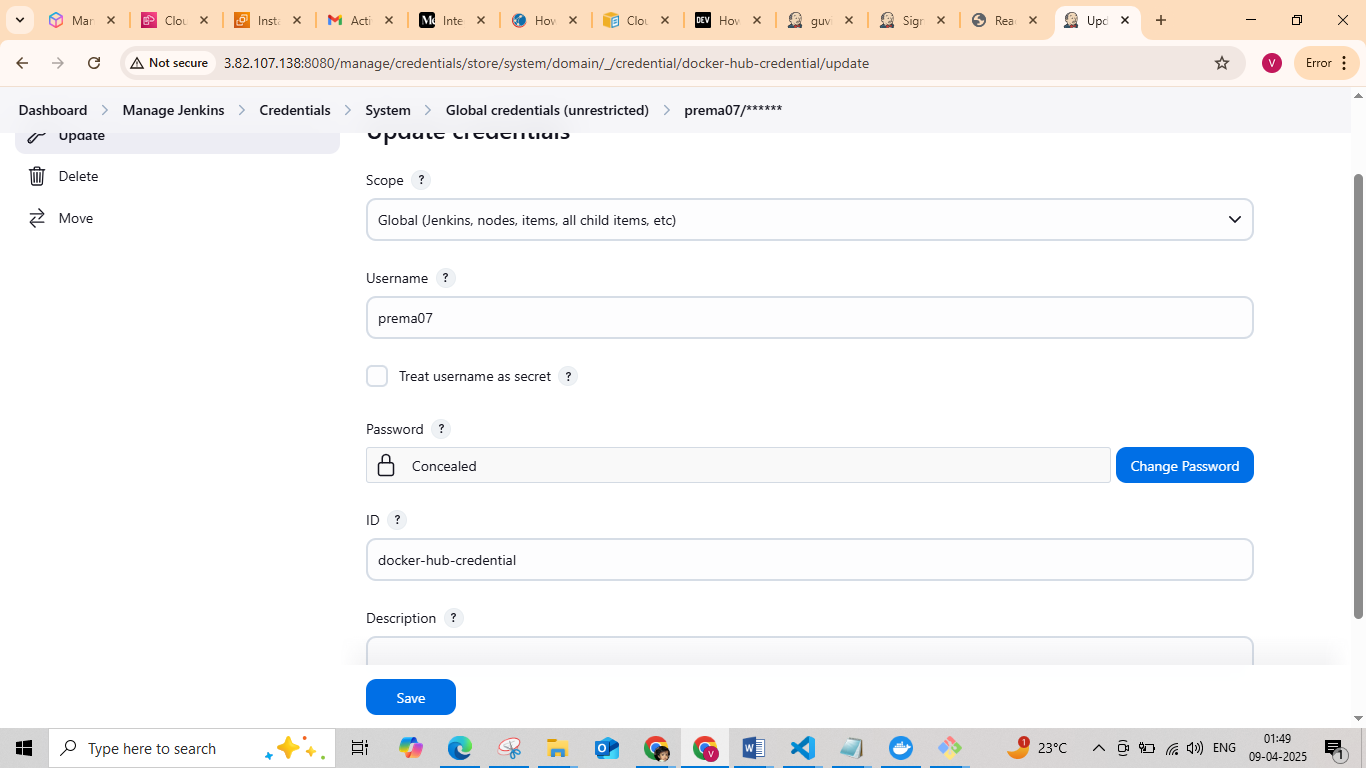




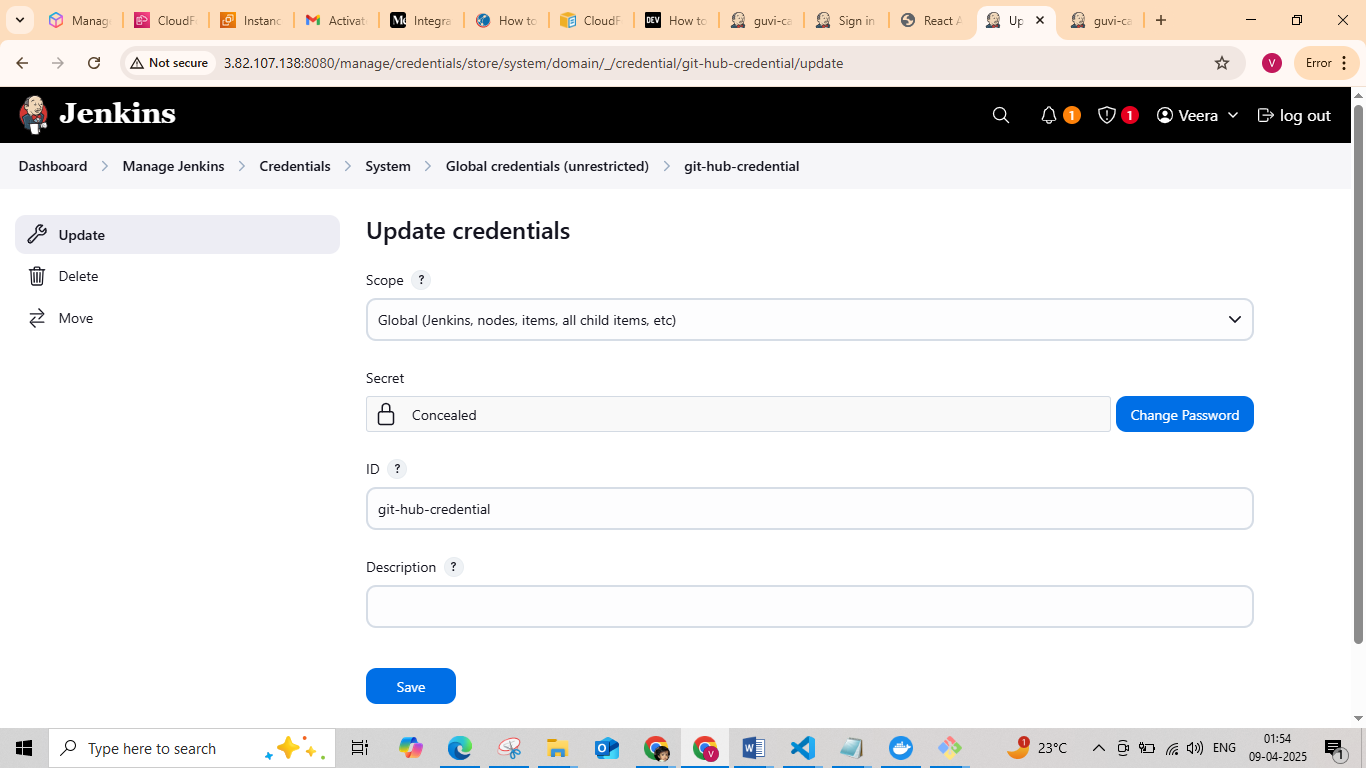
Step 11 : Set the credentials for Dockerhub and Github

Manage Jenkins 🡪 Credential

Docker hub credential (In password give the Docker hub token generate it from **Account Settings 🡪 Personal access token**)

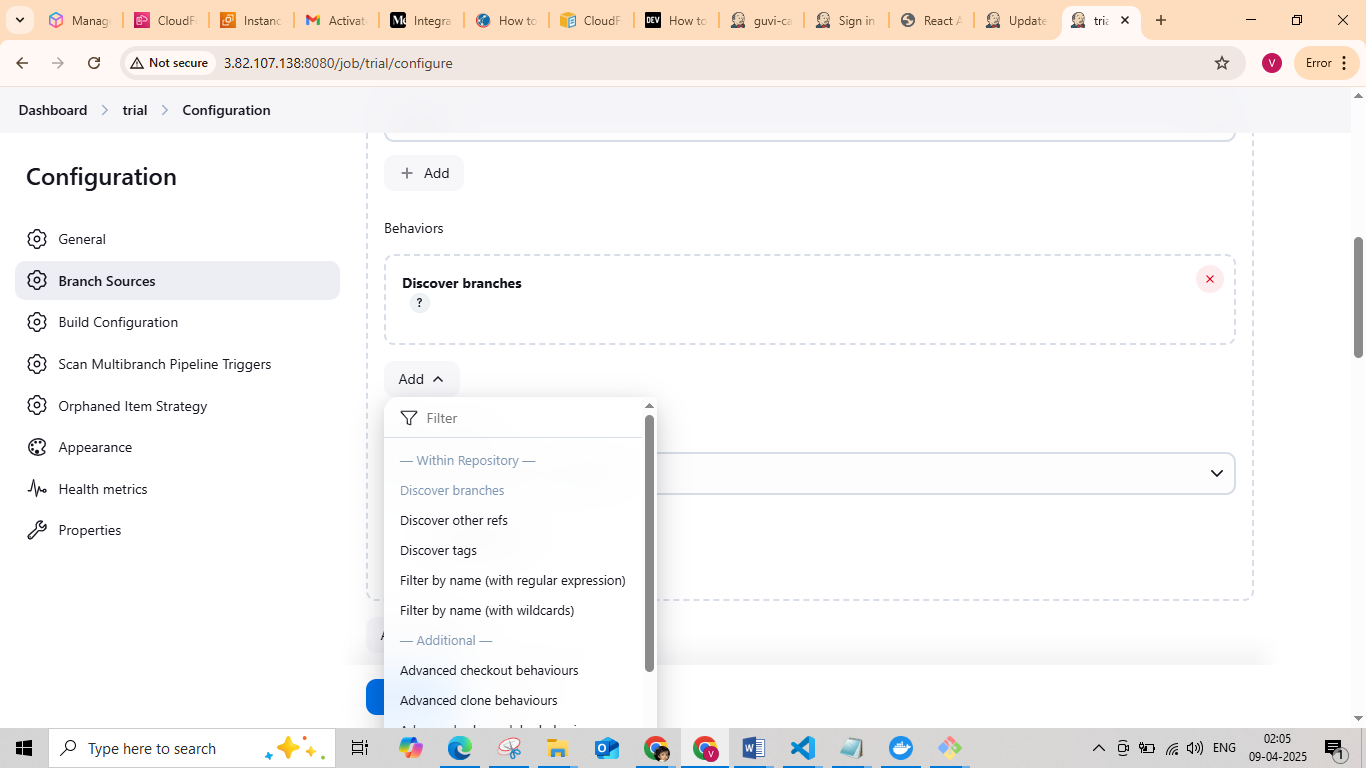


Git hub Credential (In password give the Github token generate it from **Setting** 🡪 **Developer Settings 🡪 General Access token**)

