PROJECT-2

Automated CI/CD Pipeline Setup Using Jenkins, GitHub, Docker, And AWS EC2

Pre-Requisites:

AWS Account: An active AWS account with permissions to create and manage EC2 instances.

GitHub: A GitHub account to fork the sample repository and host your code.

Maven: To build the Java project.

Docker: A Docker Hub account to build, push, and store your images.

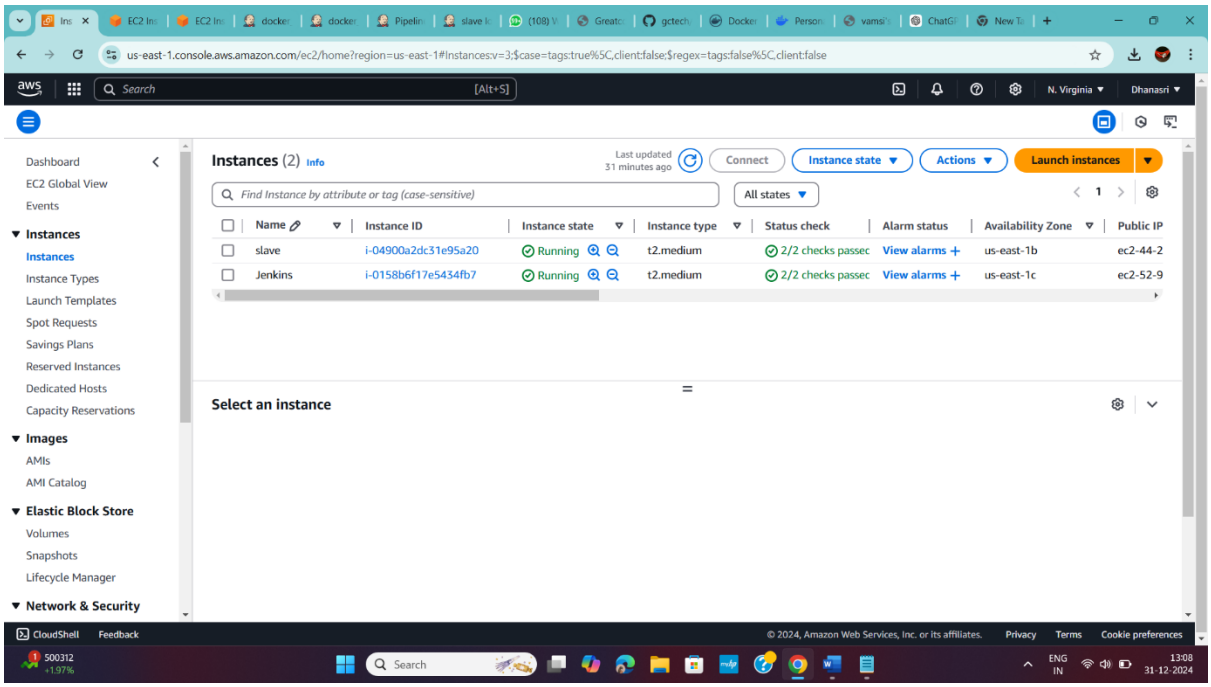
Jenkins: To create CI/CD Pipeline and to add build triggers

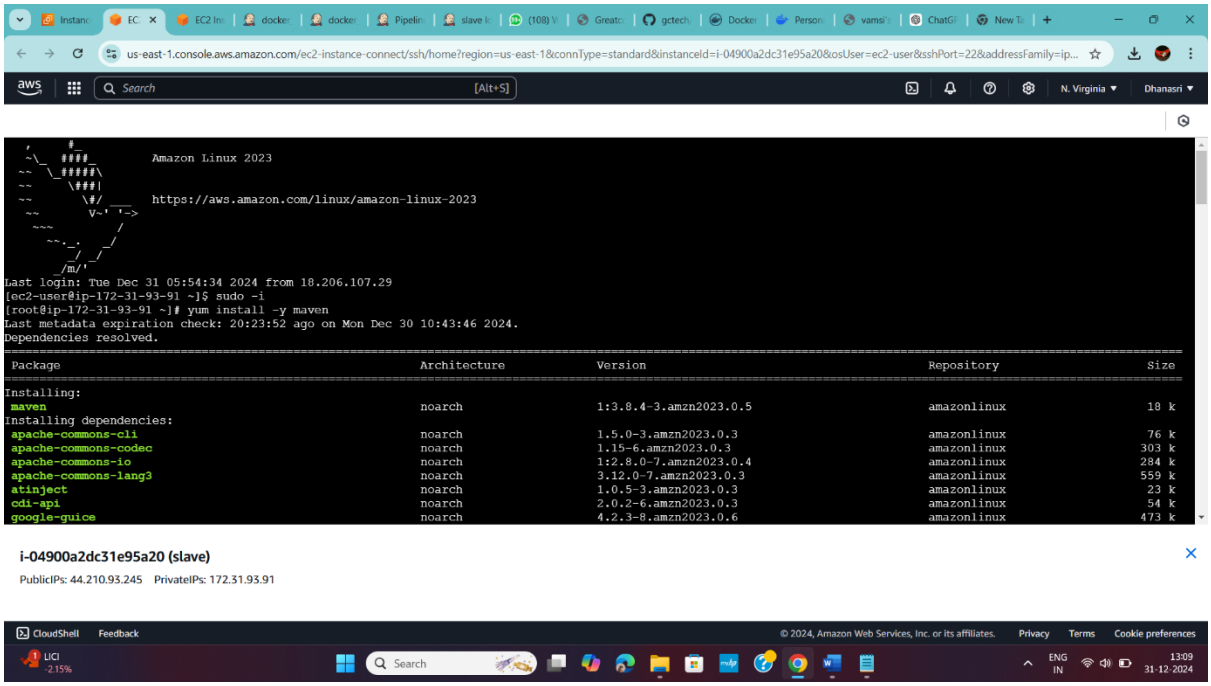
STEP 1:Install , Configure Jenkins on the Master Instance & Setup Jenkins Slave Instance

1.Launch slave and jenkins instances(master) with t2.medium configuration.

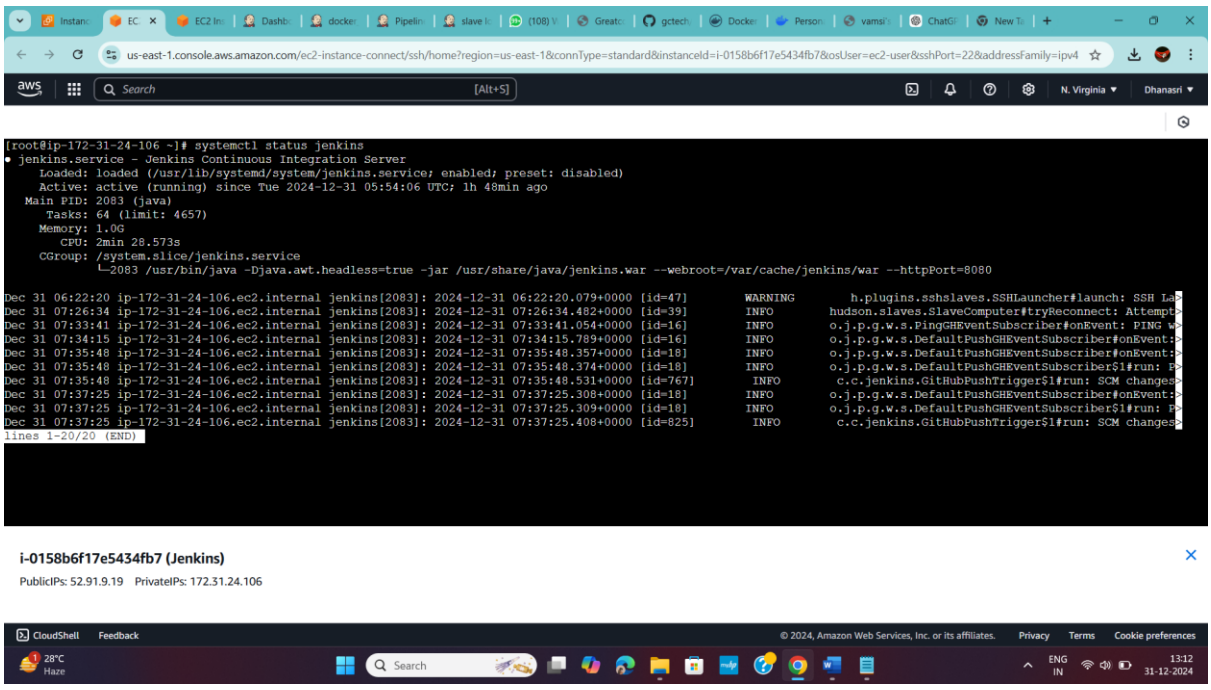
2.Setup jenkins and slave instance by connecting.

3.Install Java,maven,docker and git in node instance.

