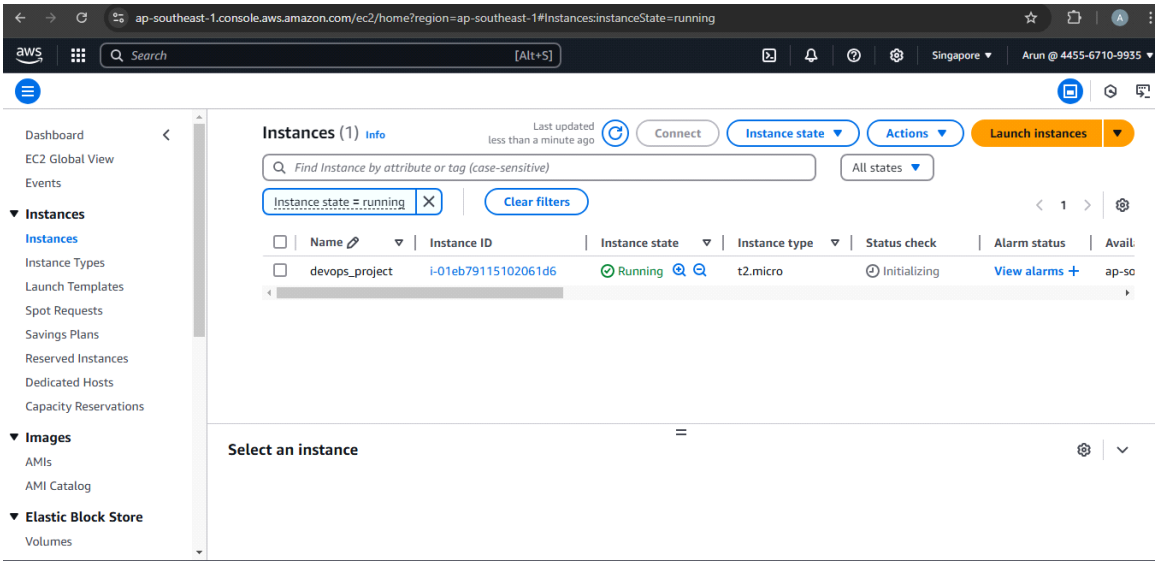
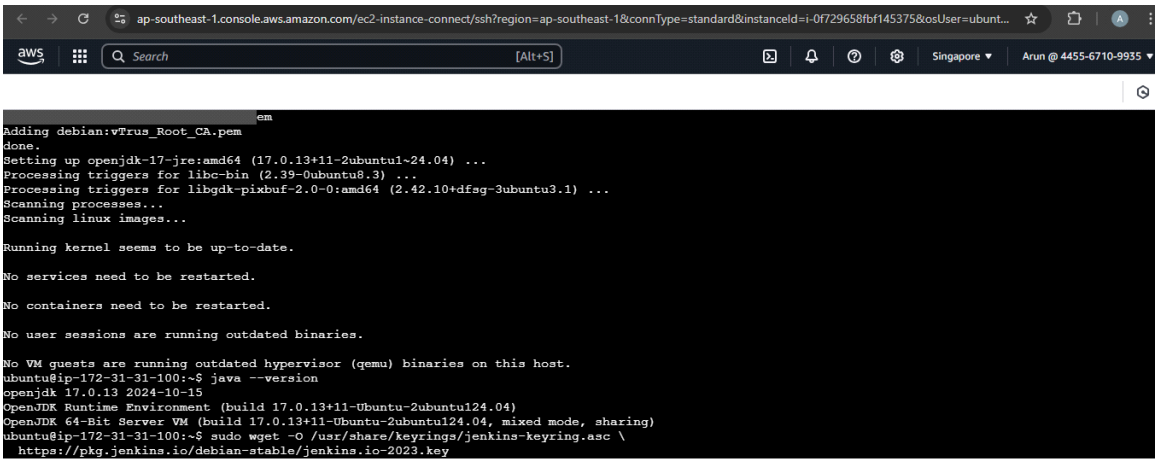
DEVOPS LIFE CYCLE IMPLEMENTATION FOR ABODE SOFTWARE

Presented by

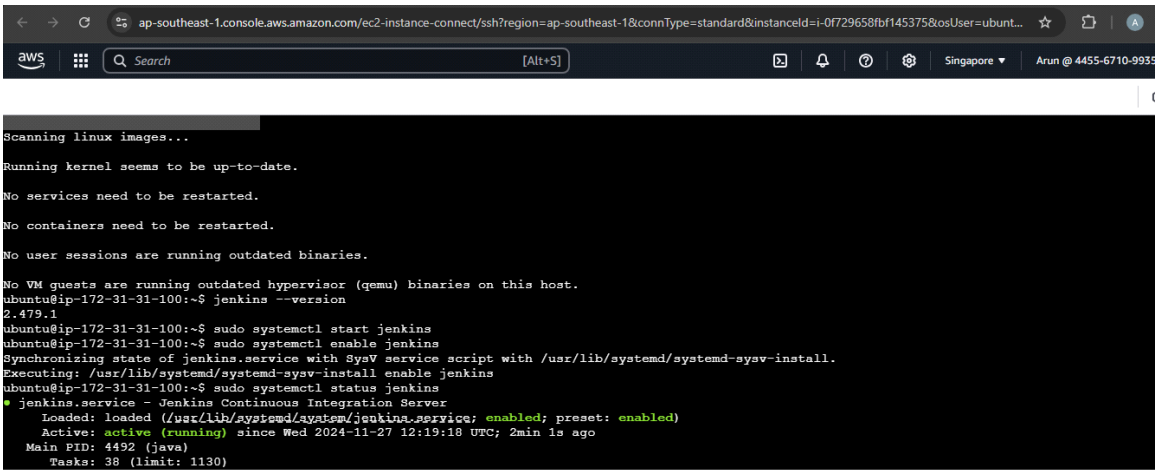
ARUNARATH S



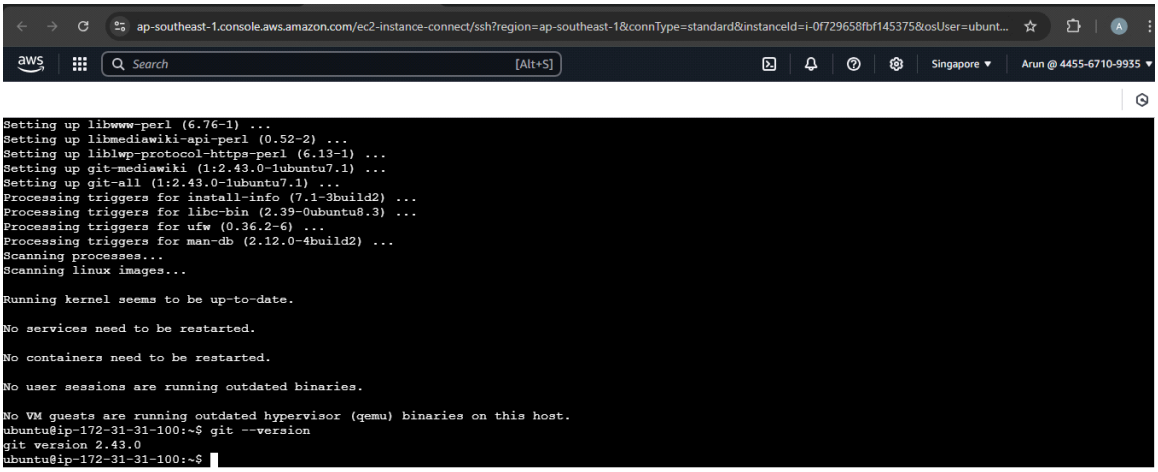
- Login to AWS console
- Create a EC2 instance



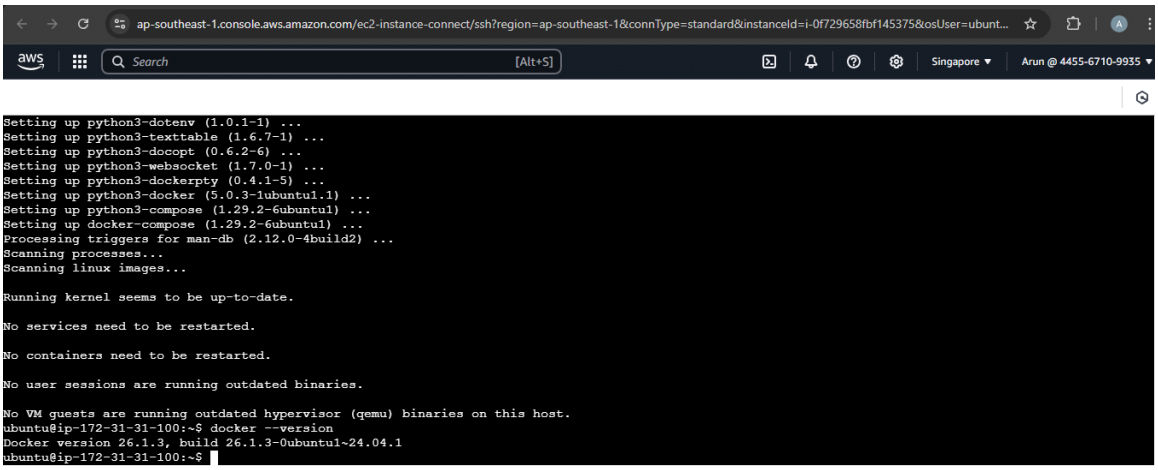
- Connect to the AWS EC2 instance
- Install the Java