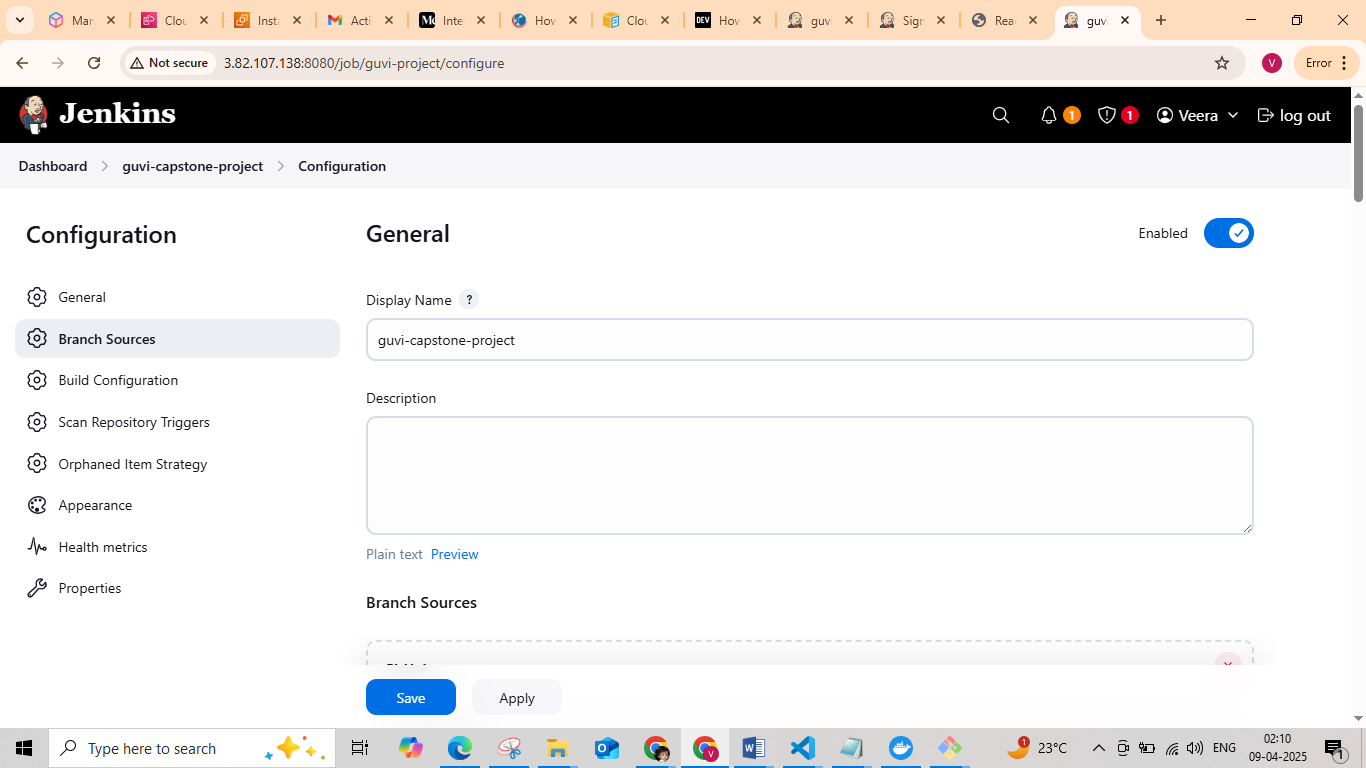


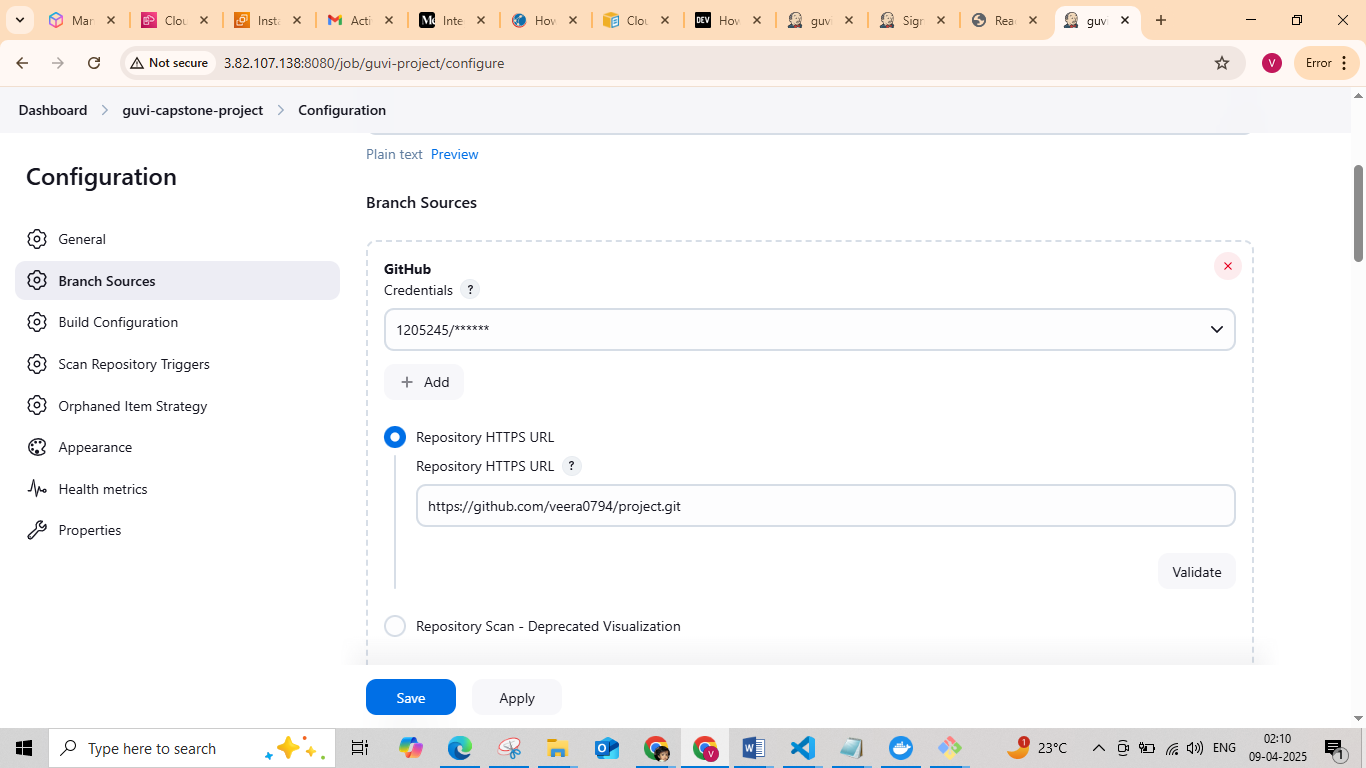
Step 12 : Create a Multibranch Project in Jenkin

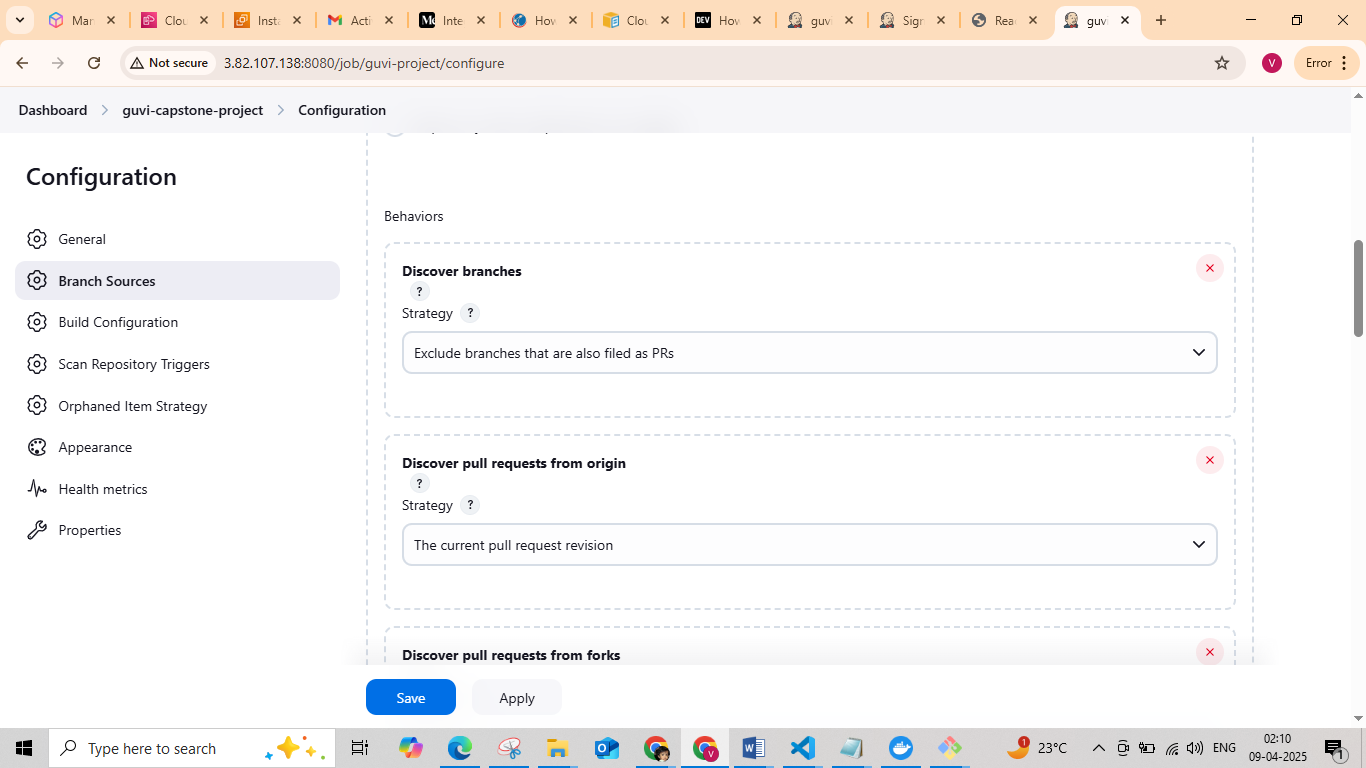
1. Dashboard 🡪 New items 🡪 Enter an item name 🡪 Multibranch Pipeline
2. Give the display name
3. In Branch Source select Github, set the URL and credential for github
4. In Discover Branch select Filter by name (Change according to configuration)