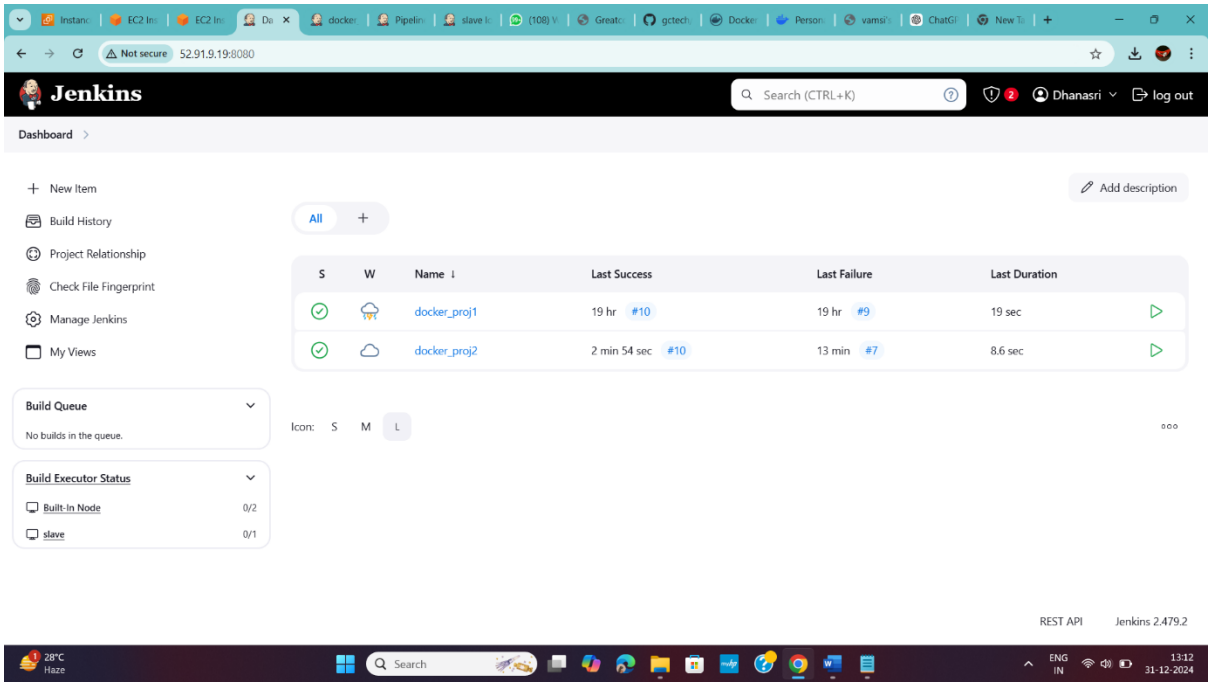




4.Start jenkins in jenkins(master) instance by doing all the configurations.

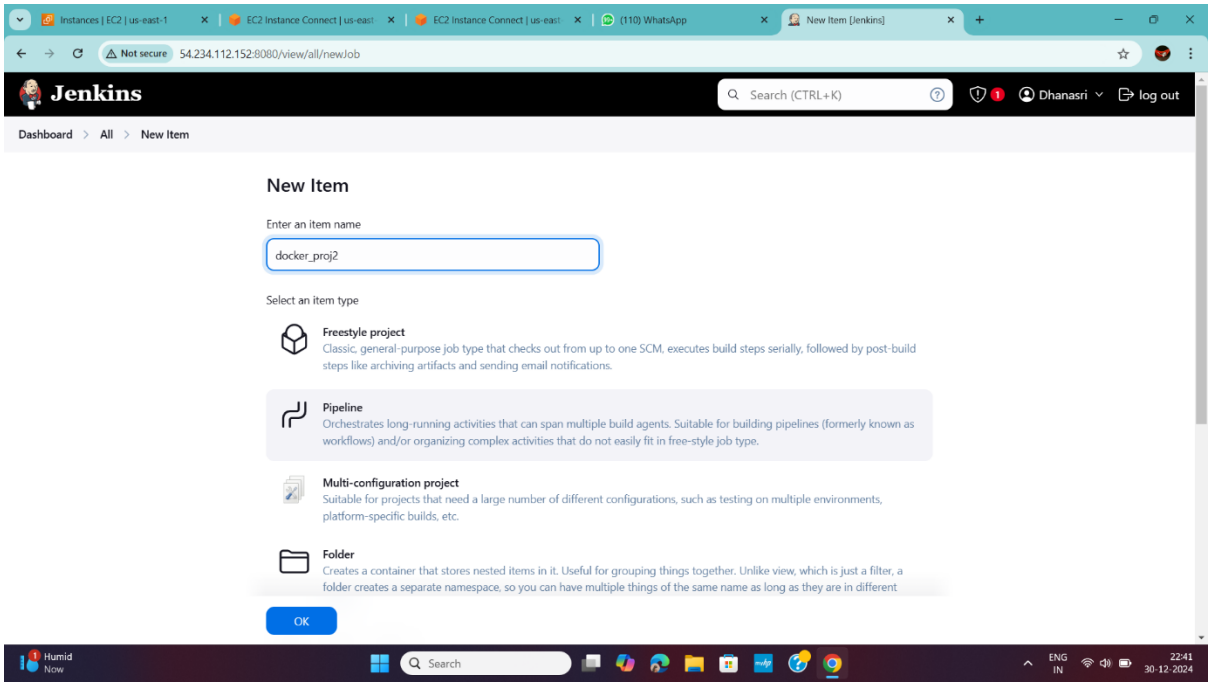


5.Start jenkins with port number 8080 by editing the inbound rules and using the ip address of jenkins instance.



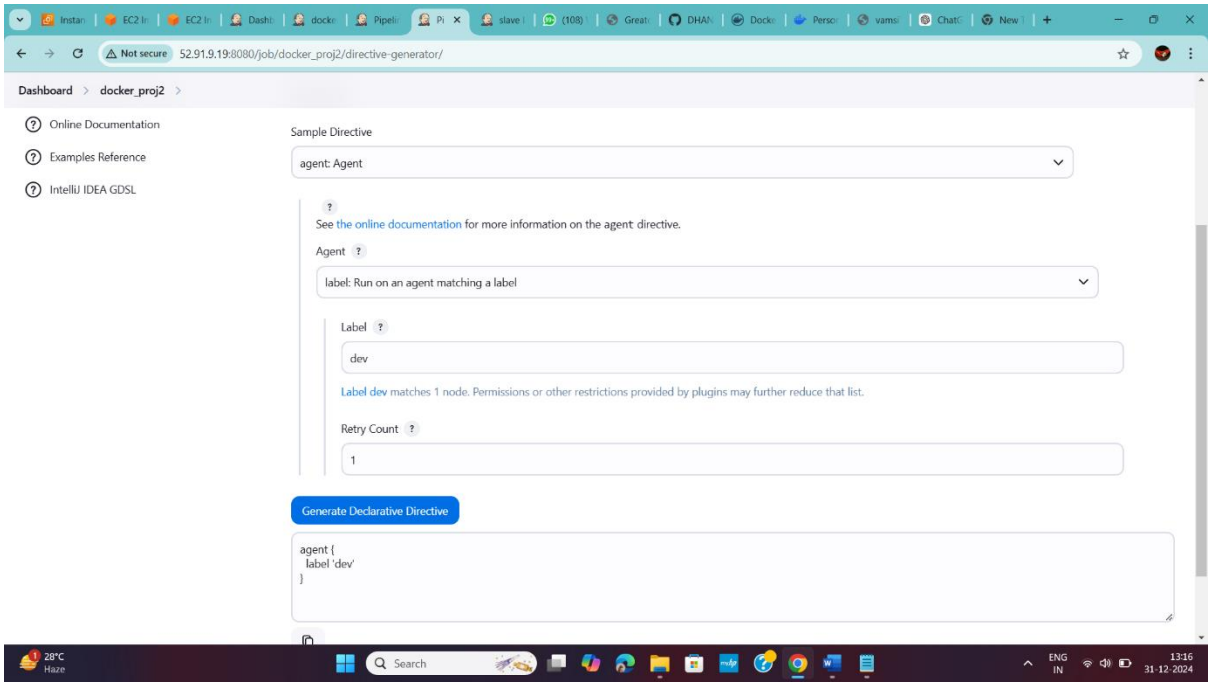
STEP2: Create a Pipeline Project

1. Login to Jenkins Dashboard:
    - Go to the Jenkins dashboard (http://<Jenkins-URL>:8080).
2. Create a New Item:
    - Click on "New Item" on the left-hand side.
    - Enter a name for the project.
    - Select "Pipeline" and click OK.
3. Configure Pipeline Project:
    - In the project configuration, provide the following:
        - Description: Add a description for your project.
        - Source Code Management:
            - Select Git and provide the repository URL.
            - Add credentials if the repository is private.
        - Branch to Build: Specify the branch name (e.g., main or master).



STEP 3: Configure Jenkins to Use the Slave Instance

1. In Jenkins, navigate to jenkins dashboard->job configuration
    - Configure the node by assigning the label.
    - Add the SSH connection details to connect to the slave instance.
2. Test the connection and save the node configuration,make the slave online.