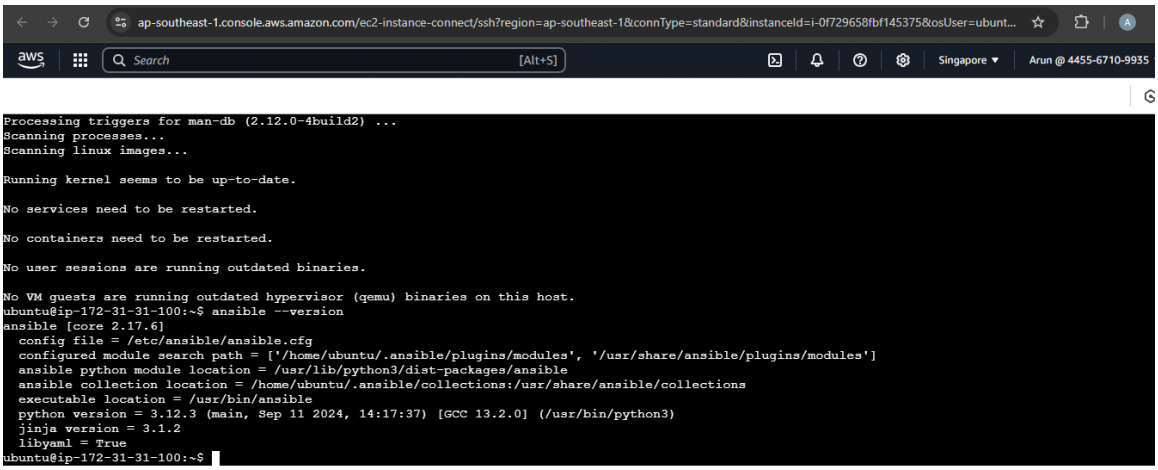
- Jenkins was installed



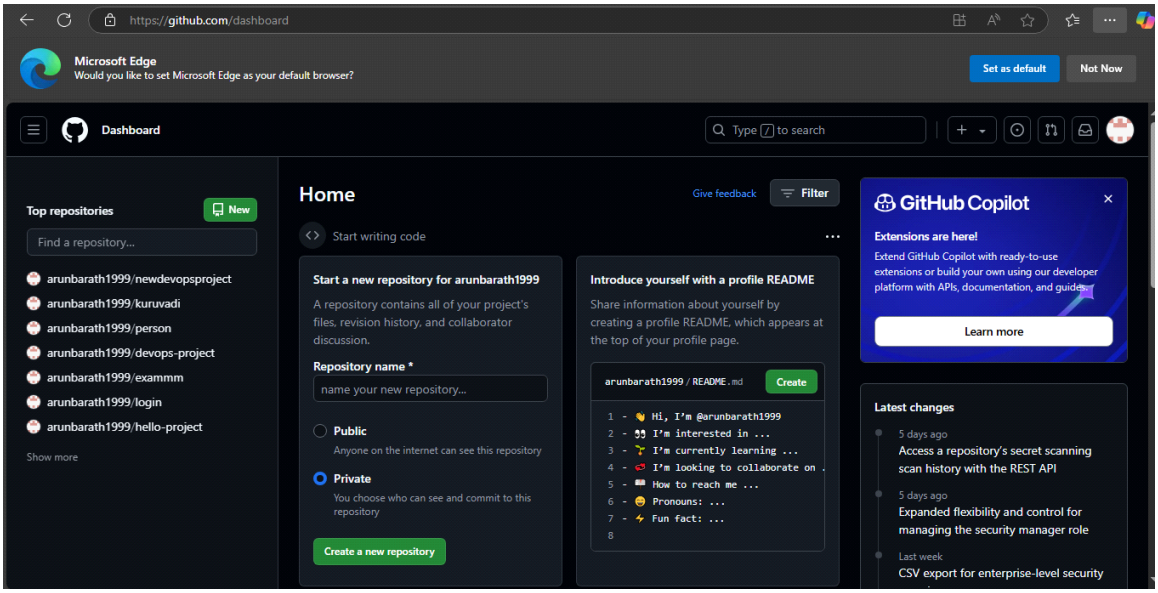
- Git was installed



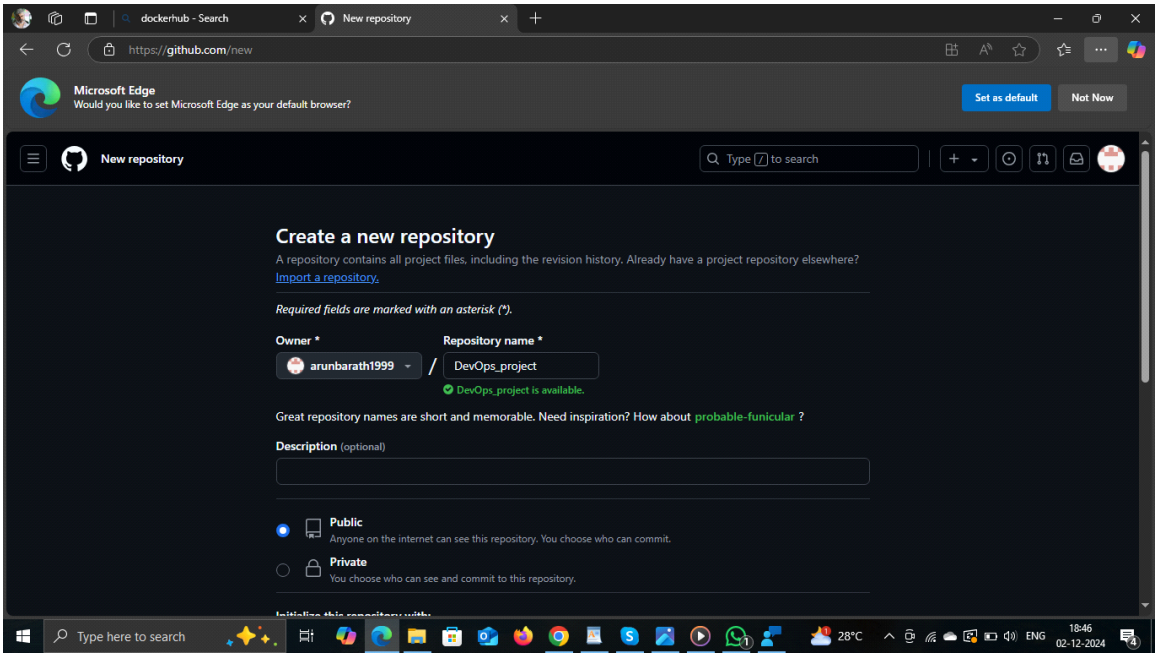
- Docker was installed



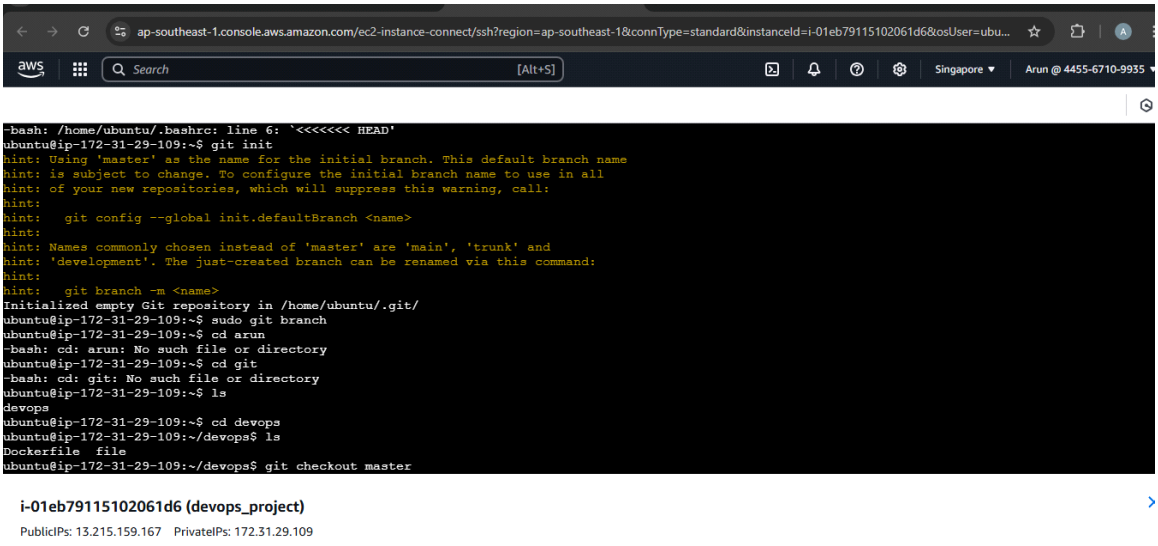
- Install the Ansible software



- In Git hub a new repository



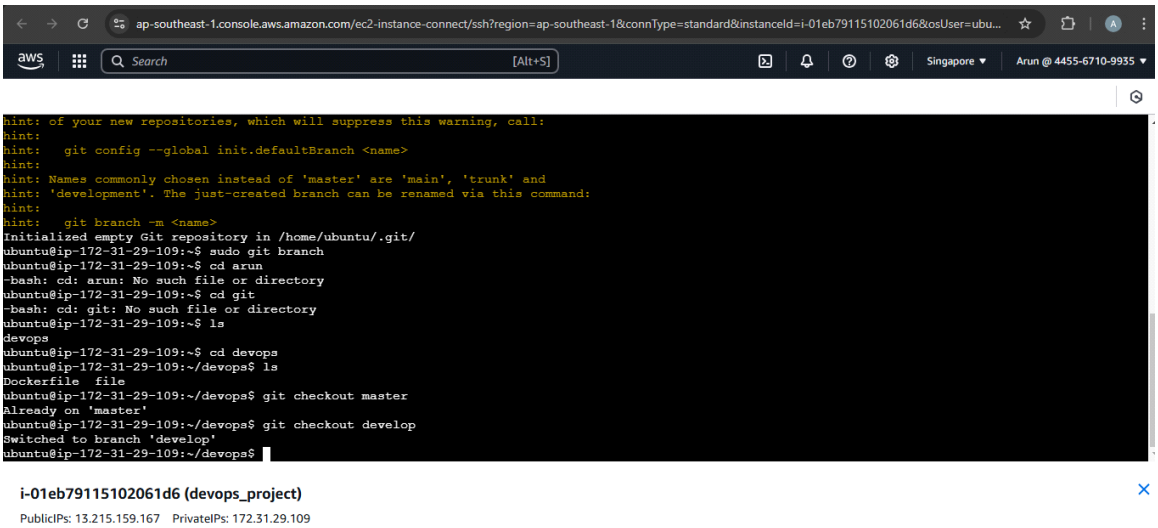
- Create a Git hub repository and copy the repository URL



- Cerate a one dirctory for initializing the Git
- Change the directory

  cd Git

  git init (to initializing the Git**)**

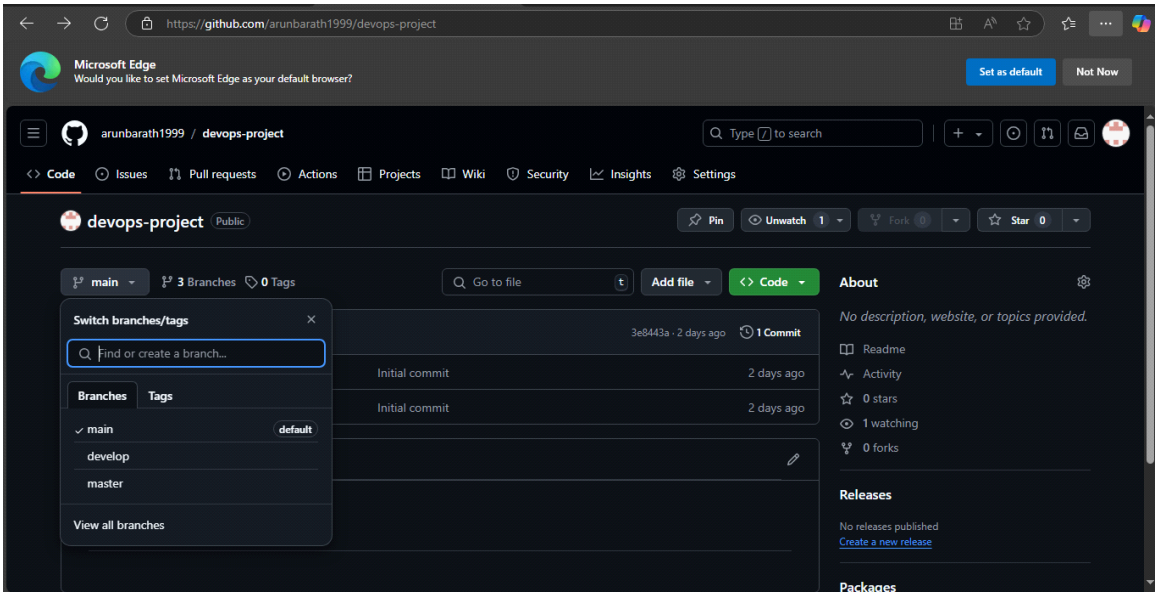


- In local repository create a file
- and commit the file
- create a branch