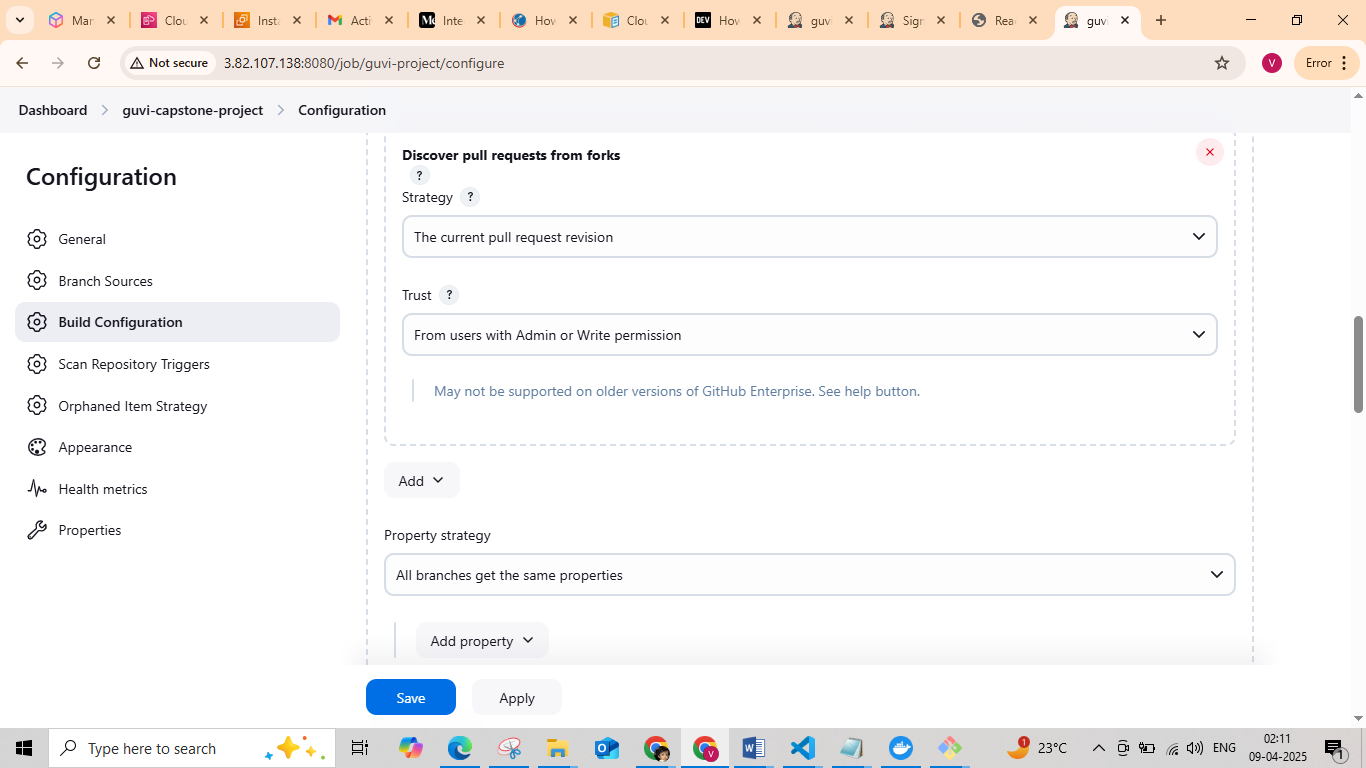


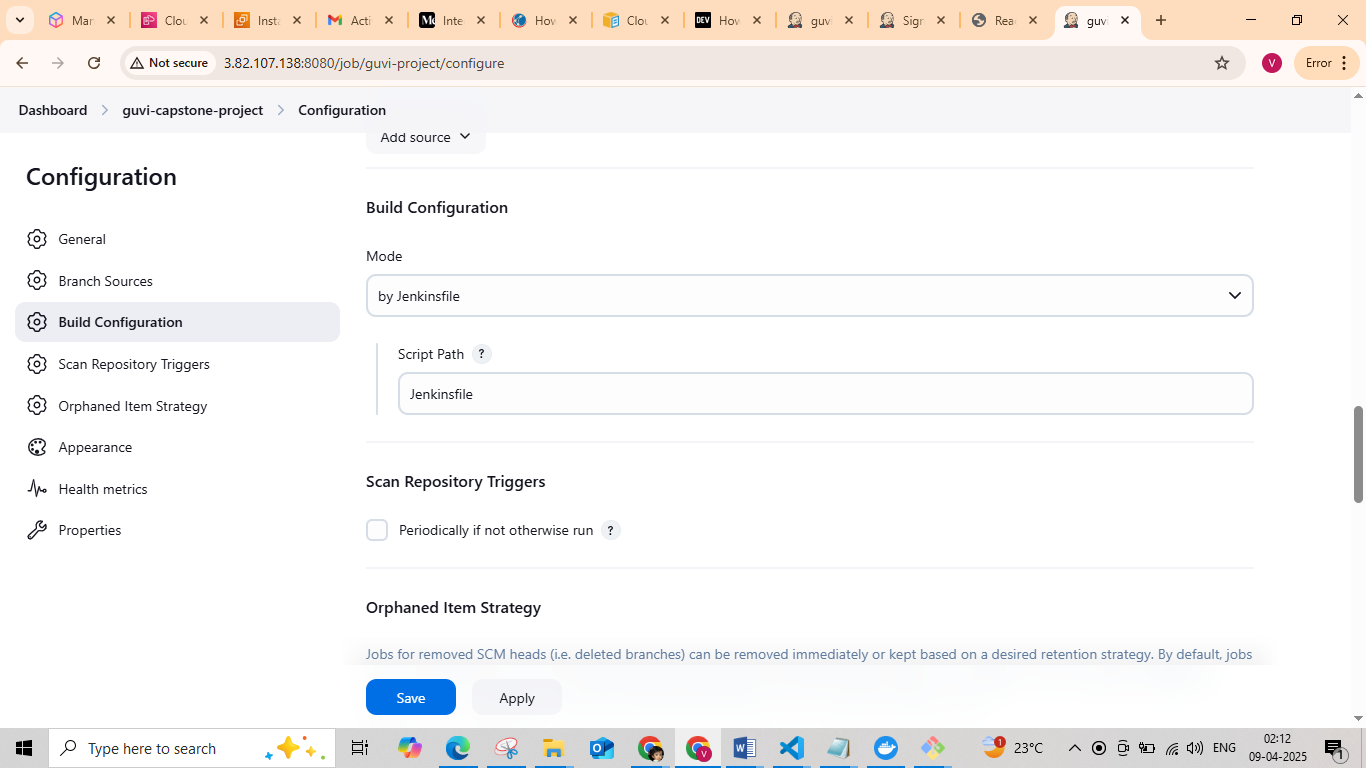
1. Save and Apply

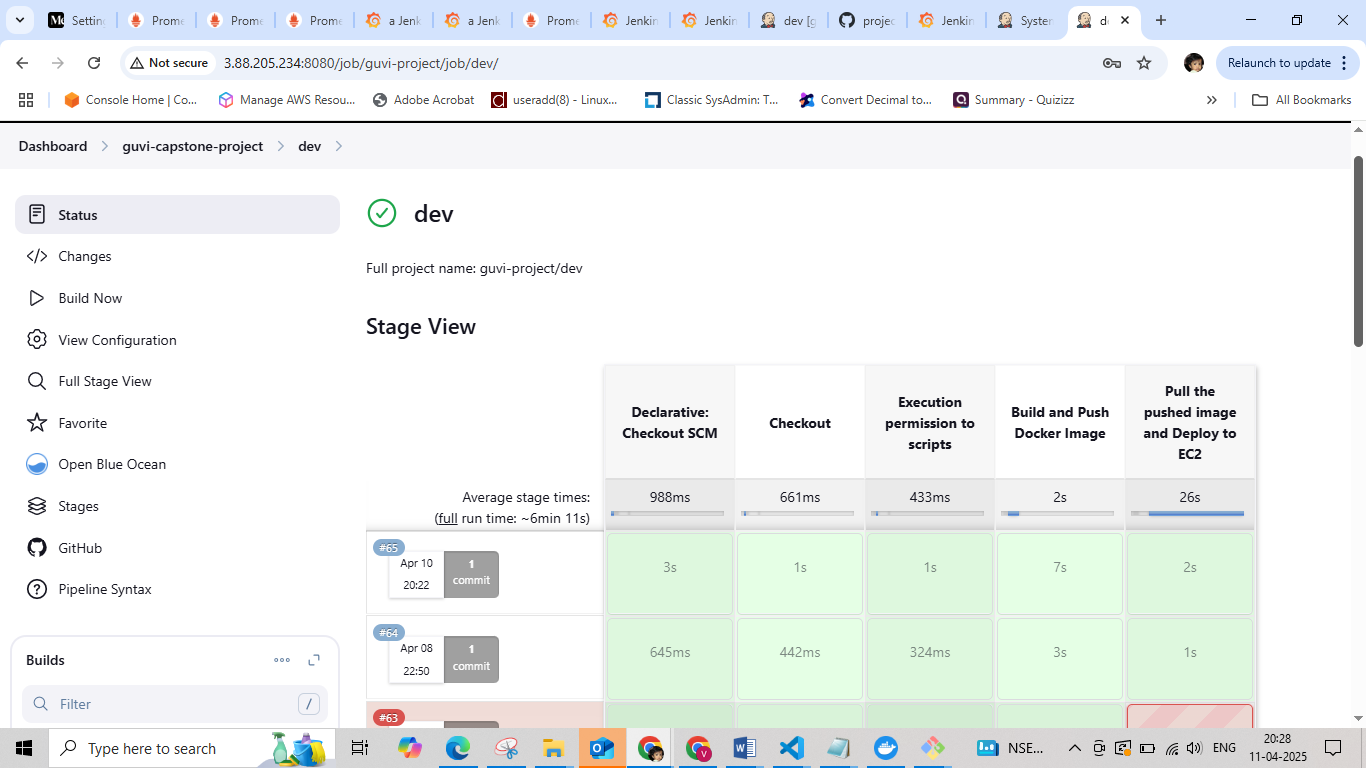


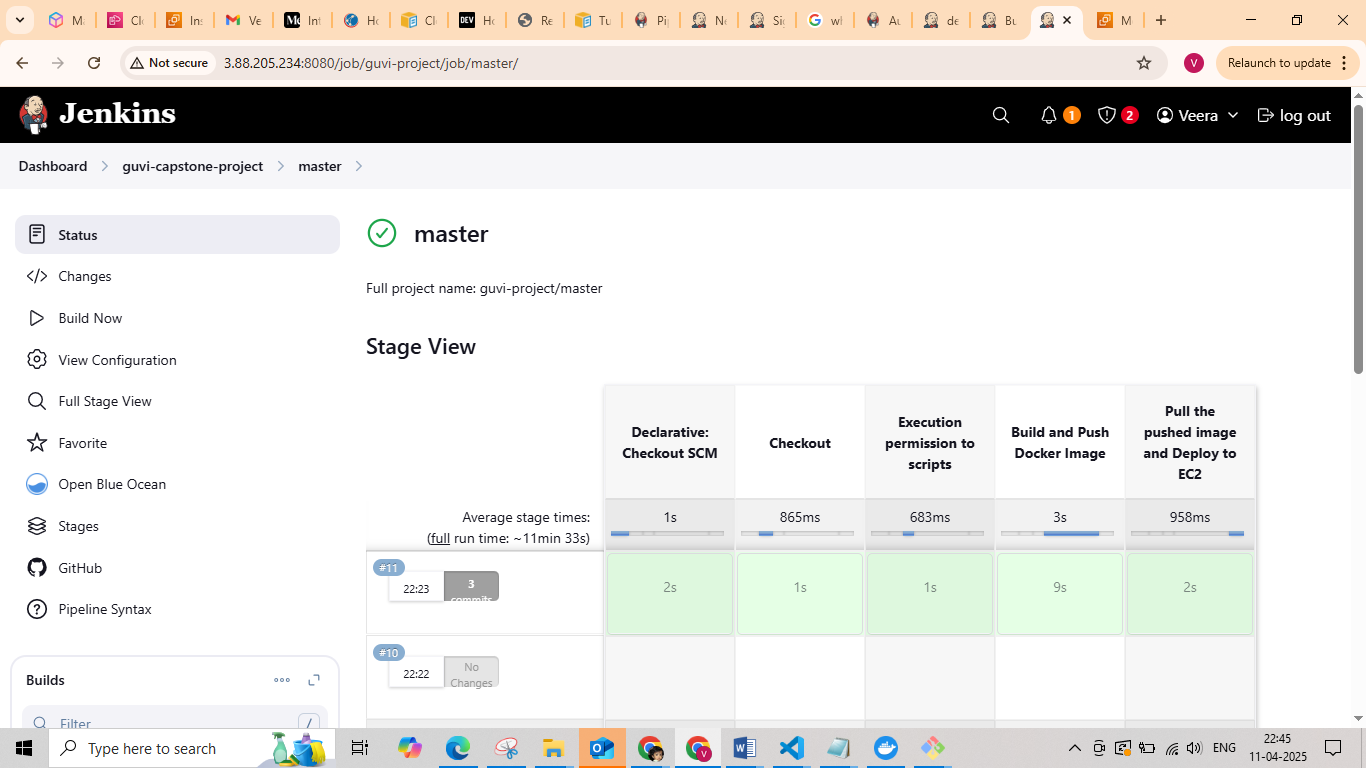