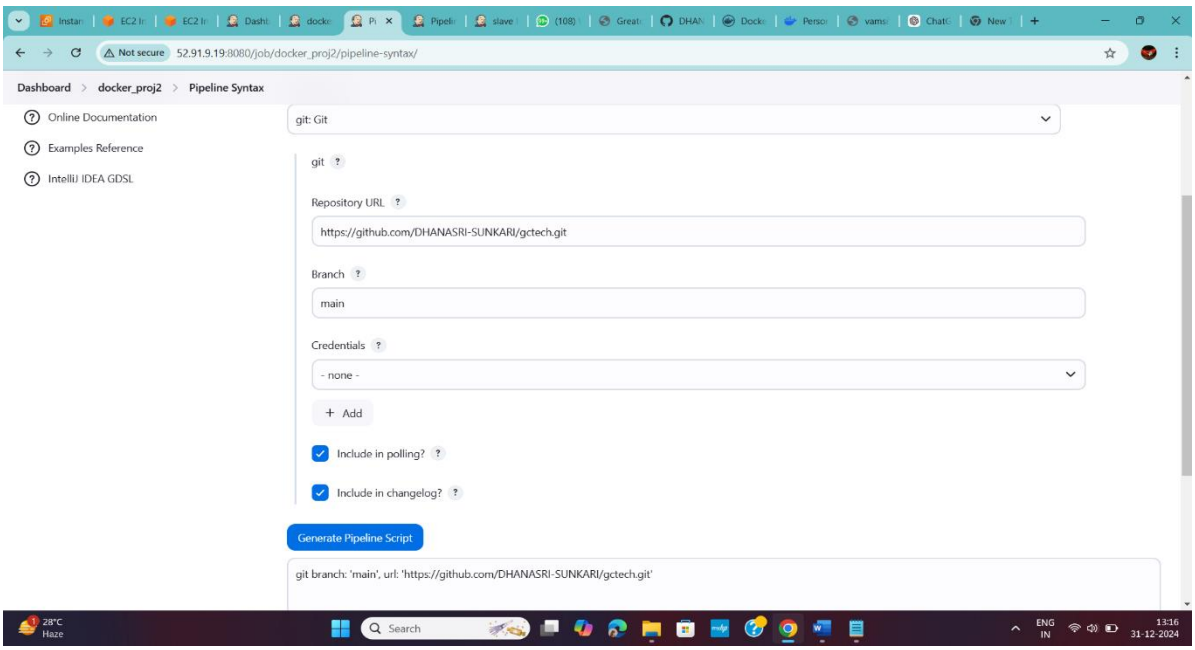
STEP 4: Fork the Sample Repository

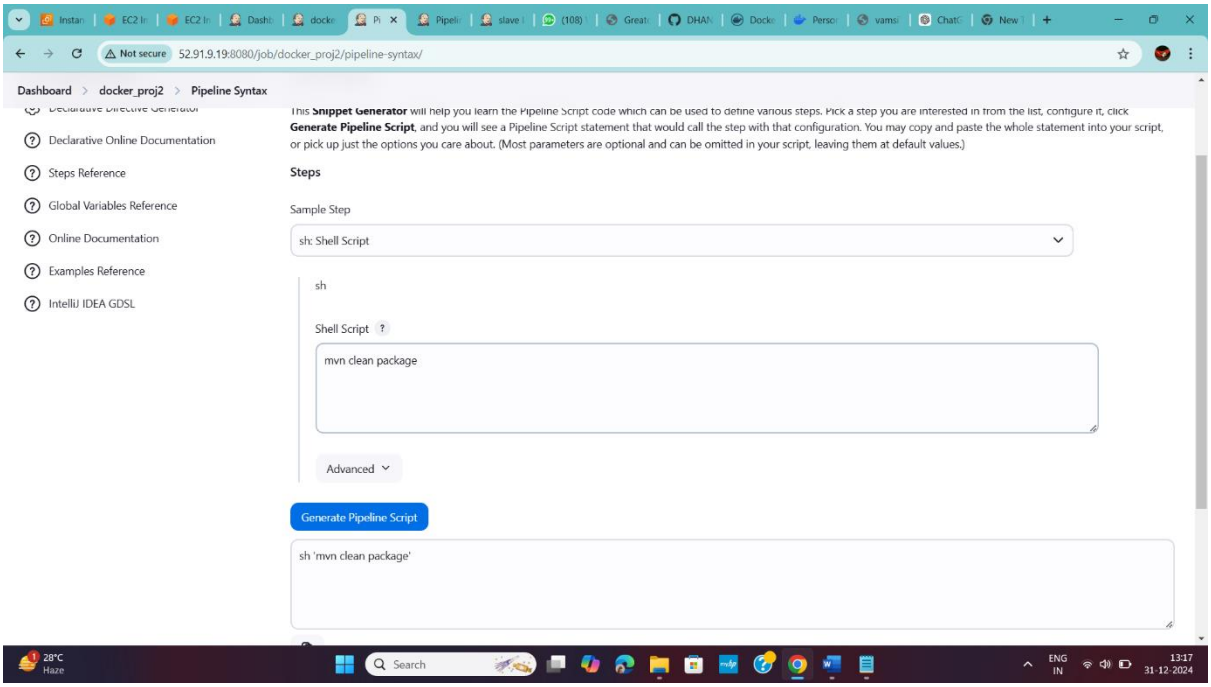
1. Log in to your GitHub account and fork the repository: https://github.com/byramala/gctech.git.

2.configure the GitHub repository in Jenkins:

To generate node-pipeline > configure > pipeline syntax > snippet generator.

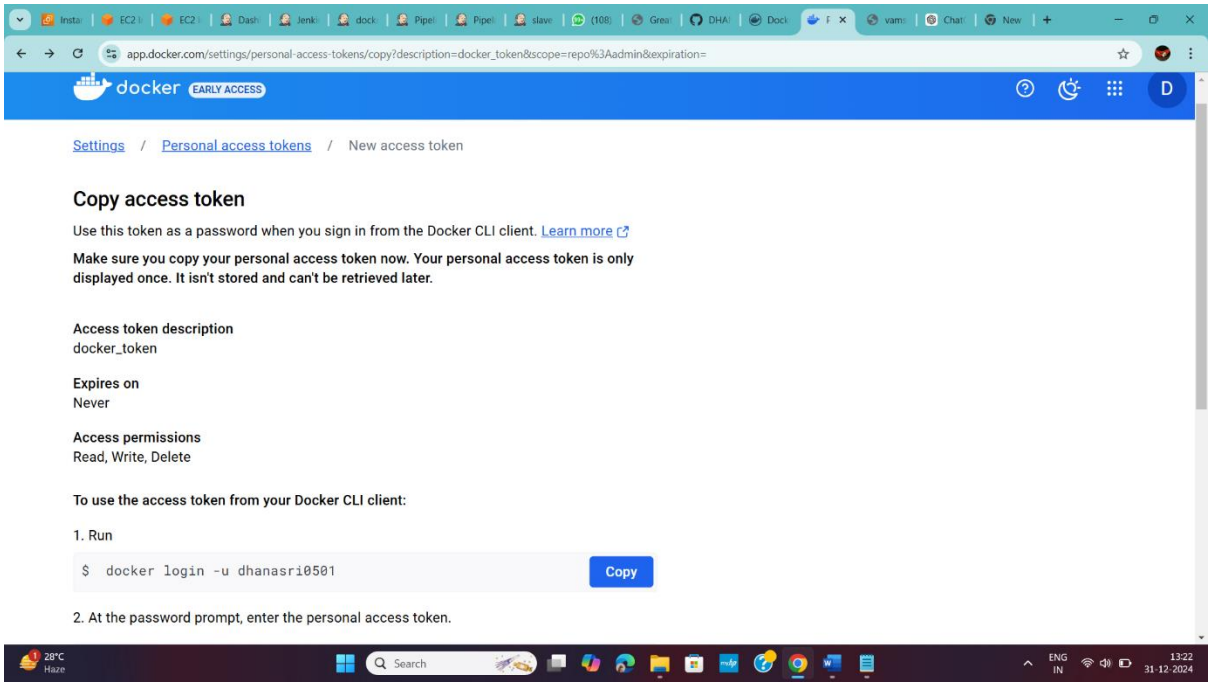
3. Build the war file using Maven.