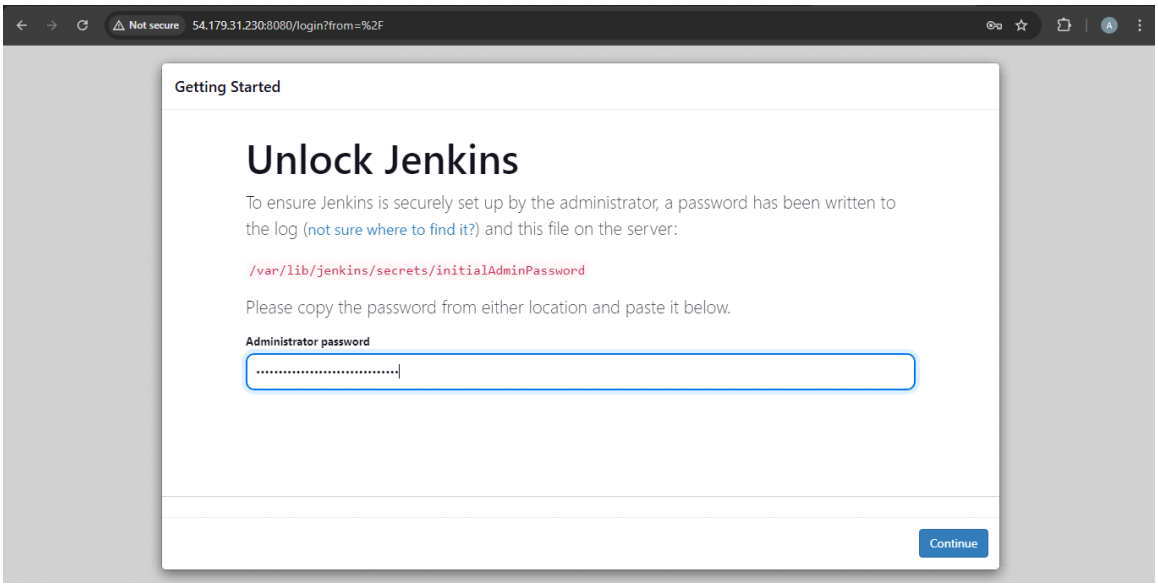
sudo git branch develop(for creating the branch)

```
22  mkdir devops
23  cd devops
24  touch file
25  vim file
26  git init
27  git add .
28  git remote add origin https://github.com/arunbarath1999/devops-project.git
29  git push origin master
30  git commit -m "welcome to devops"
31  git push origin master
32  git branch develop
33  git push origin develop
```

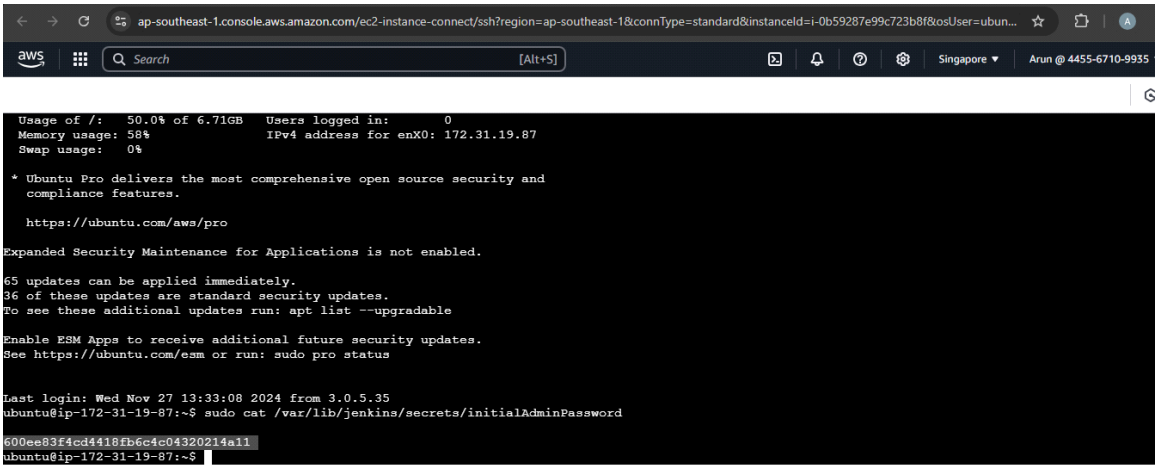
- sudo git remote add origin **<git hub repository>**
- Git push origin master **(for pushing the master branch to Git hub)**
- Git push origin develop **(for pushing the develop branch to Git hub)**