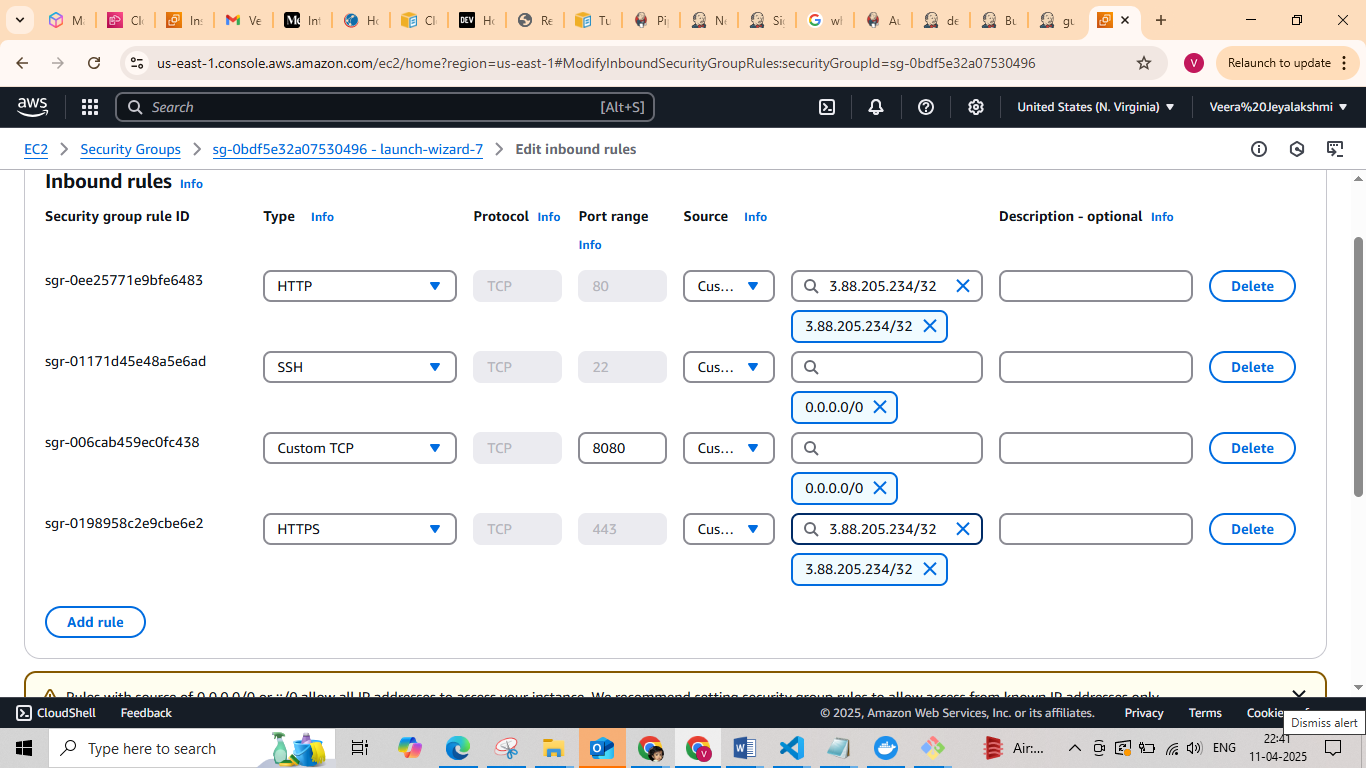




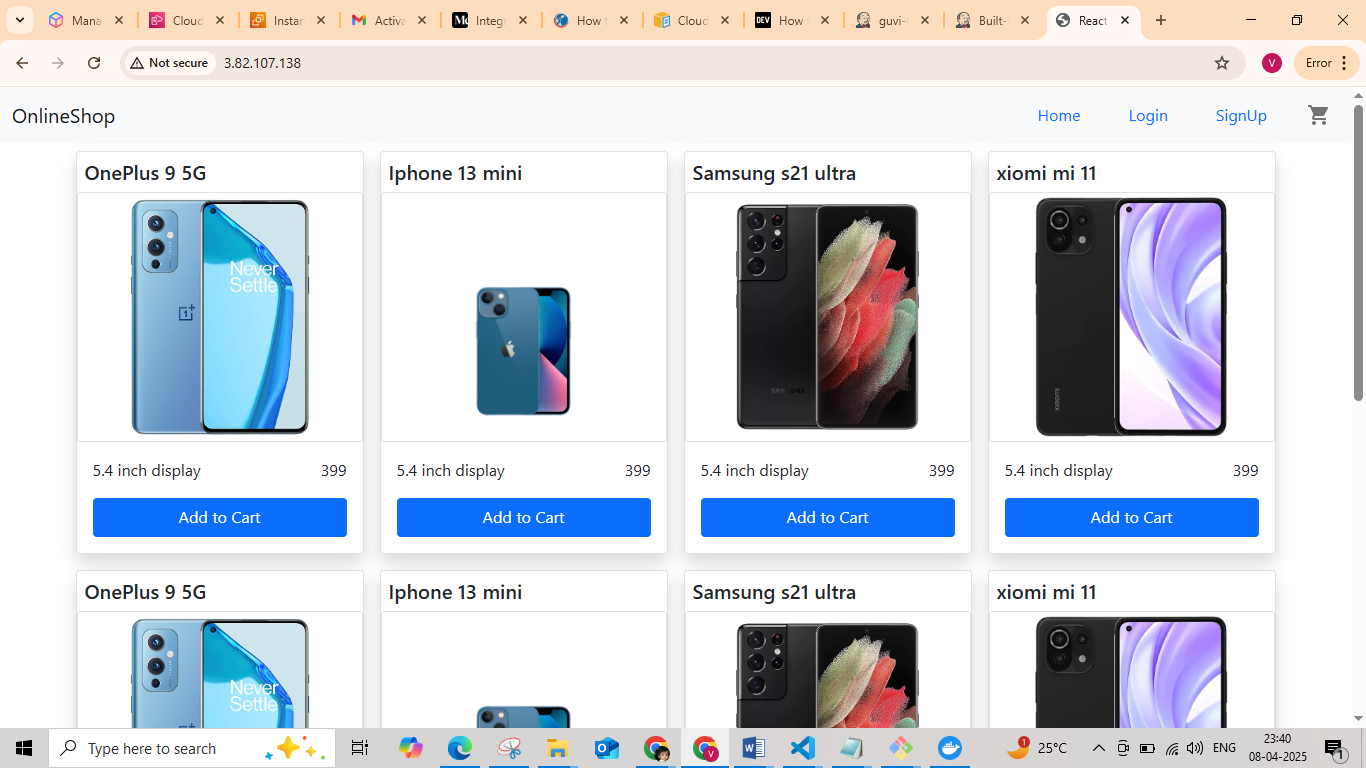




Security Group

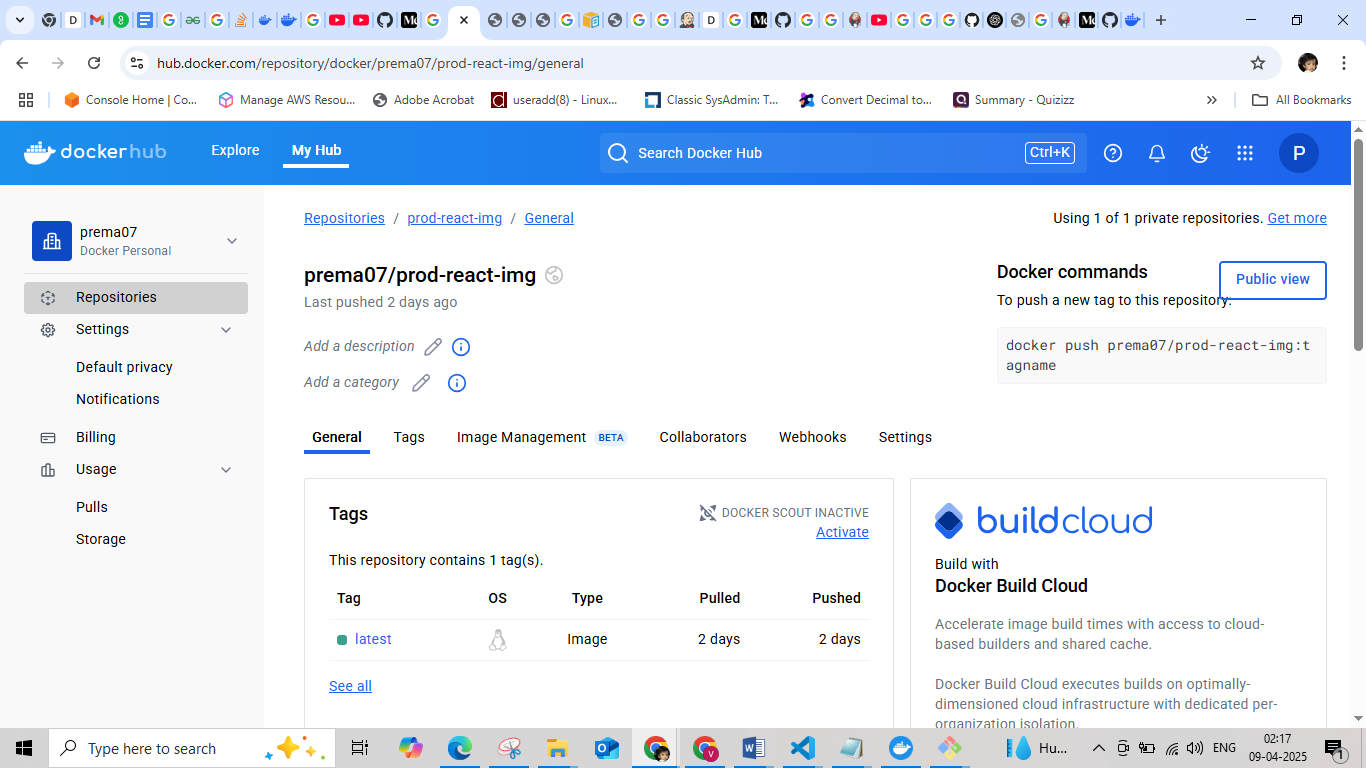


Open in the browser – http://<Ip-address of jenkin instance>