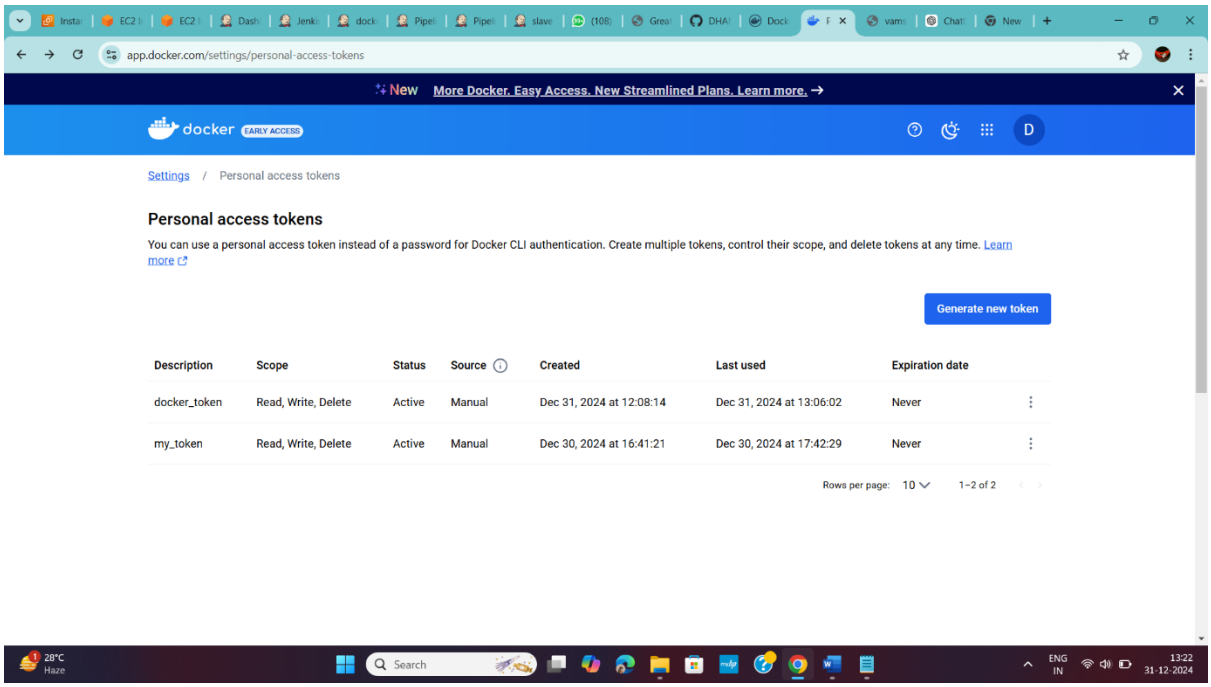


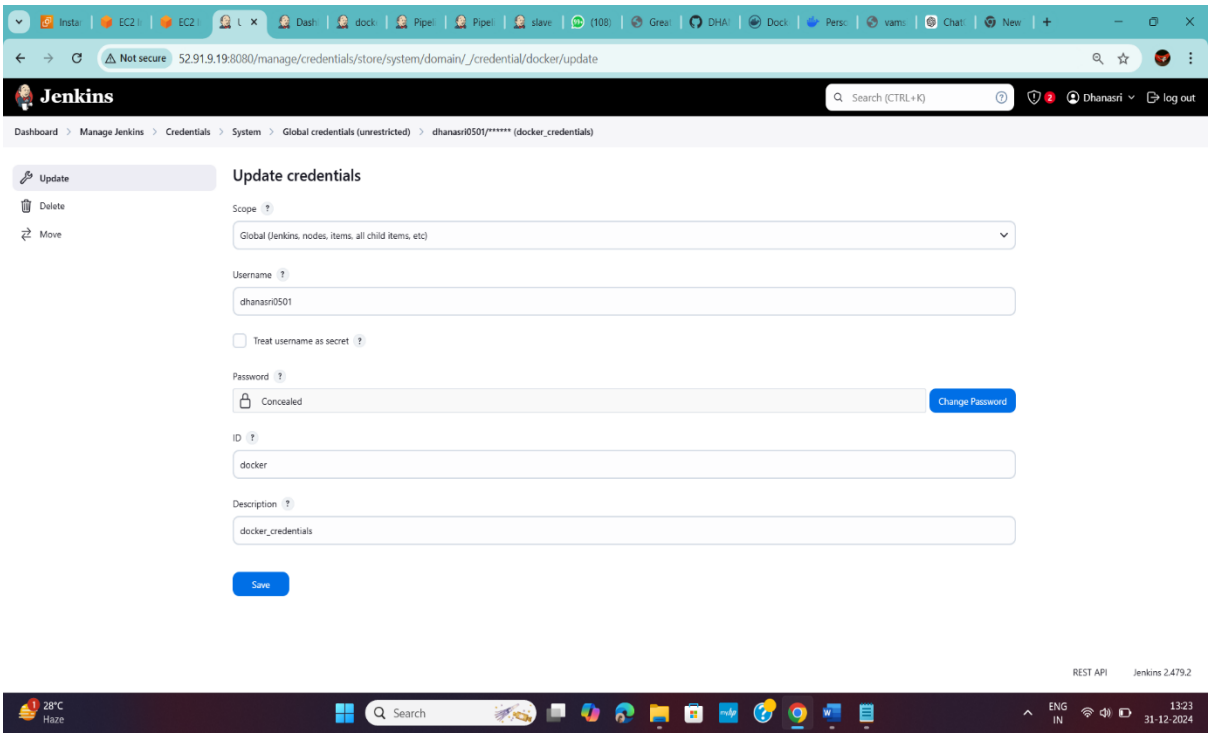


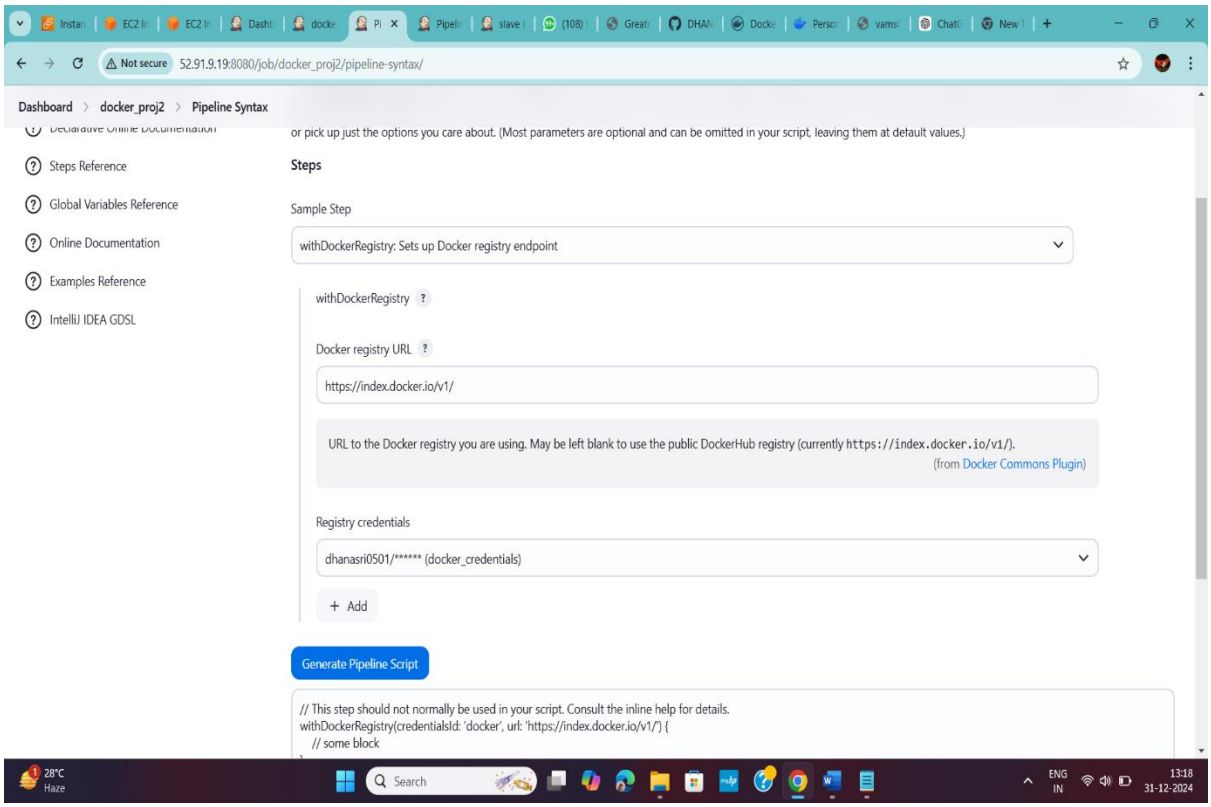
STEP 5: Configure Docker Image Build and Push