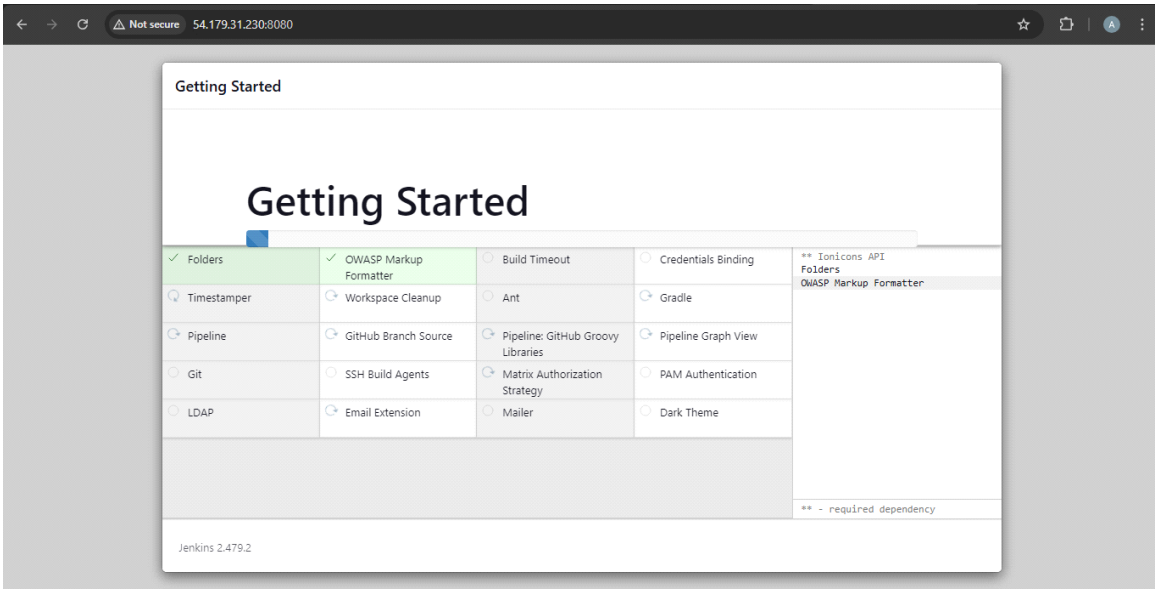
- Two branches from local repo was pushed to Git hub



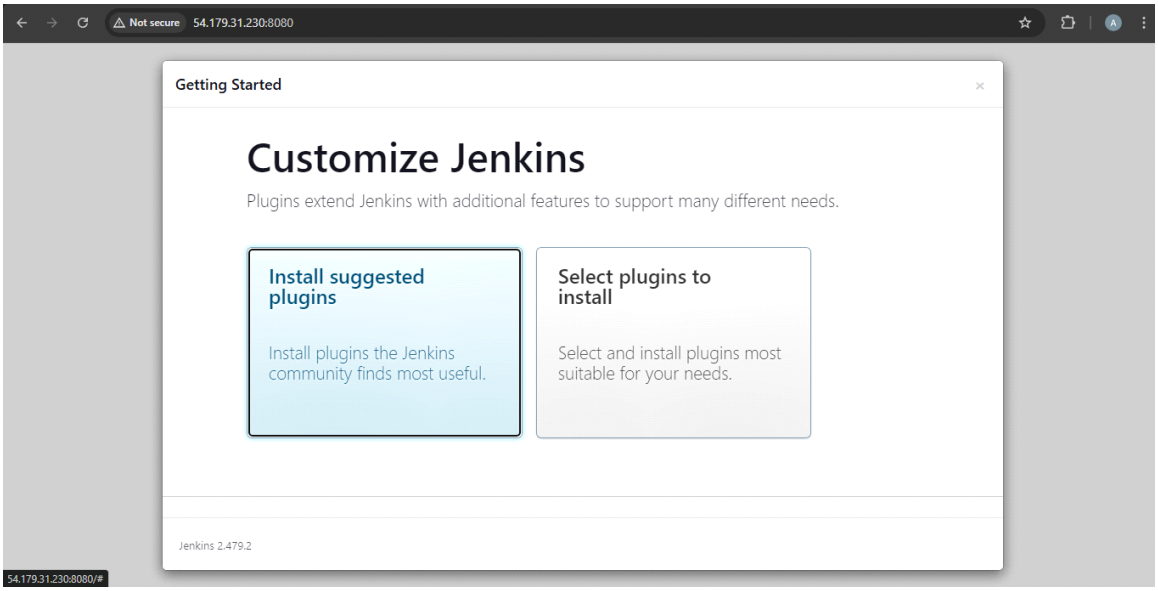
- now copy the public ip address of the Ec2 instance where jenkins was installed **<public ip address>:8080**
- copy the path for administrator password



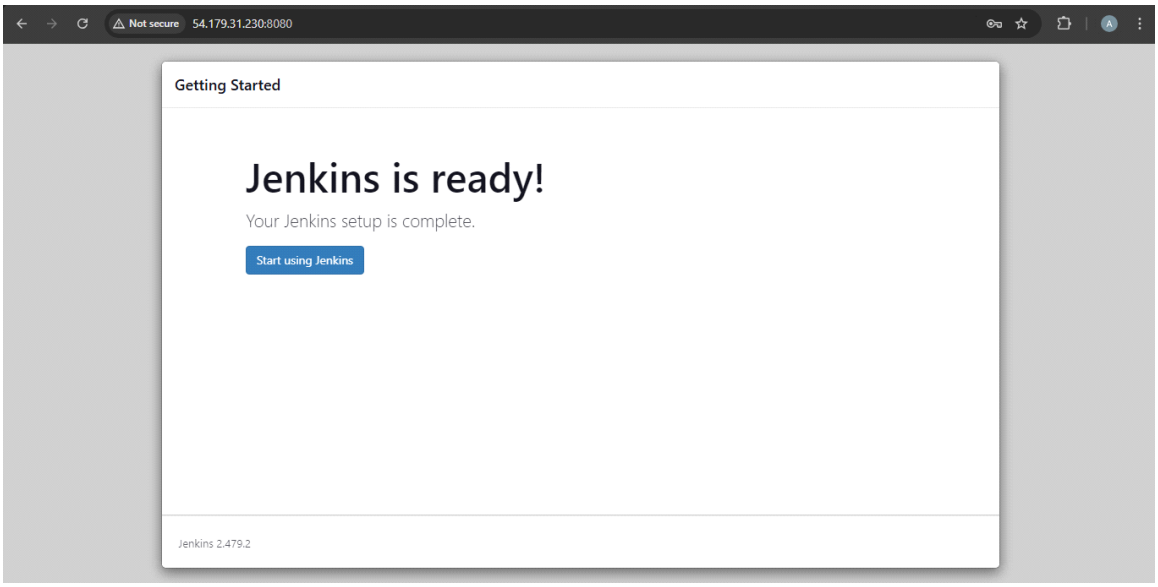
- sudo cat /var/lib/jenkins/secrates/initialadminpassword **(for administrator password)**
- copy the password and past in the Administrator password



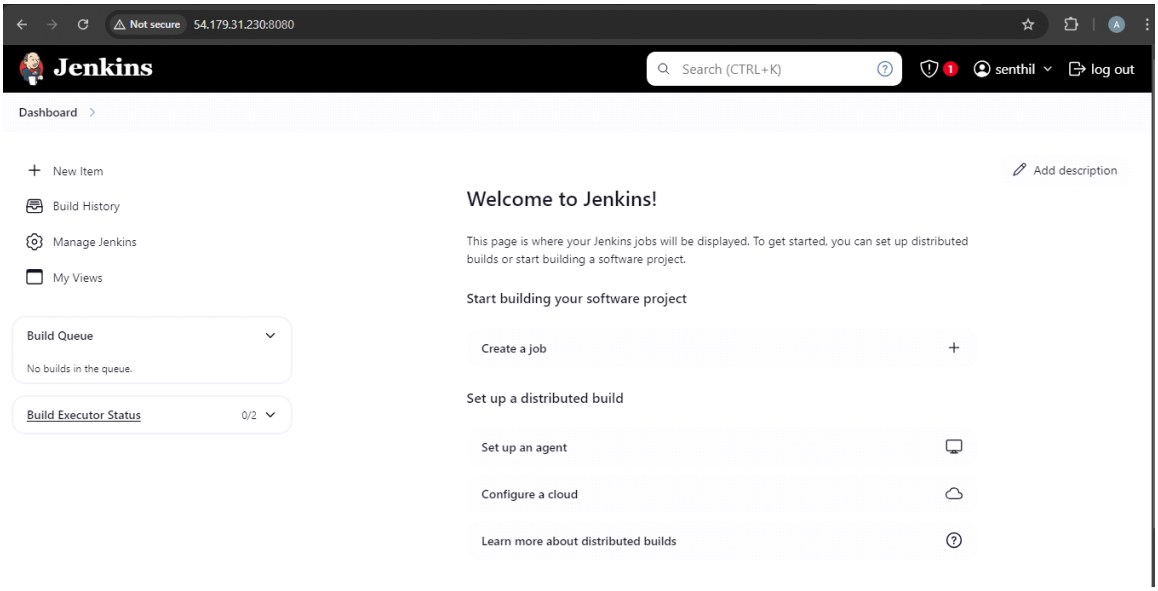
- required plugins was getting installed