Docker-hub-repo

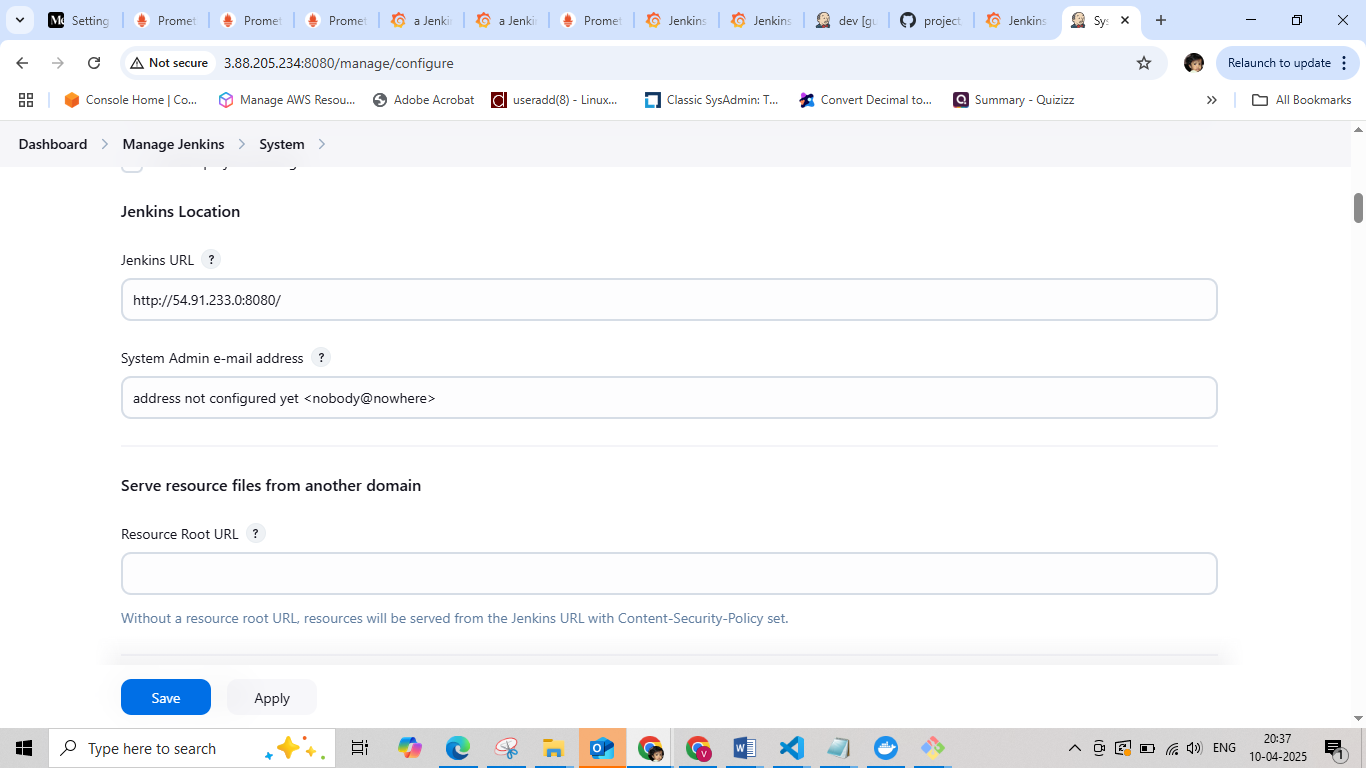


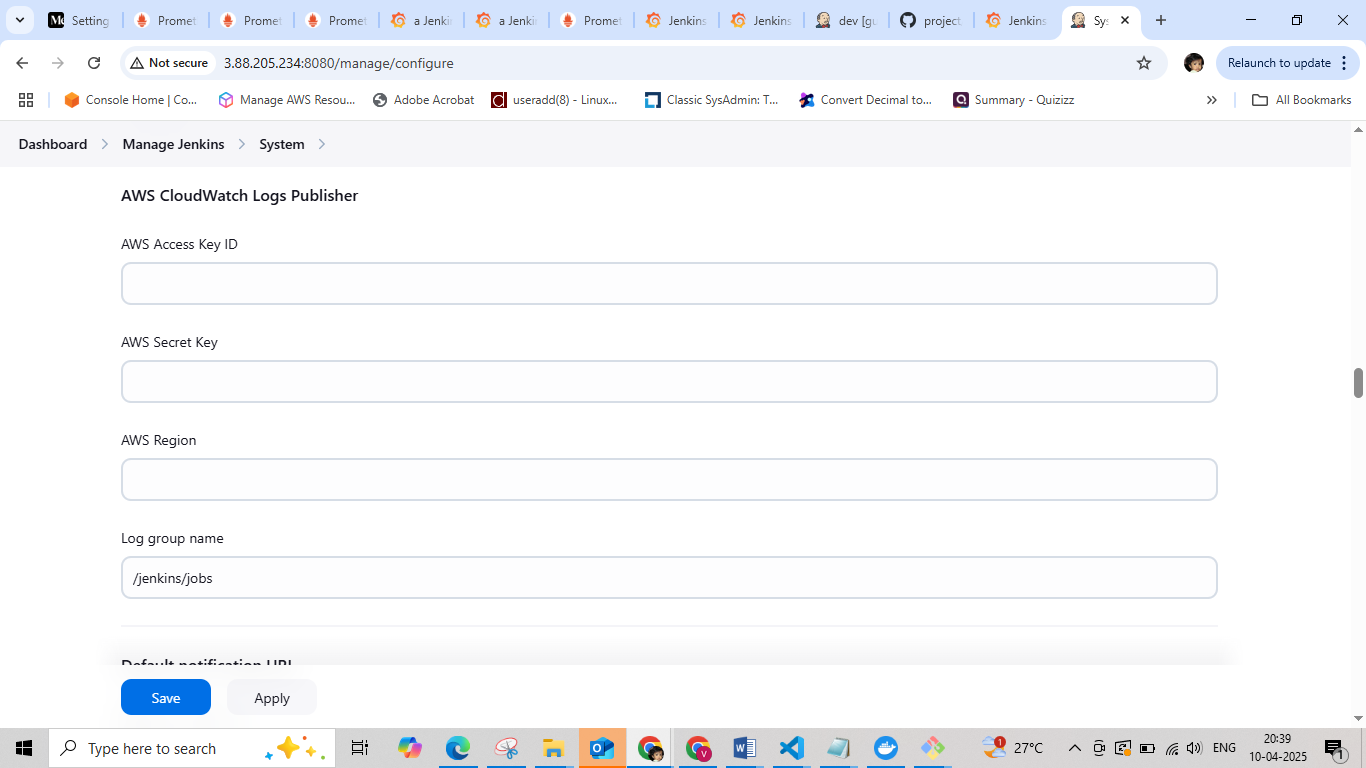


Configure the Jenkin Setup

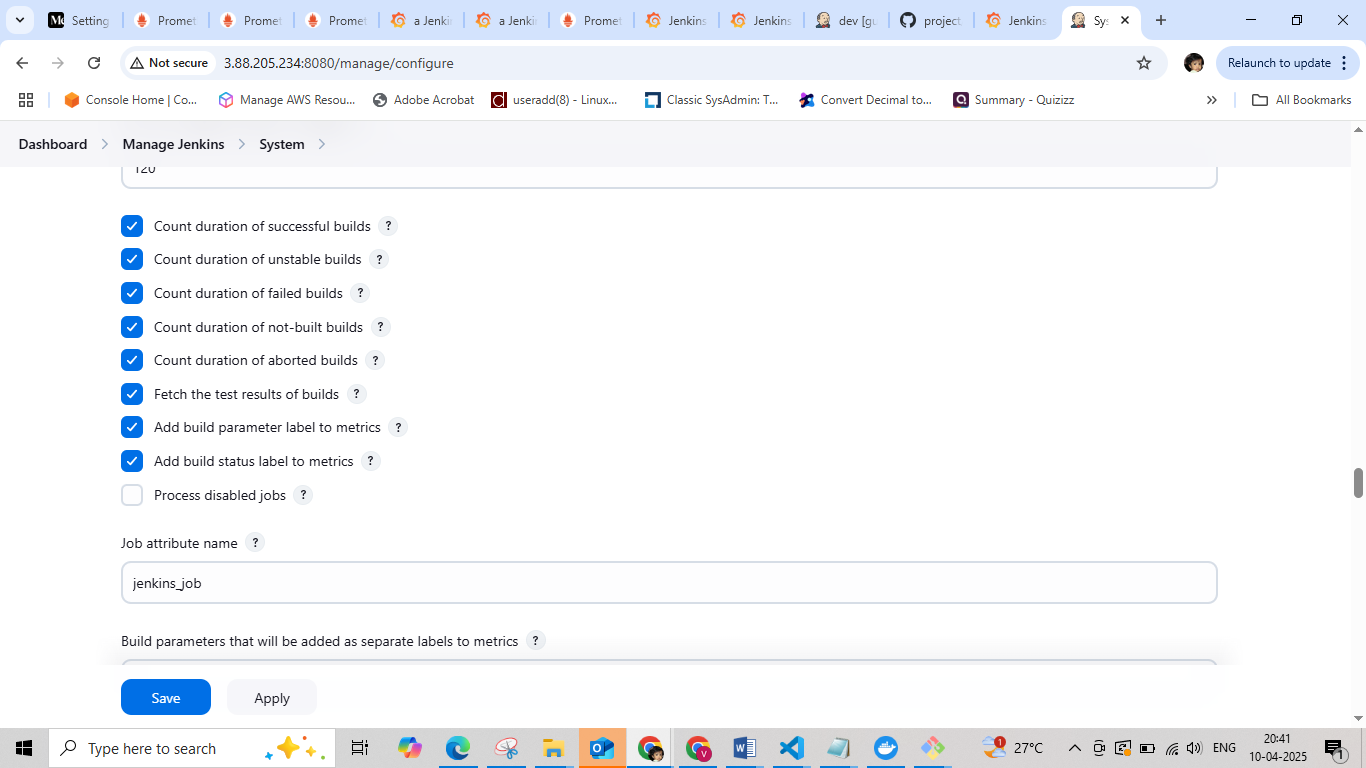