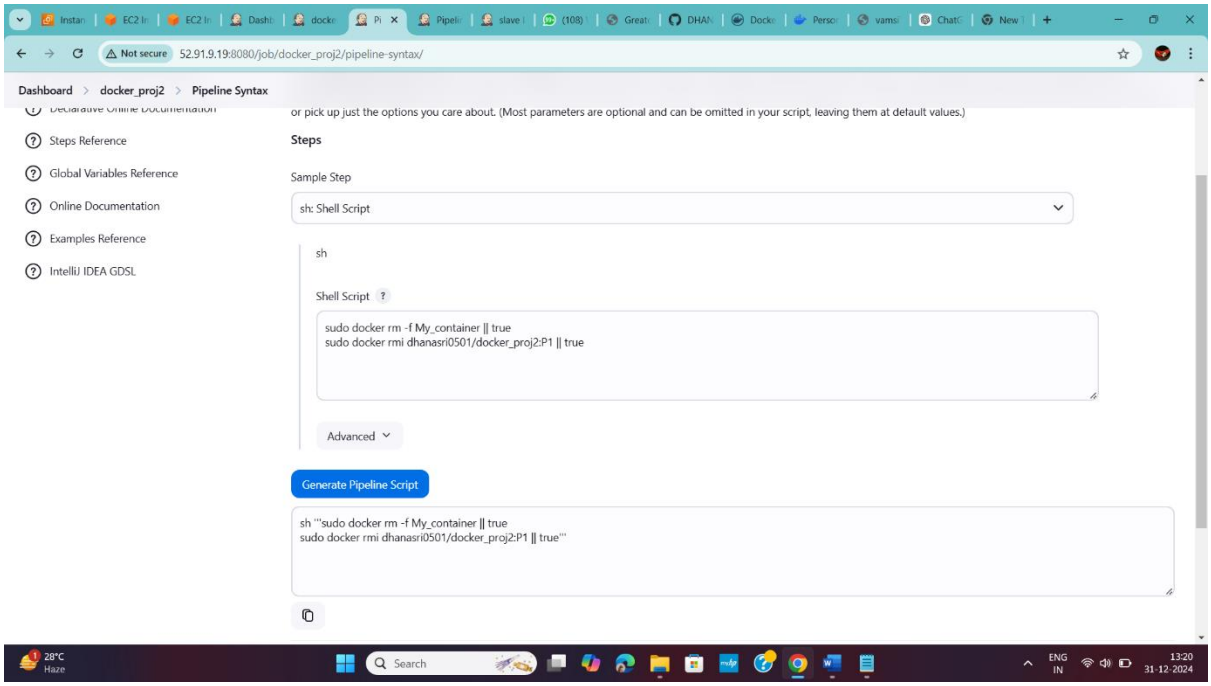
1. Add a post-build step "Execute Shell" with the following script (executed on the slave node)
2. Adding docker credentials username and password in withDockerRegistry.
3. To add this Configure the generated personal access token with read write delete permission at jenkins credentials.
4. Save the job configuration.

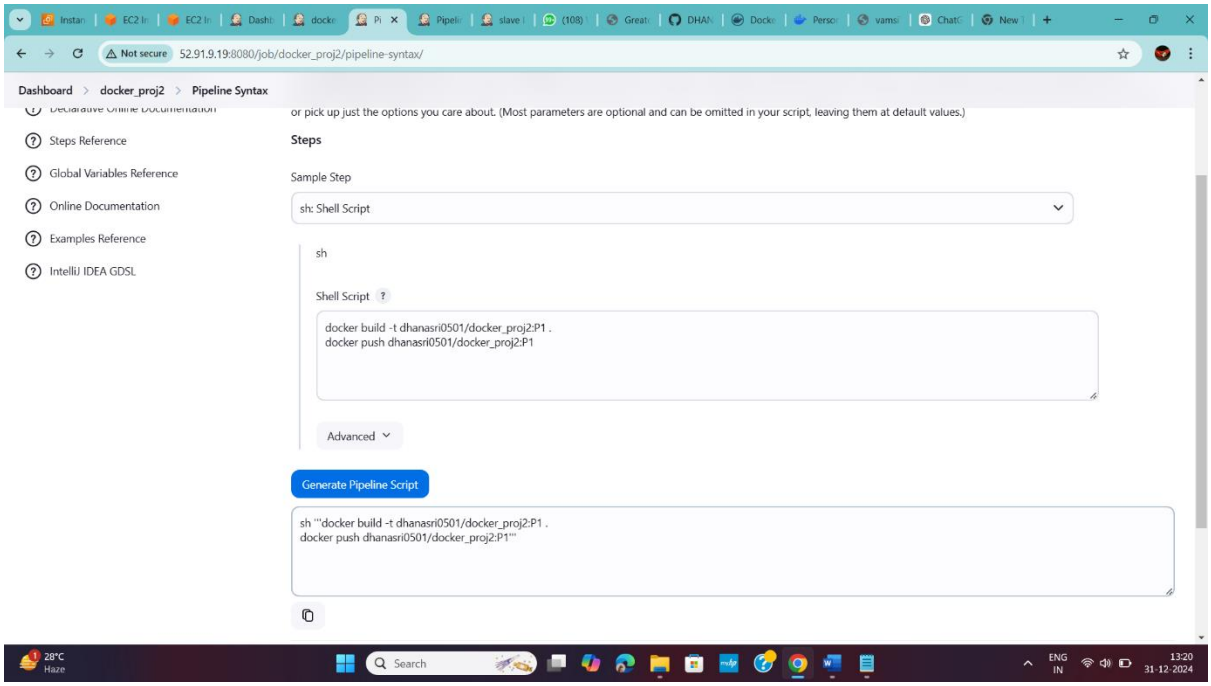




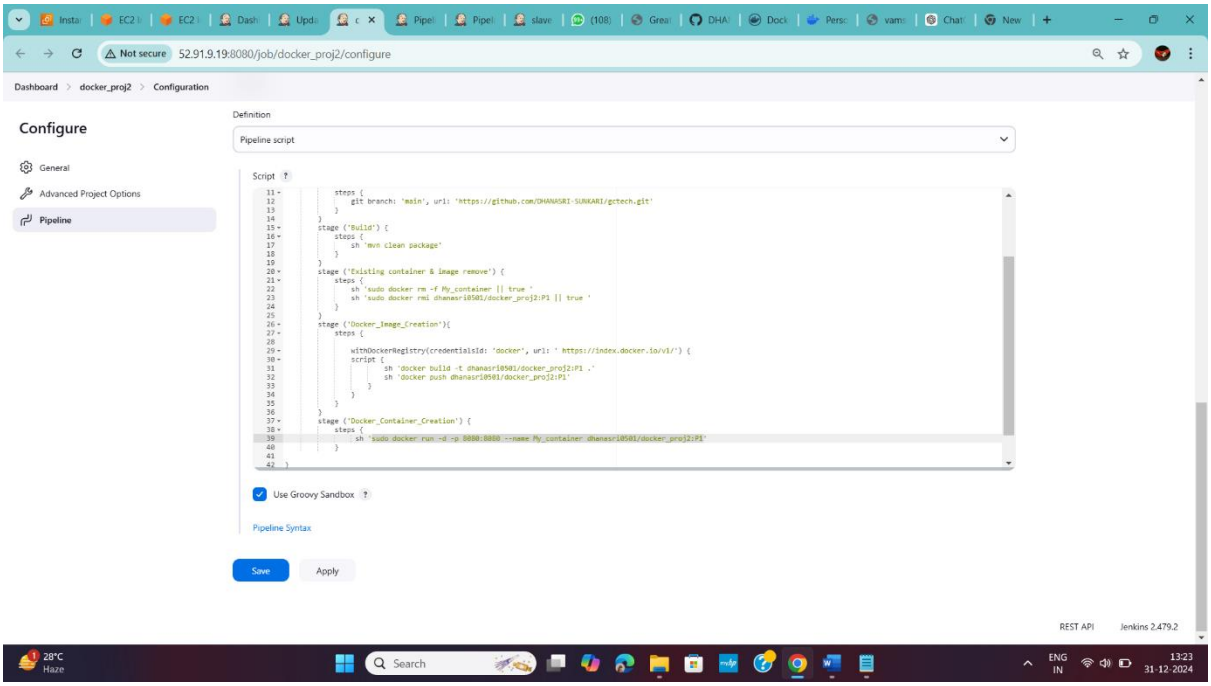

Not secure 52.91.9.19:8080/manage/credentials/store/system/domain/_/credential/docker/update
Jenkins
Search (CTRL+K)
Dhanasri
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > dhanasri0501/****** (docker_credentials)
Update
Delete
Move
Update credentials
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
dhanasri0501
Treat username as secret
Password
Concealed
Change Password
ID
docker
Description
docker_credentials
Save
REST API
Jenkins 2.479.2
28°C
Haze
Search
ENG
IN
13:23
31-12-2024


Not secure 52.91.9.19:8080/job/docker_proj2/pipeline-syntax/
Dashboard > docker_proj2 > Pipeline Syntax
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL
or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)
Steps
Sample Step
withDockerRegistry: Sets up Docker registry endpoint
withDockerRegistry
Docker registry URL
https://index.docker.io/v1/
URL to the Docker registry you are using. May be left blank to use the public DockerHub registry (currently https://index.docker.io/v1/).
(from Docker Commons Plugin)
Registry credentials
dhanasri0501/****** (docker_credentials)
+ Add
Generate Pipeline Script
// This step should not normally be used in your script. Consult the inline help for details.
withDockerRegistry(credentialsId: 'docker', url: 'https://index.docker.io/v1/') {
// some block
28°C
Haze
Search
ENG
IN
13:18
31-12-2024