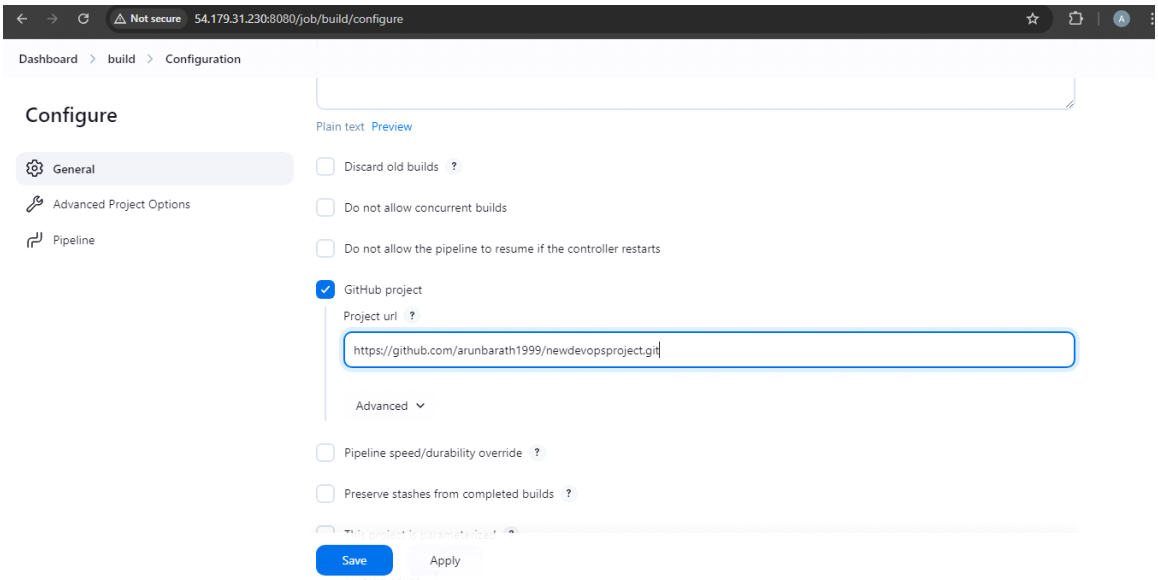
- selected the install suggested plugins



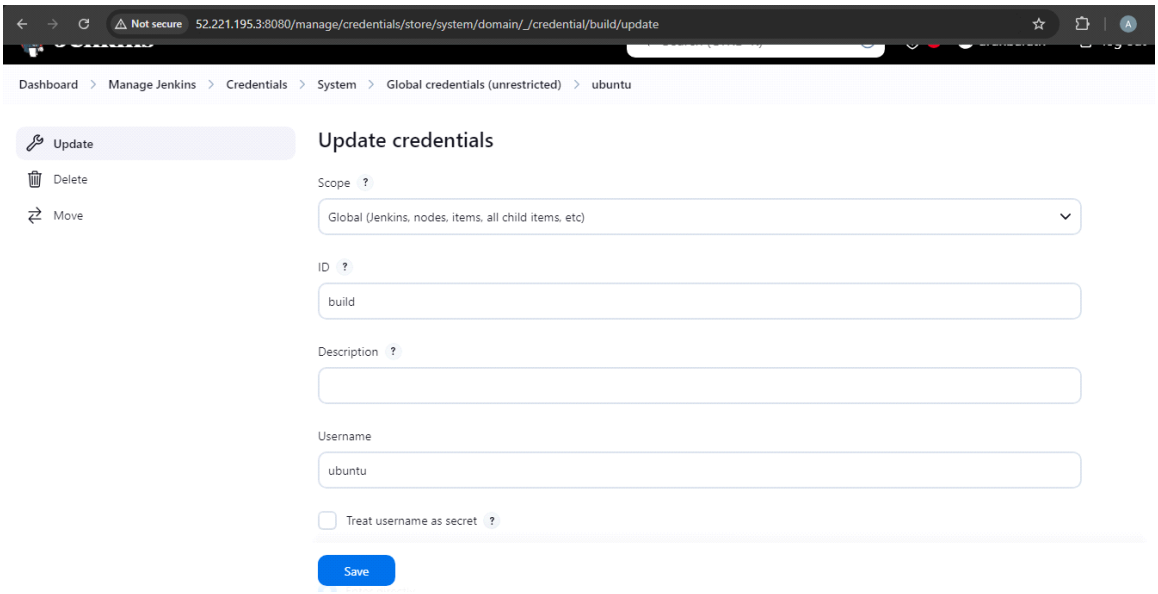
- Jenkins is ready



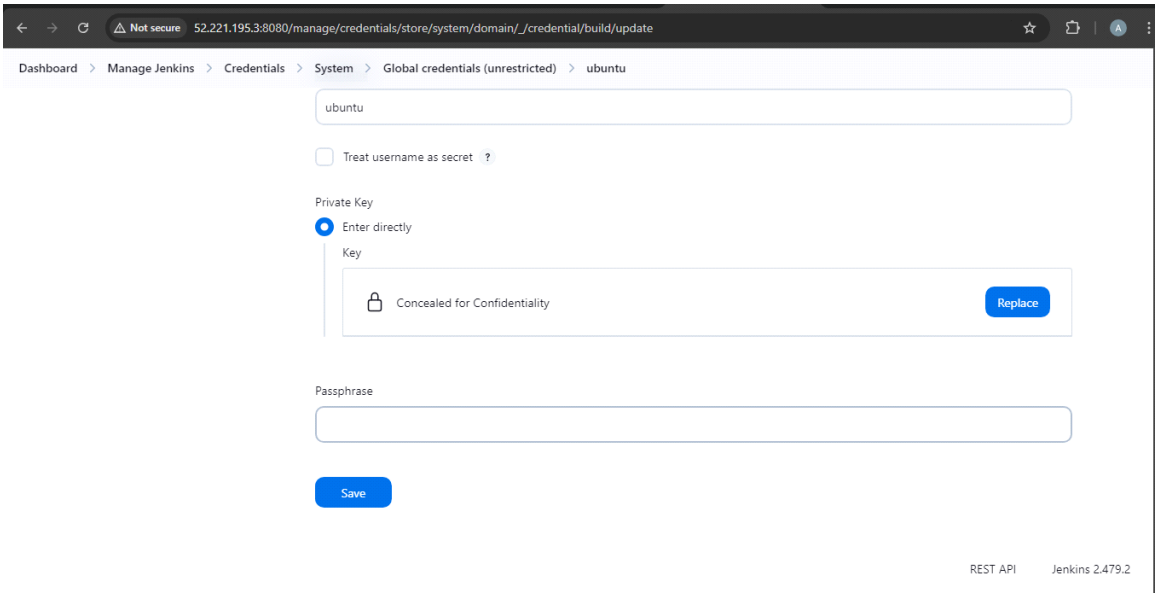
- click on new item for creating the job



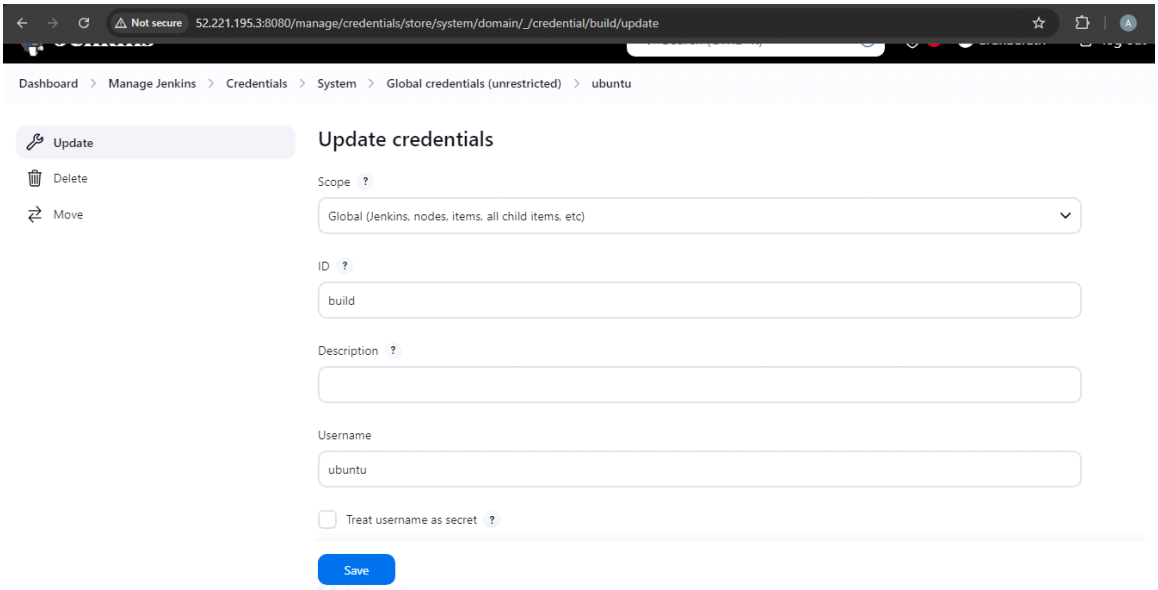
- click on the github project
- in project url past the git hub repository url