 **Go to Jenkins → Manage Jenkins → Configure Global Security.**

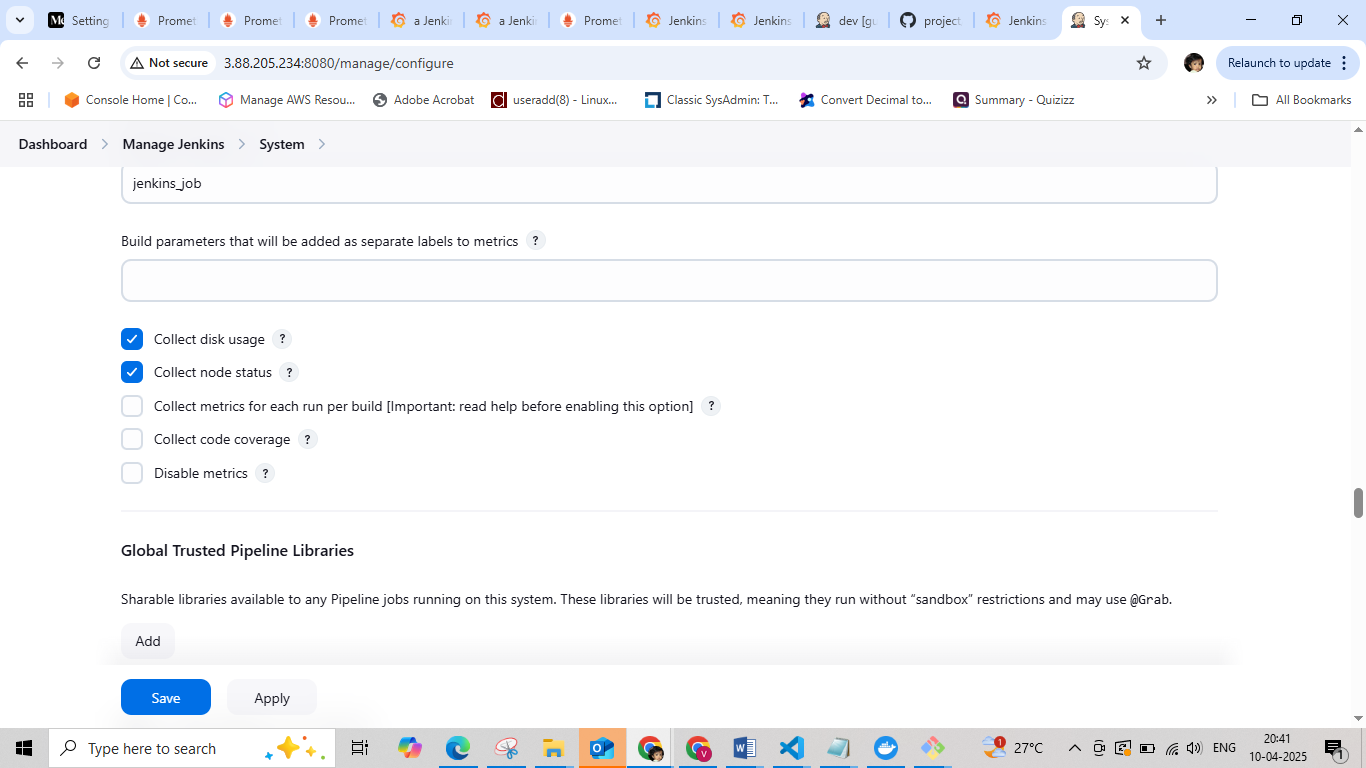
 Scroll down to the **Prometheus** section and enable the Prometheus end point.