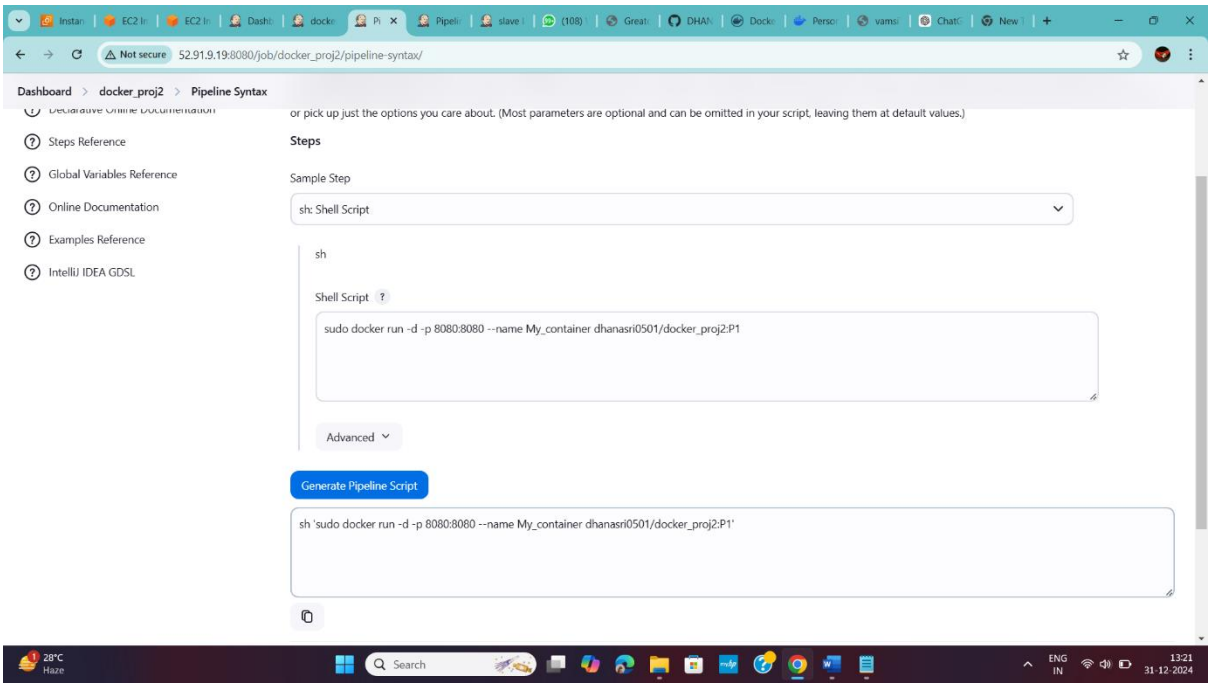


4.Pulling the docker image from the docker hub and deploying the docker image using the docker container



STEP 6: Deploy Docker Image Locally or on Slave Instance

1. Pull the Docker image:
2. Run the Docker container:



Pipeline script:

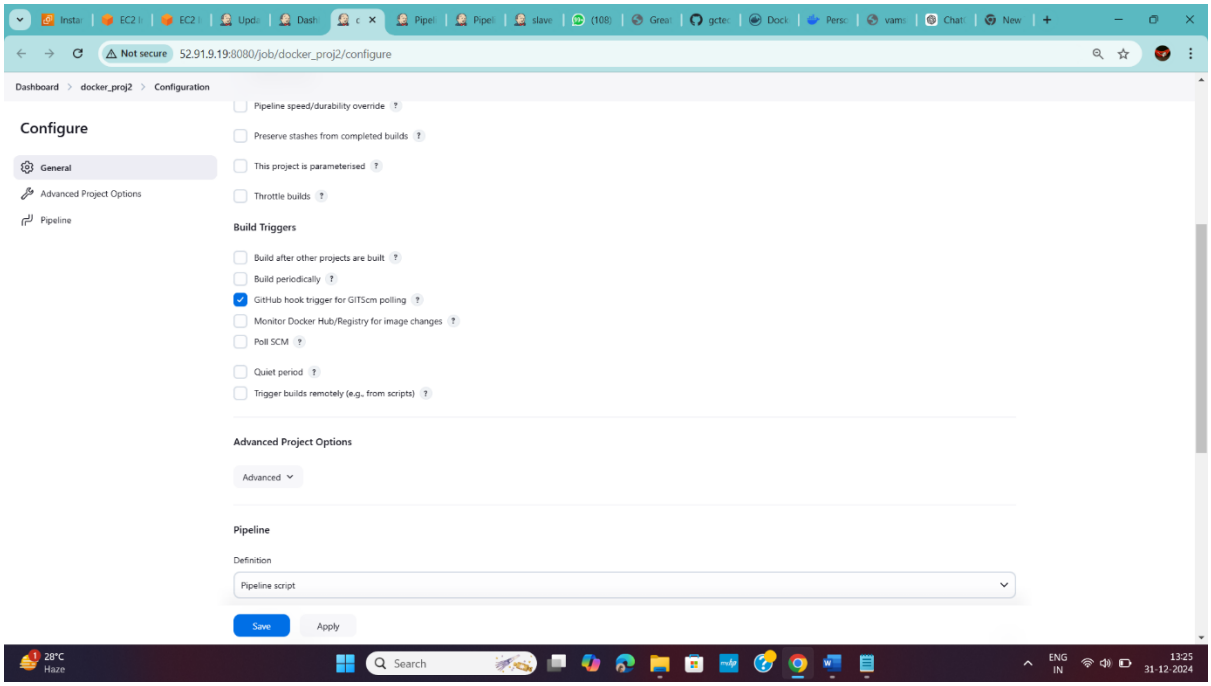
```
pipeline {
    agent {
```

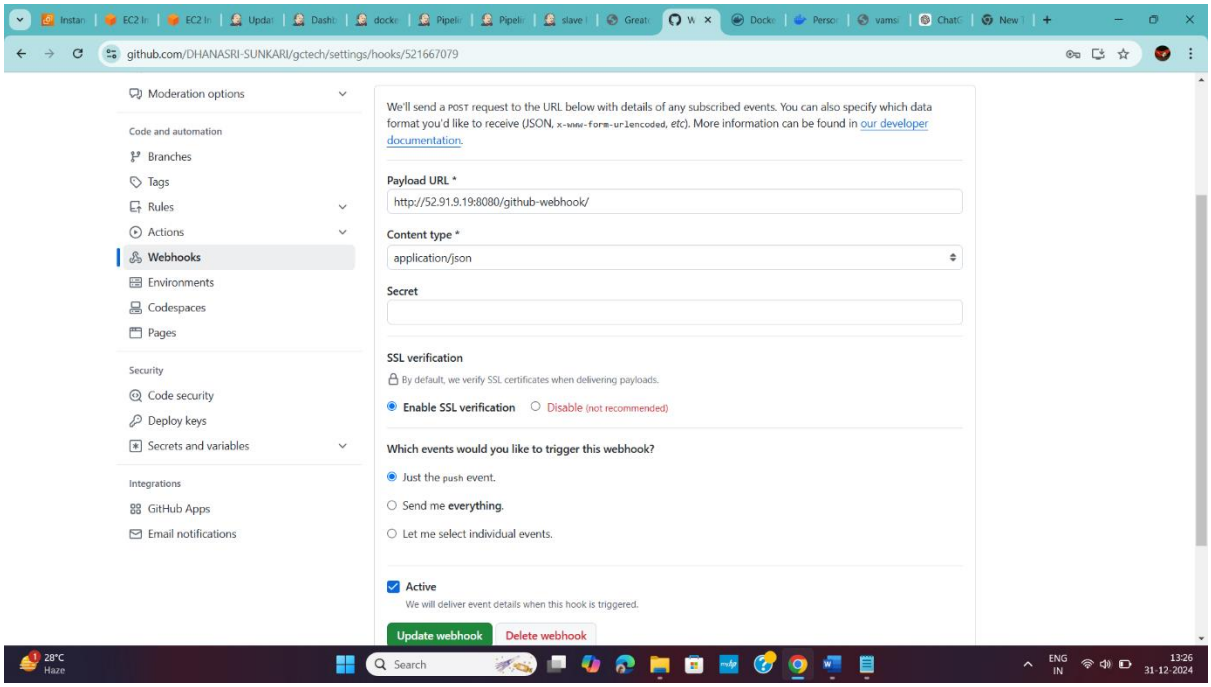
```
        label 'dev'
    }
    stages {
        stage('Clone') {
            steps {
                git branch: 'main', url: 'https://github.com/DHANASRI-
SUNKARI/gctech.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('container & image remove') {
            steps {
                sh 'sudo docker rm -f My_container'
                sh 'sudo docker rmi dhanasri0501/docker_proj2:P1'
            }
        }
        stage ('docker') {
            steps {
                withDockerRegistry(credentialsId: 'docker', url: '
https://index.docker.io/v1/') {
                    script {
                        sh 'docker build -t dhanasri0501/docker_proj2:P1 .'
                        sh 'docker push dhanasri0501/docker_proj2:P1'
                    }
                }
            }
        }
        stage ('run') {
            steps {
                sh 'sudo docker run -d -p 8080:8080 --name My_container
dhanasri0501/docker_proj2:P1'
            }
        }
    }
}
```

STEP 7: Configure Jenkins for Auto-Trigger

1. Install the GitHub Plugin in Jenkins.

2. In your GitHub repository:
    - Navigate to "Settings" -> "Webhooks".
    - Add a new webhook with the Jenkins URL: http://<your-master-ec2-public-ip>:8080/github-webhook/.
3. In Jenkins, configure the job:
    - Under "Build Triggers," select "GitHub hook trigger for GITScm polling"