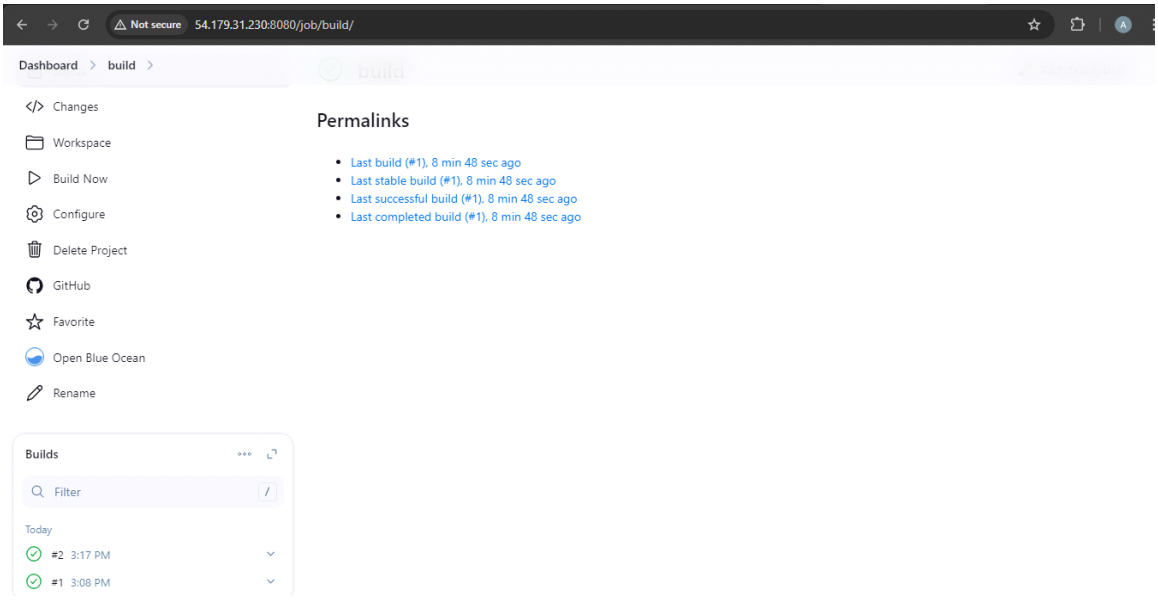
- create a Jenkins credentials

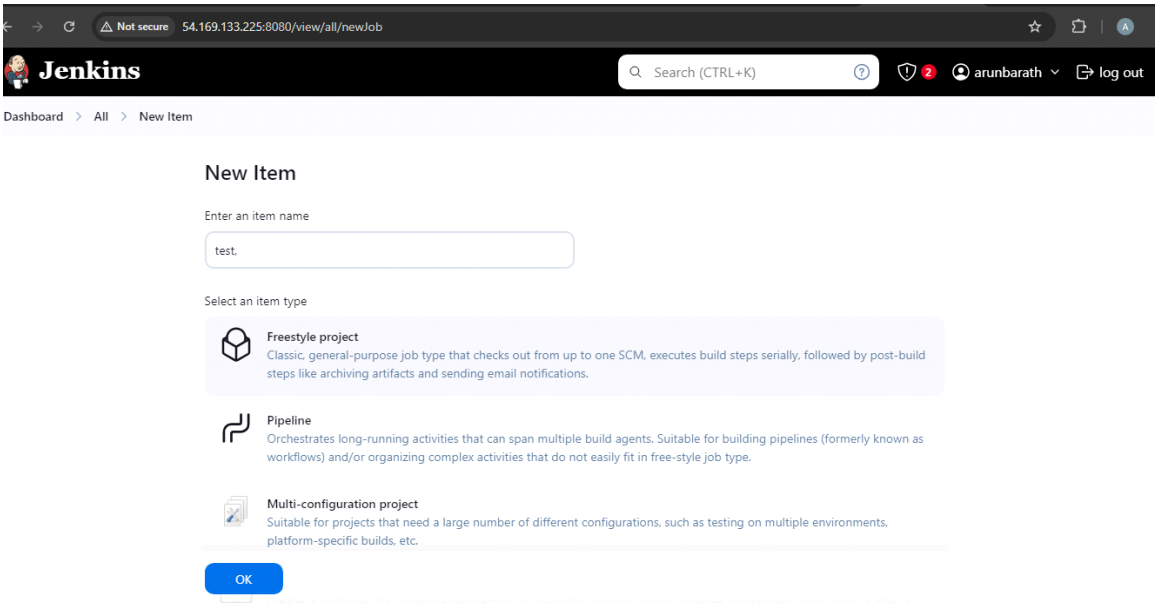


- provide the private key

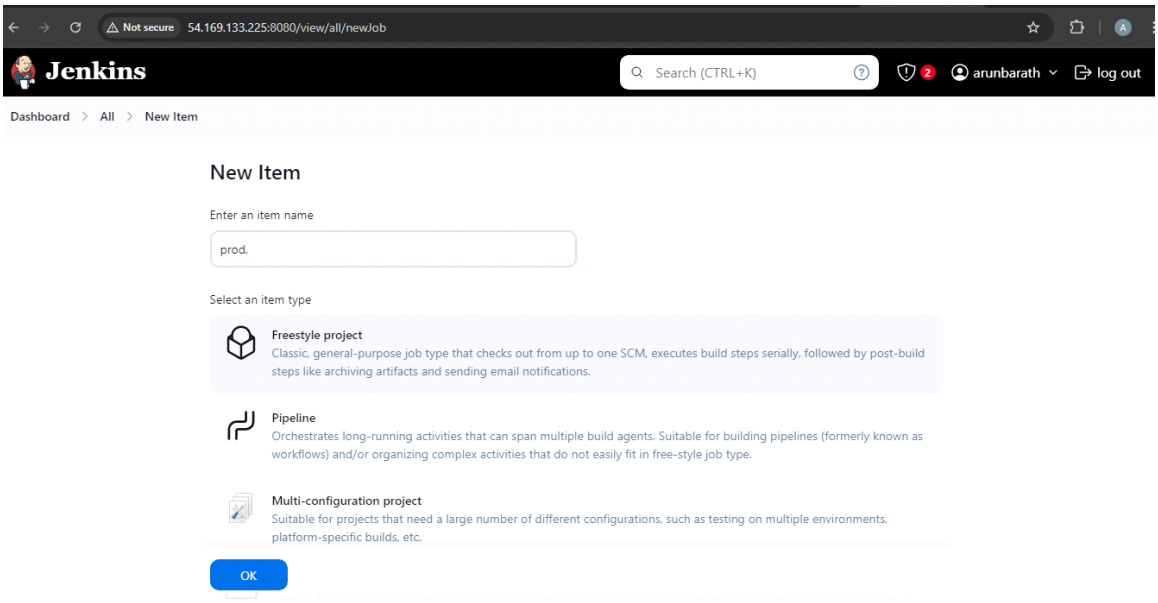


- selecte the credentials which we created and save