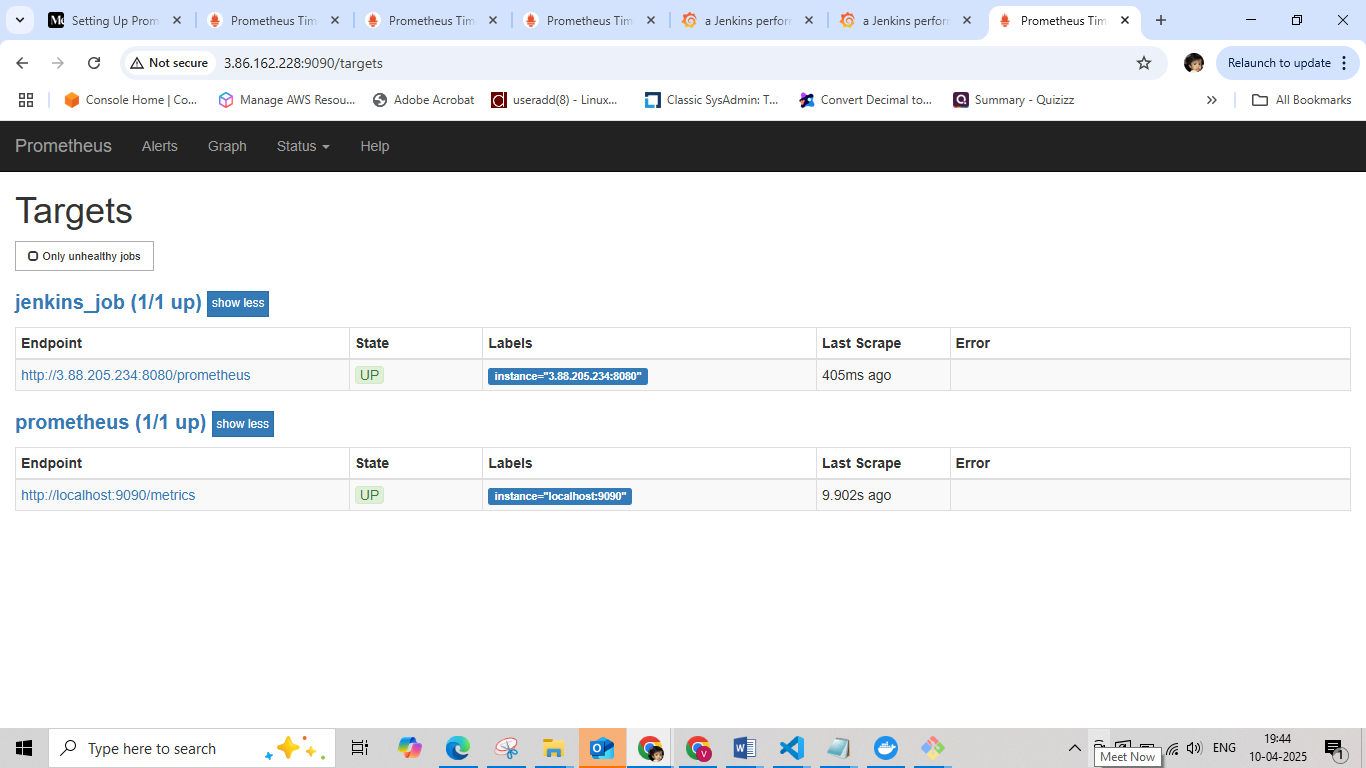




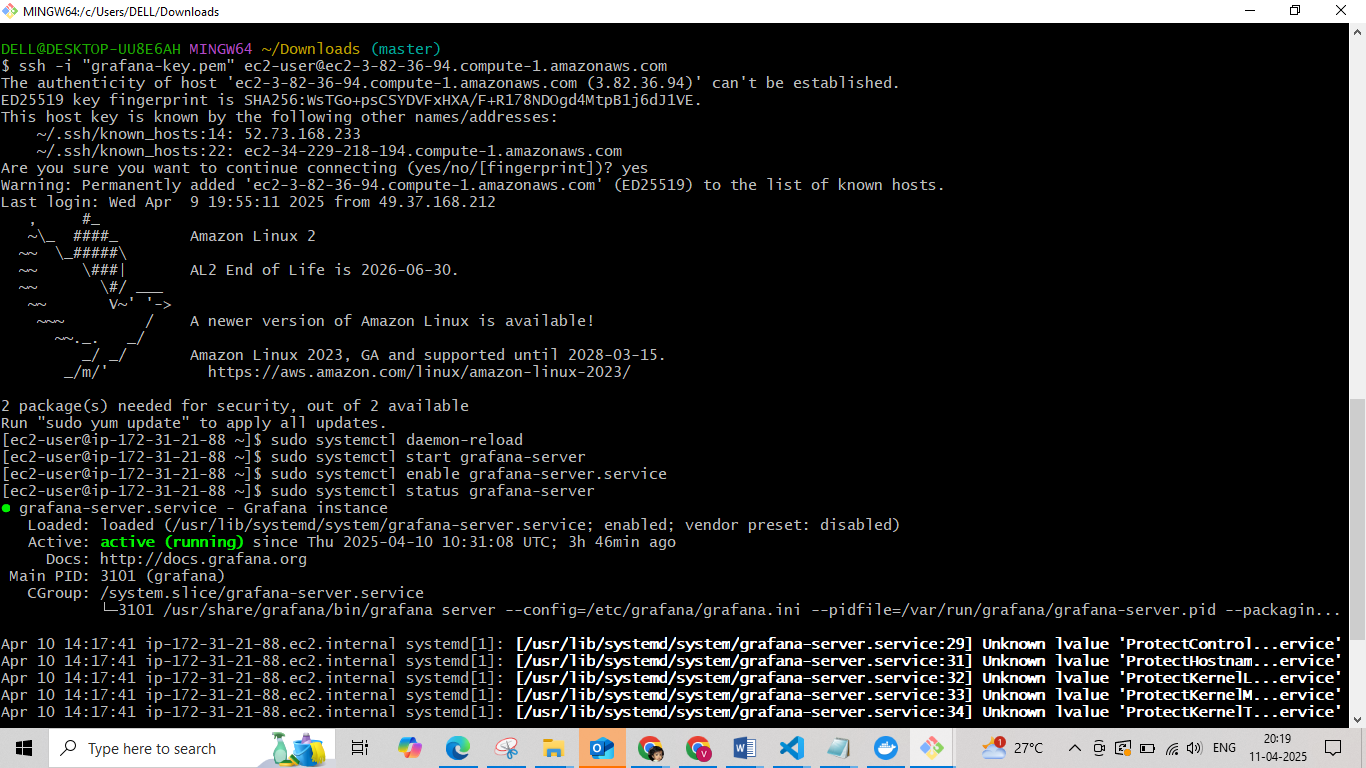
Update the Prometheus.yml file with Jenkins target and restart the Prometheus.

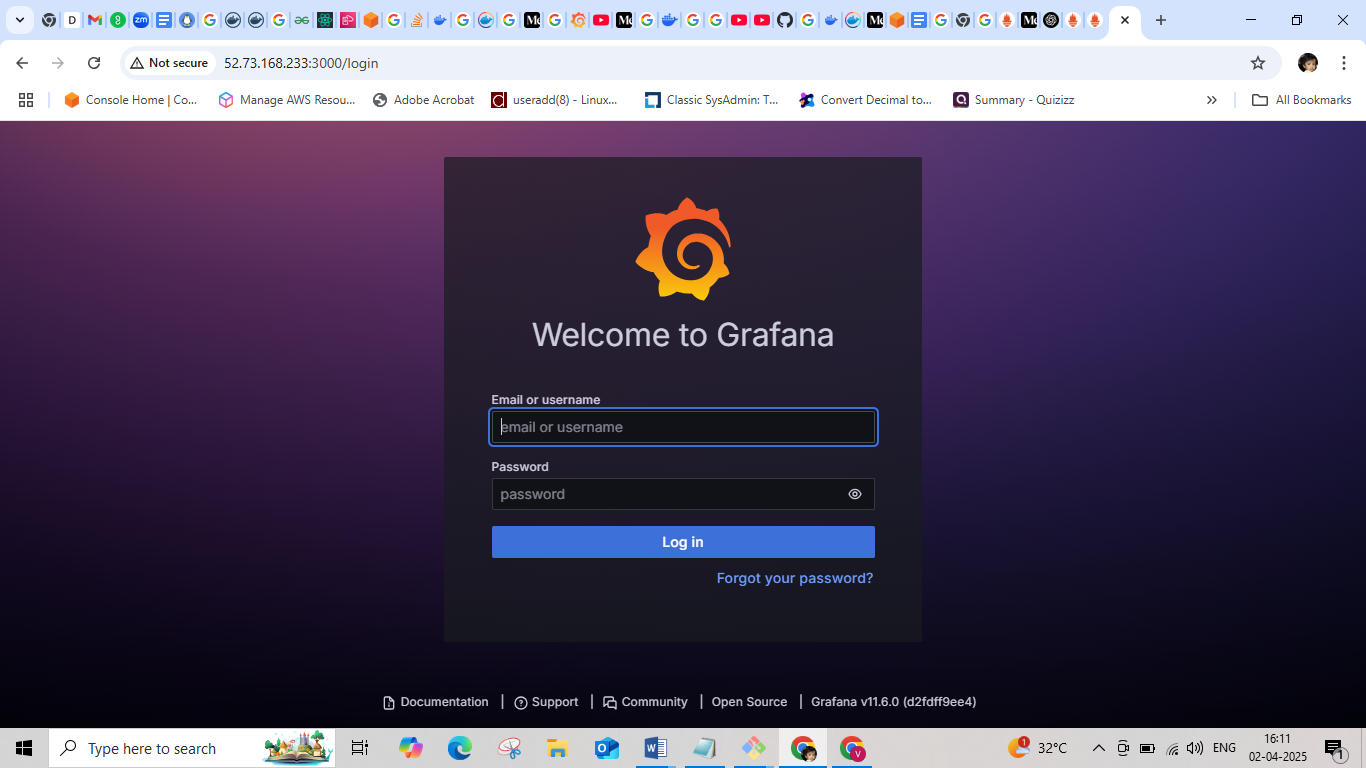
|  |
| --- |
| - job\_name: 'jenkins\_job'  metrics\_path: '/metrics' # Or '/prometheus' if that works  static\_configs:  - targets: ['<jenkins-ip>:8080'] # Replace with your Jenkins instance IP |
| [ec2-user@ip-172-31-90-118 prometheus]$ sudo vim prometheus.yml  [ec2-user@ip-172-31-90-118 prometheus]$ sudo systemctl daemon-reload  [ec2-user@ip-172-31-90-118 prometheus]$ sudo systemctl enable prometheus  [ec2-user@ip-172-31-90-118 prometheus]$ sudo systemctl restart prometheus  [ec2-user@ip-172-31-90-118 prometheus]$ sudo systemctl status prometheus |