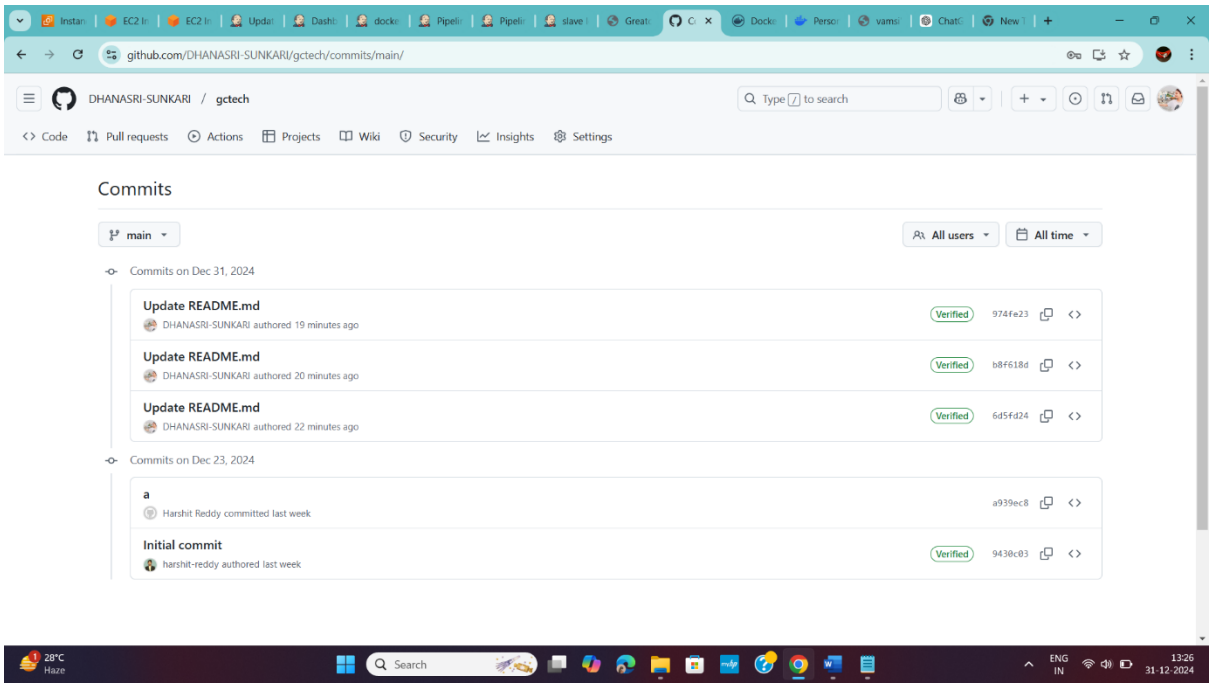


STEP 8: Auto-Trigger Pipeline Execution

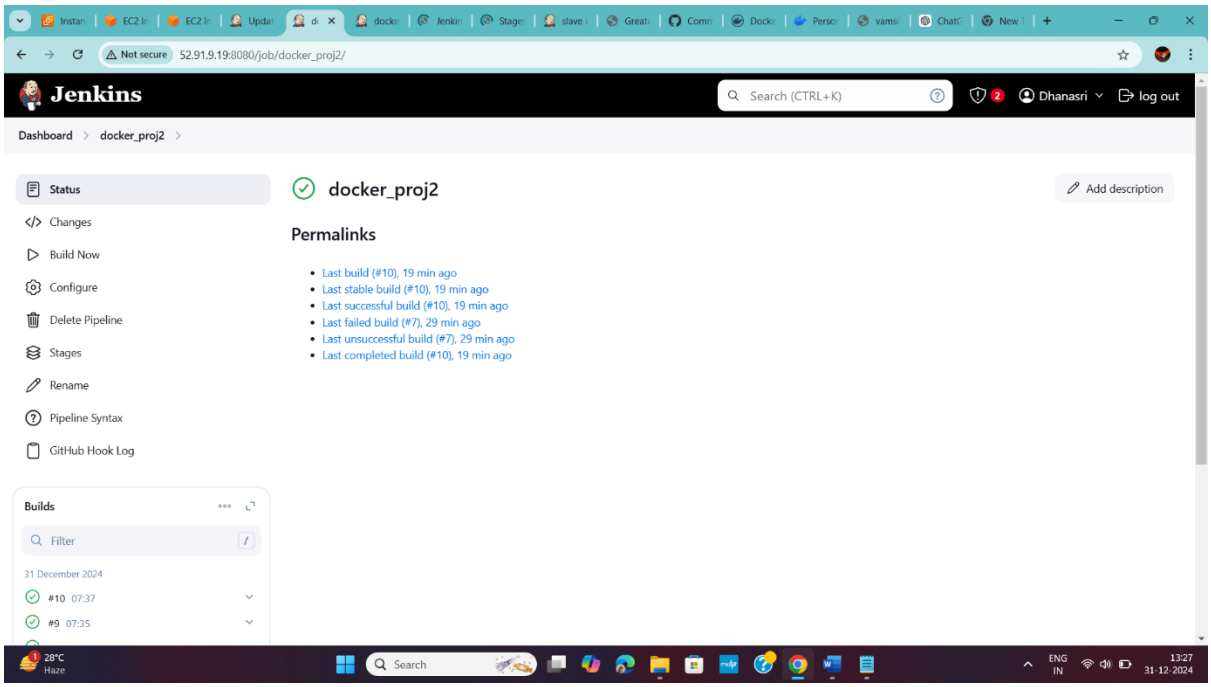
1. Push a commit to the forked repository:
2. git add .
3. git commit -m "Update sample project"