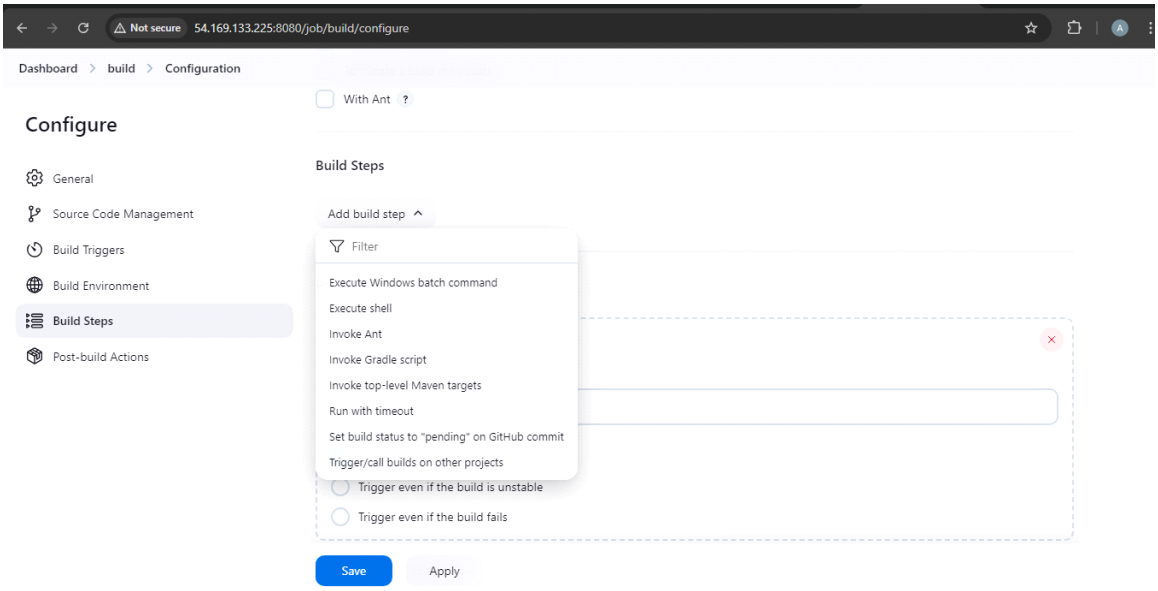




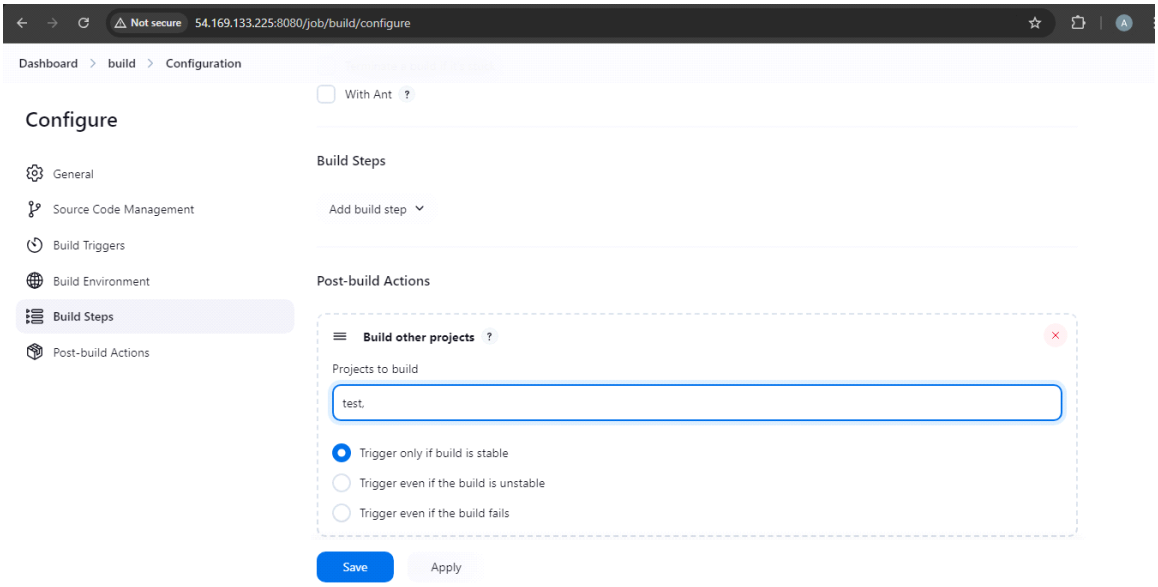
- create a test job



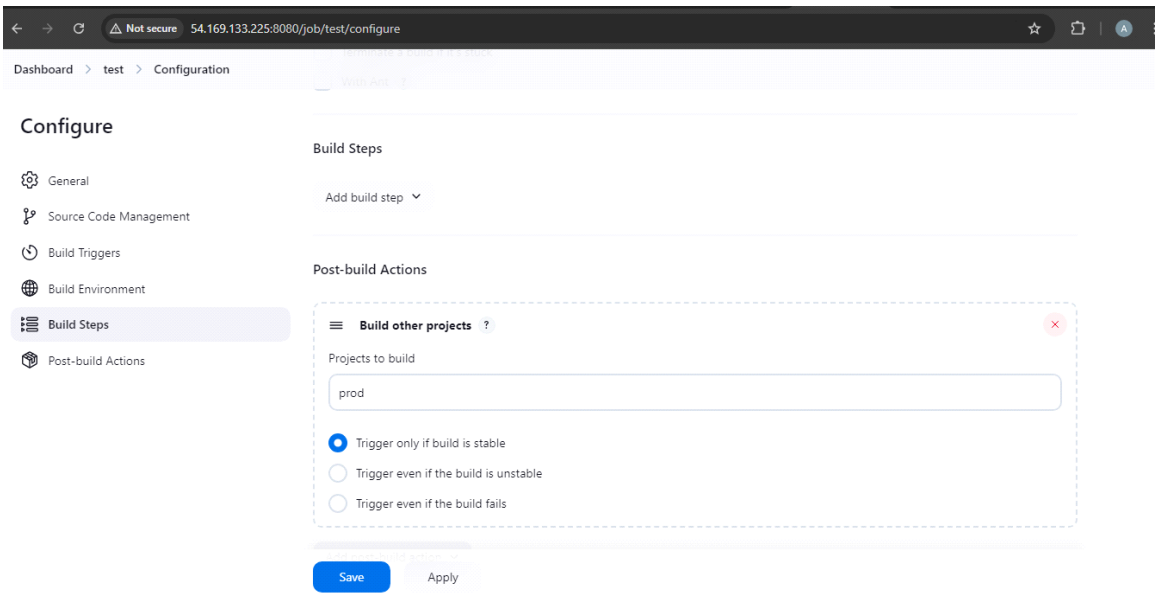
- created a prod job



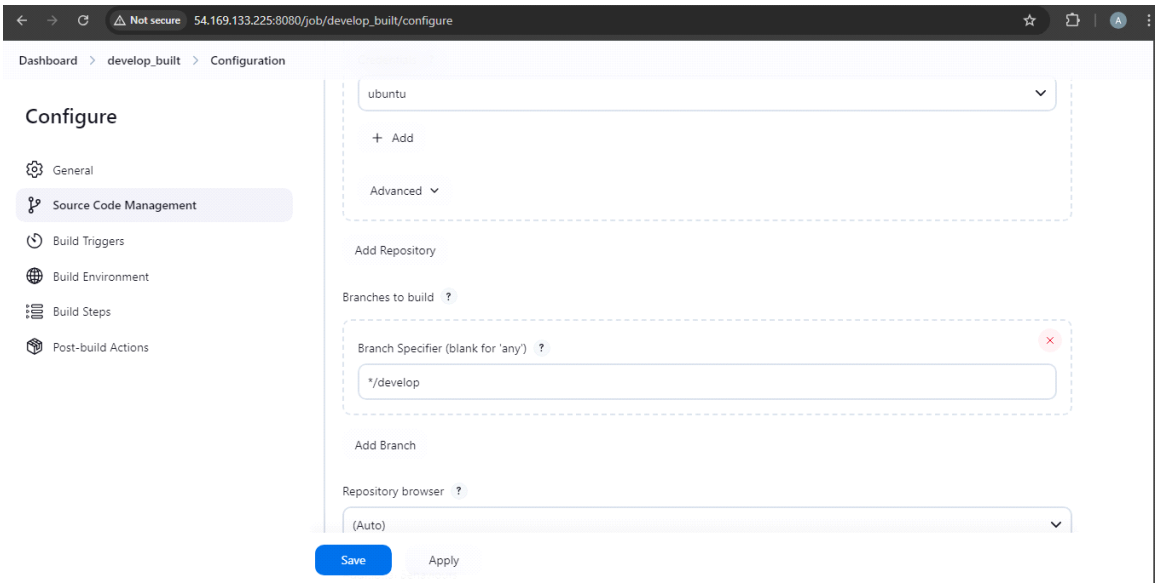
- under built environment select the built other project