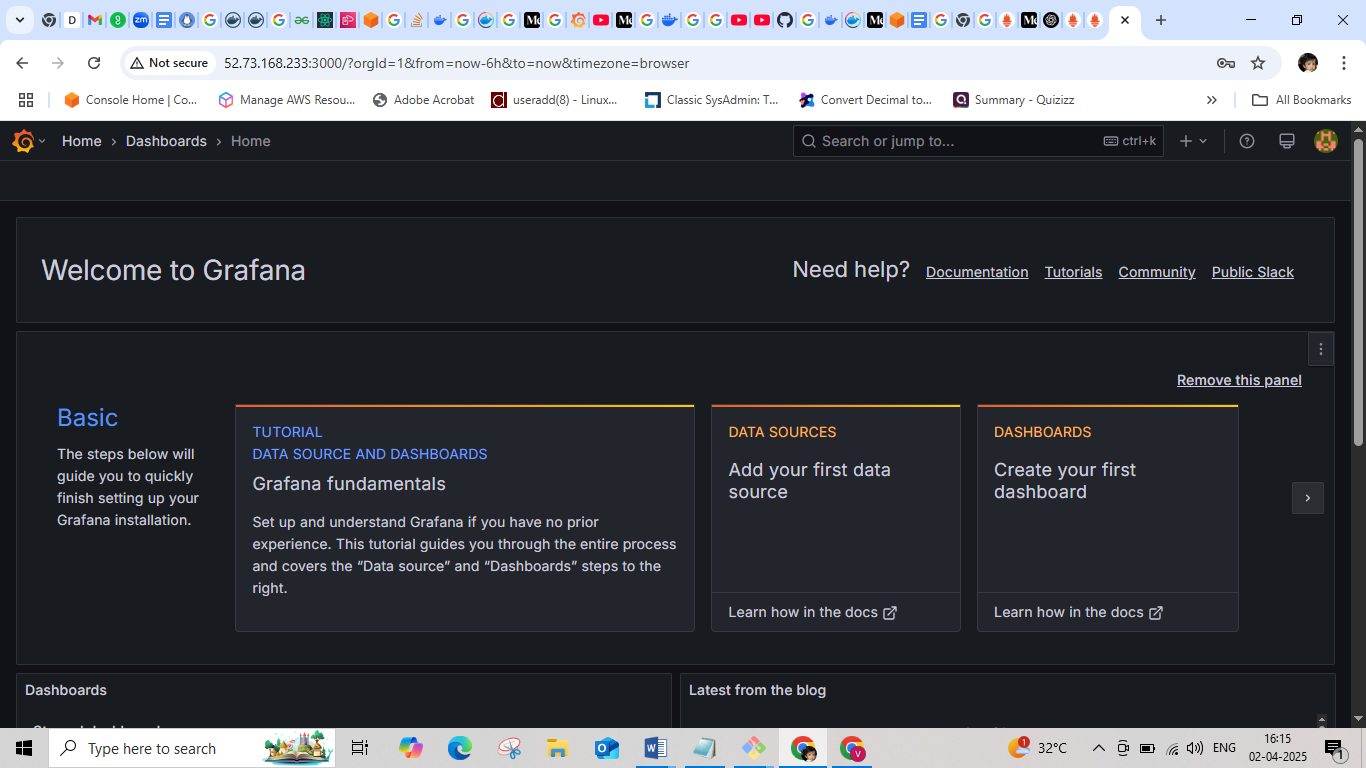
Open the browser http:// <IP-address of Prometheus server>:9090

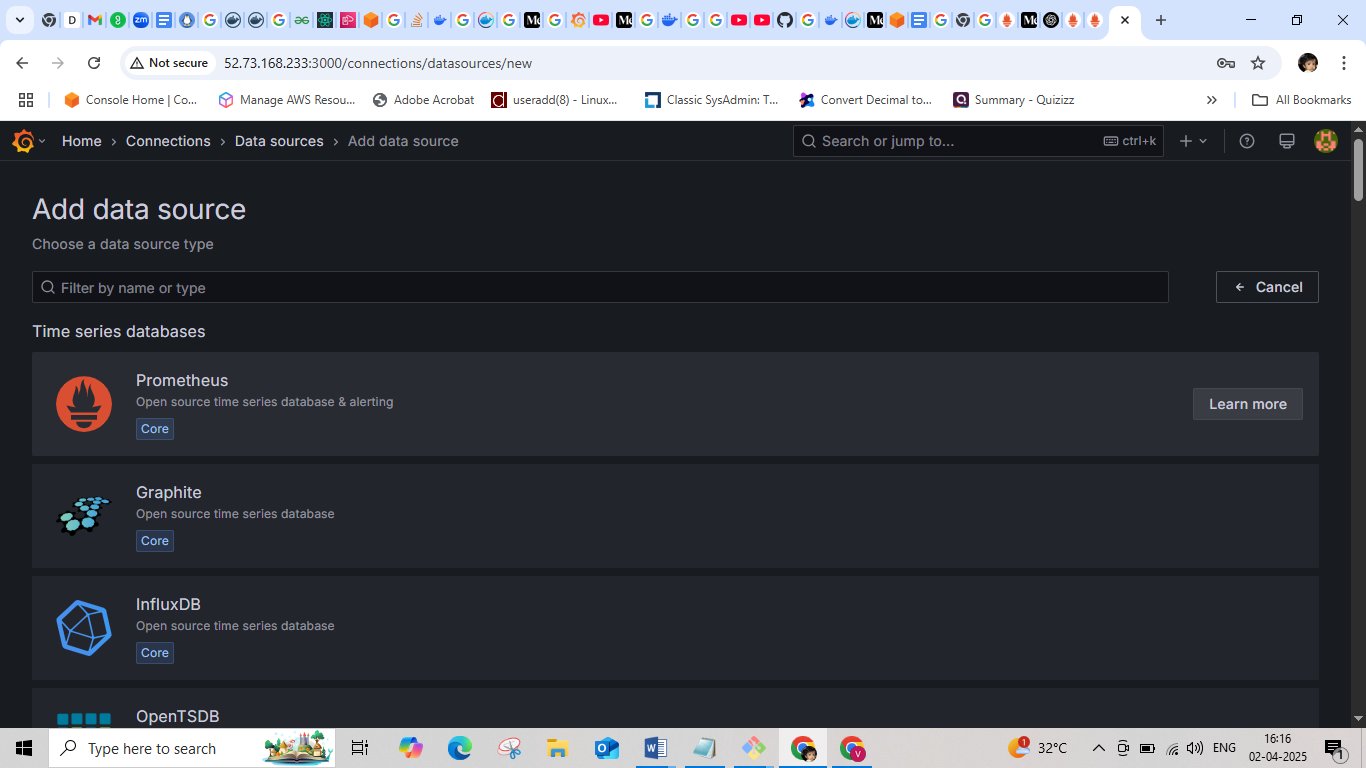


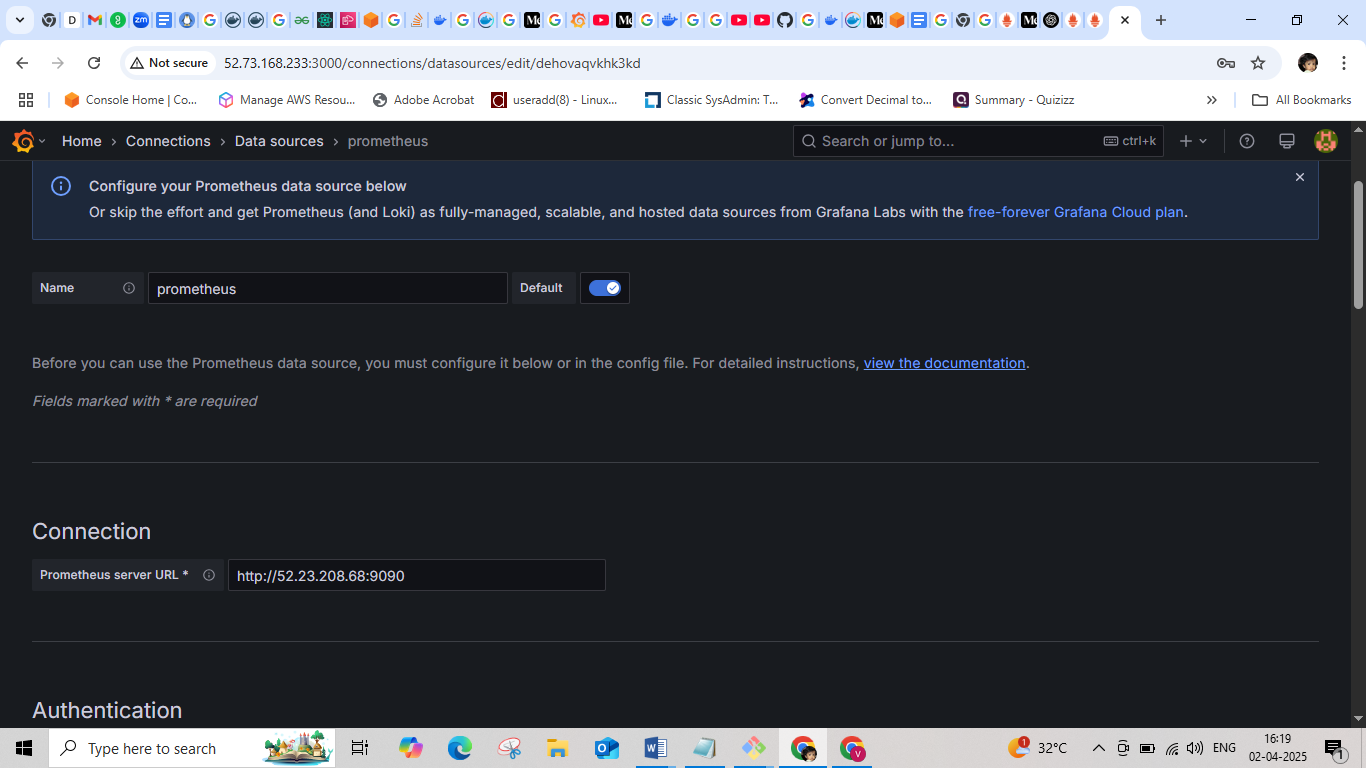
Start the Grafana











Add the Prometheus URL as http://Ip\_address of prometheus instance:9090 then save and test it.

Import the dashboard