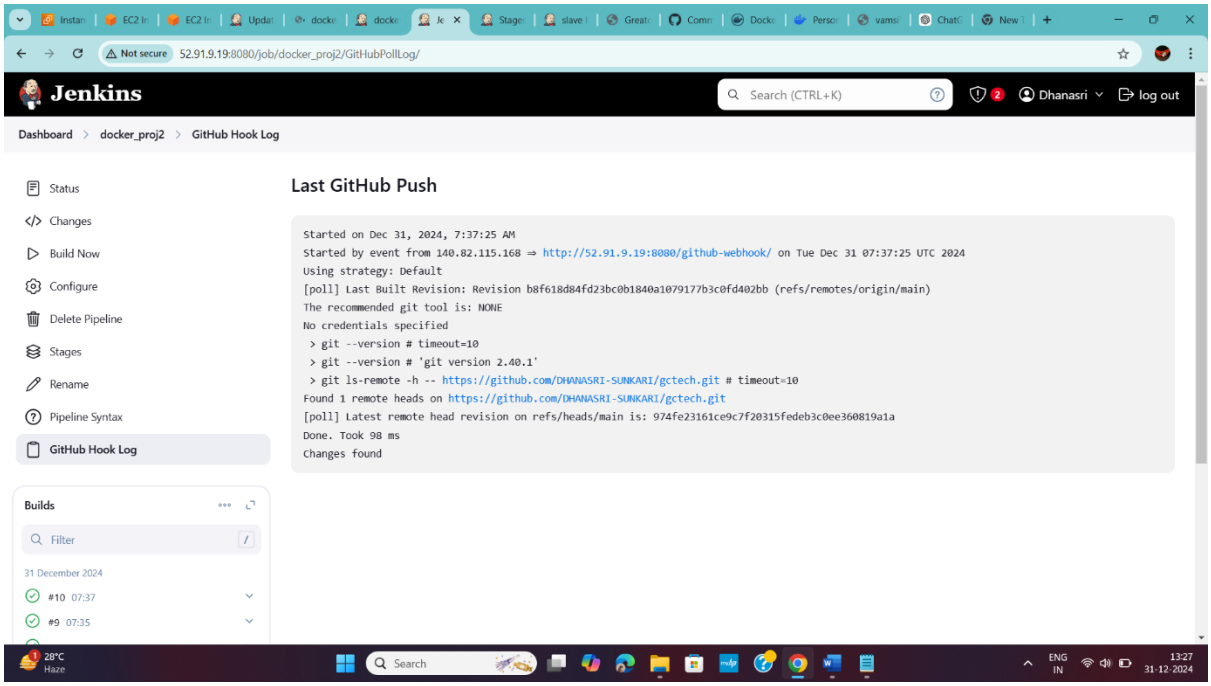
git push origin main

4. Jenkins should automatically start the pipeline and push the updated Docker image to Docker Hub.



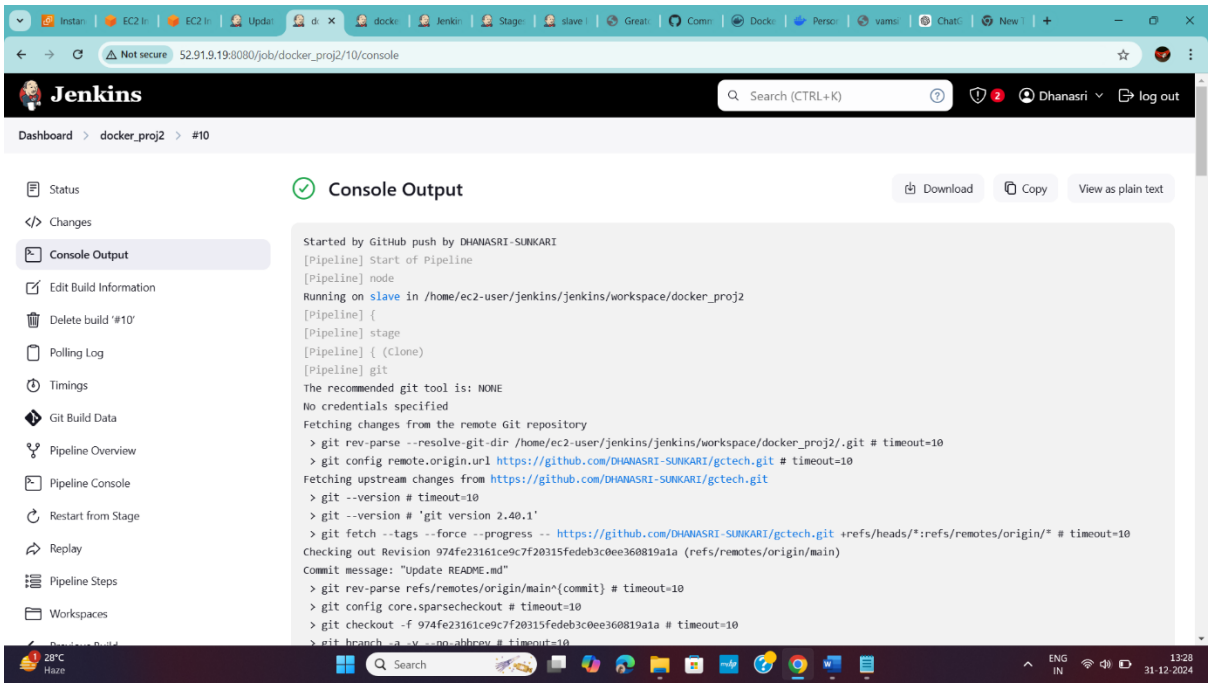
4.Committing changes in GitHub and verifying auto-build





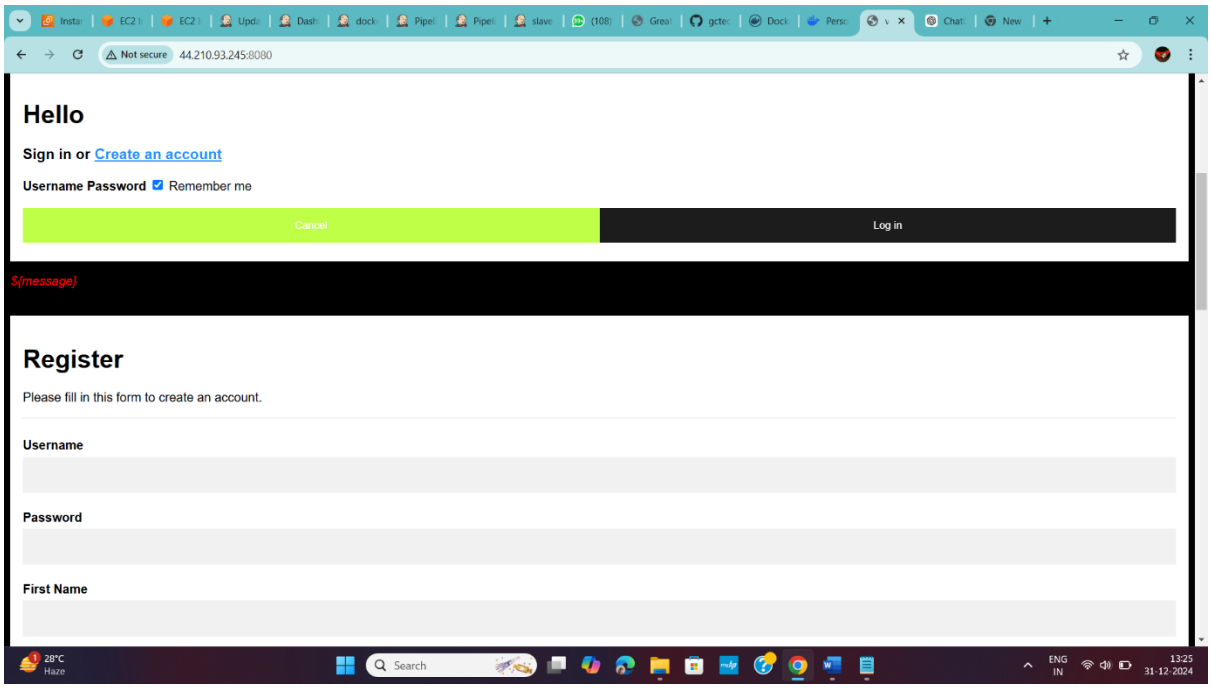
GitHub Hook log

STEP 9: 1.Accessing the application and Deploying container on 8080 Port

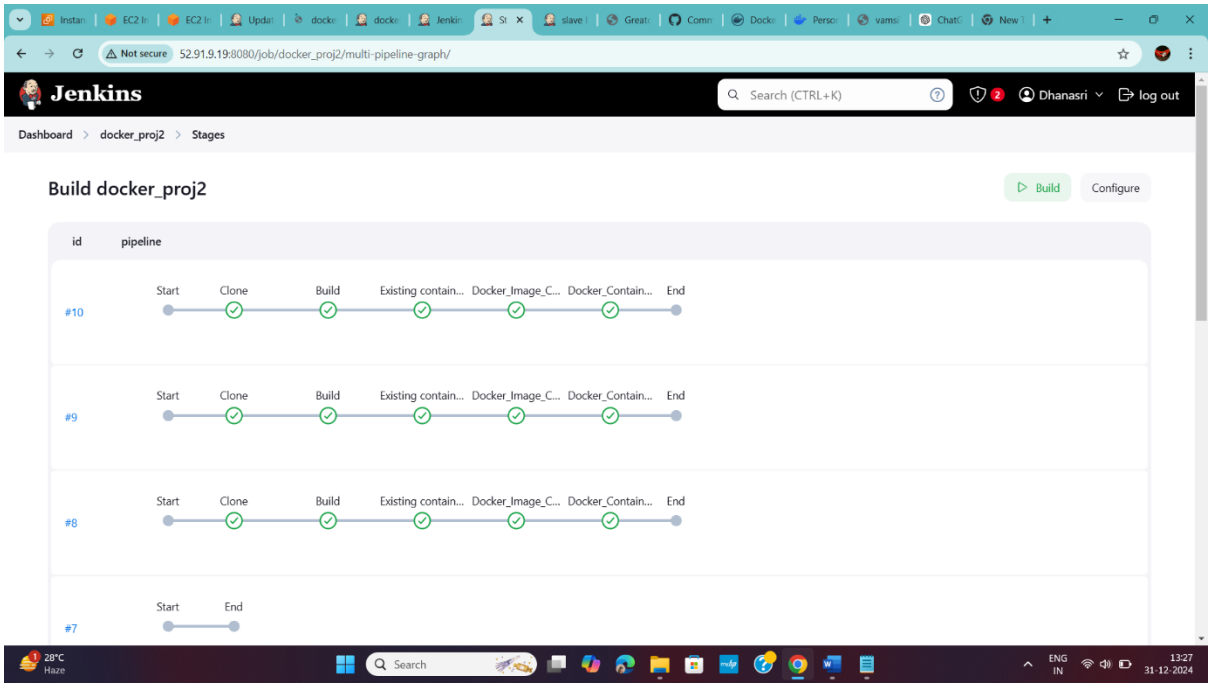


Monitor the console output for each step. #log information is attached in below link:

https://drive.google.com/file/d/1GMSgEZE3XDHUkrFDGjtH-Rk3tshEGzOb/view?usp=sharing



2.Build steps of pipeline project



Troubleshooting Common issues:

- **Pipeline Syntax Issues**

- Verify the Jenkinsfile syntax using the built-in **Pipeline Syntax** tool.
- Check for missing or misplaced braces, stages, or steps.

- **Credential Issues**

- Ensure required credentials (e.g., API keys, SSH keys) are correctly configured in Jenkins.
- Validate if the credentials are properly referenced in the pipeline script.

- **Plugin Problems**

- Ensure all required plugins are installed and up-to-date.
- Restart Jenkins if plugin updates cause inconsistencies.

- **Environment Variables**

- Debug by printing environment variables (env.) in the pipeline.
- Confirm that required environment variables are set.

- **Agent/Node Issues**

- Verify that the agent/node is online and has the required dependencies installed.
- Check resource availability (e.g., CPU, memory).

- **SCM Configuration**

- Confirm the source code repository URL and branch name are correct.
- Check for SCM plugin compatibility and access permissions.

SUBMITTED BY

DHANASRI SUNKARI