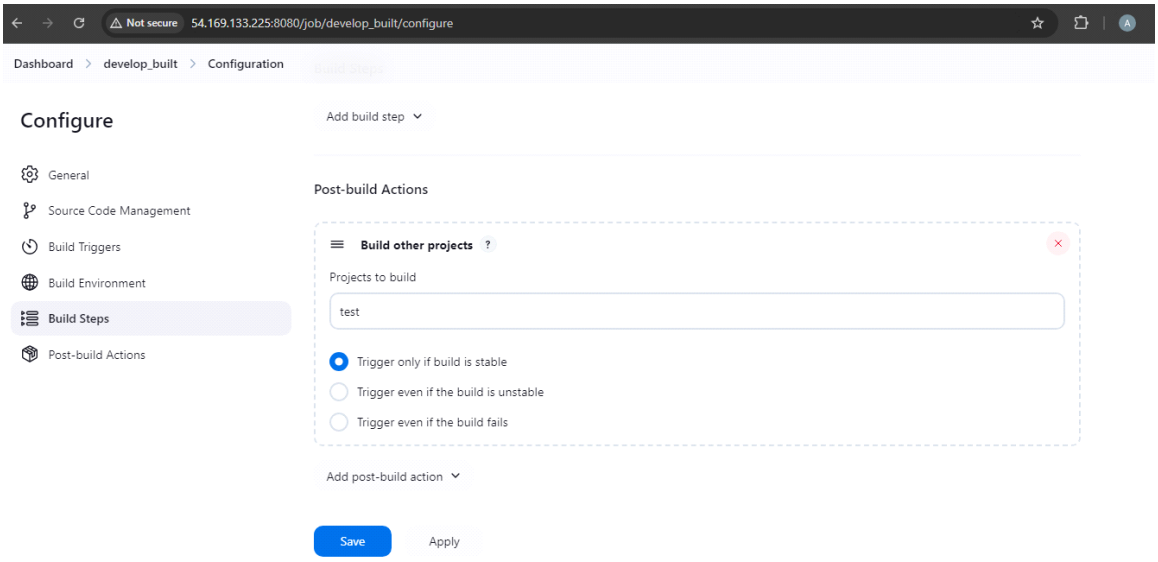


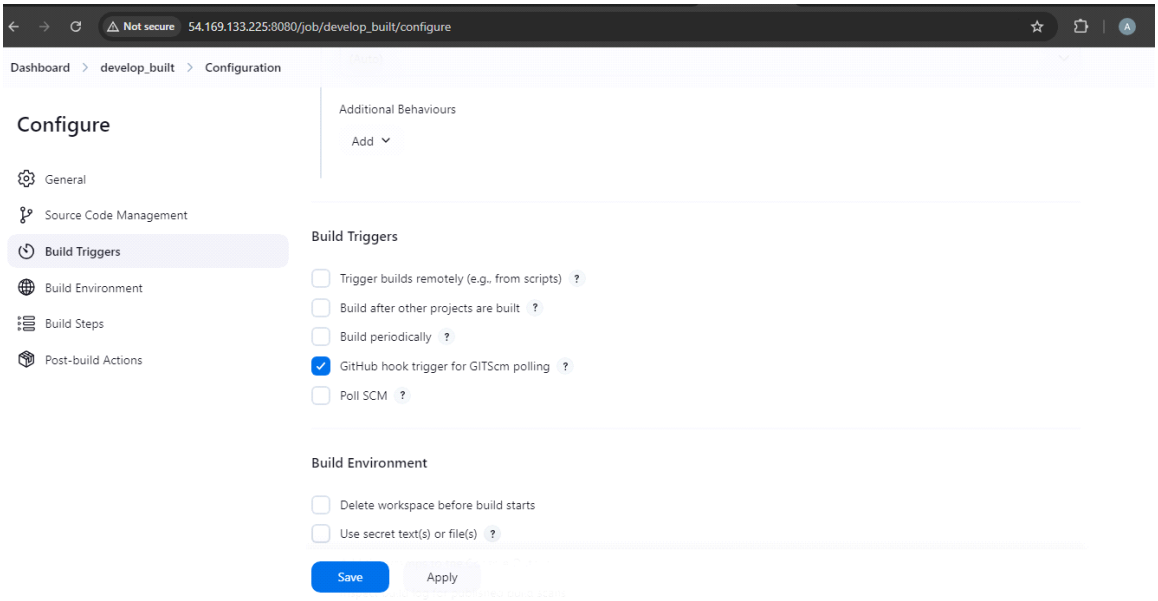
- under built steps select the trigger only if the built is stable
- click on save



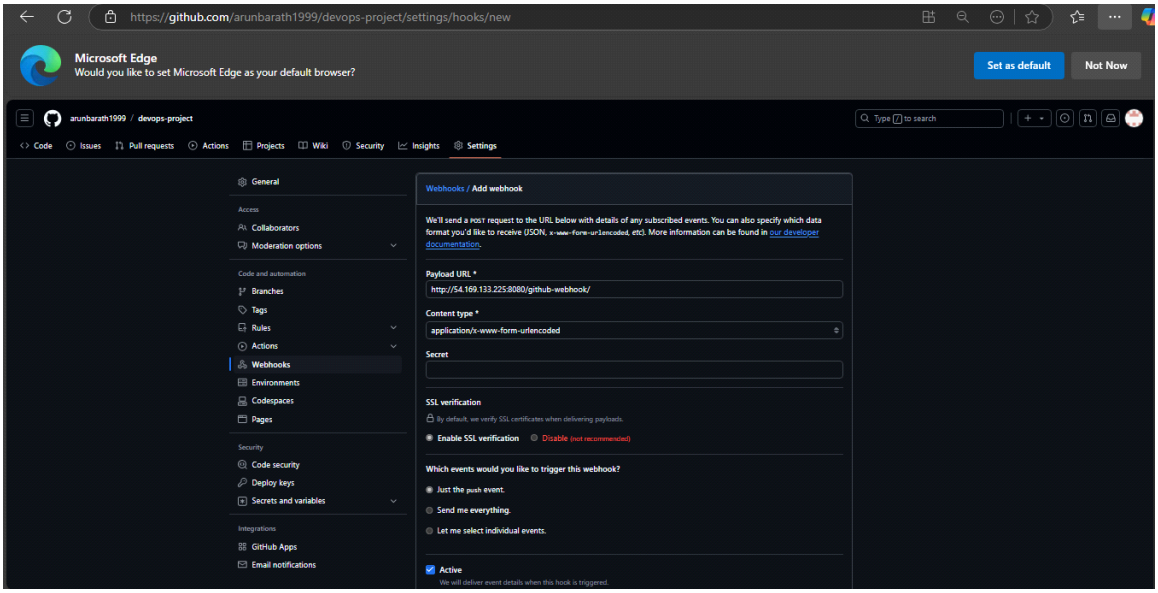
- under built steps select the trigger only if the built is stable
- click on save


Not secure
54.169.133.225:8080/job/develop_built/configure
Dashboard
develop_built
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
ubuntu
Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/develop
Add Branch
Repository browser
(Auto)
Save
Apply

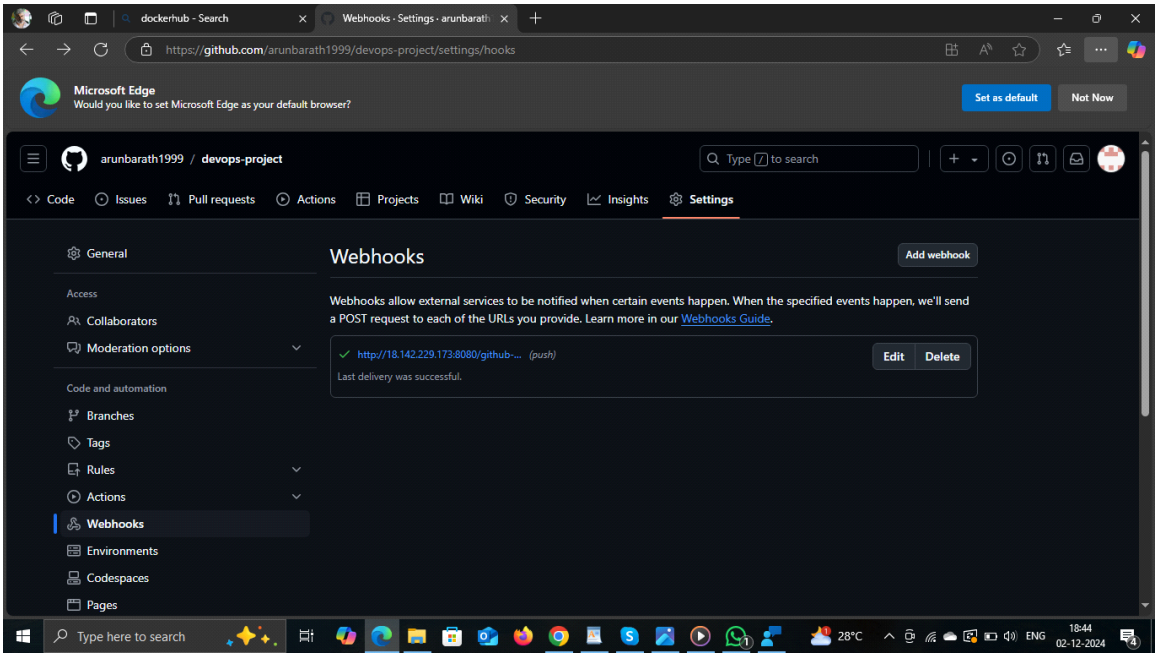

Not secure
54.169.133.225:8080/job/develop_built/configure
Dashboard
develop_built
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Add build step
Post-build Actions
Build other projects
Projects to build
test
Trigger only if build is stable
Trigger even if the build is unstable
Trigger even if the build fails
Add post-build action
Save
Apply



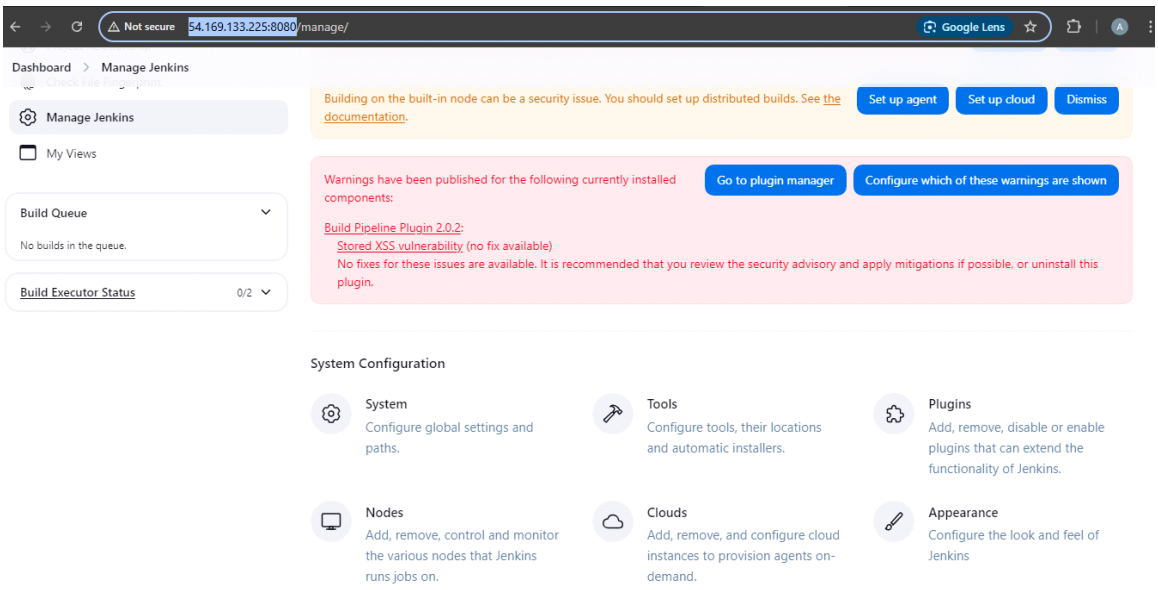
- for the automatic getting building the job when the file is pushed to Github branch



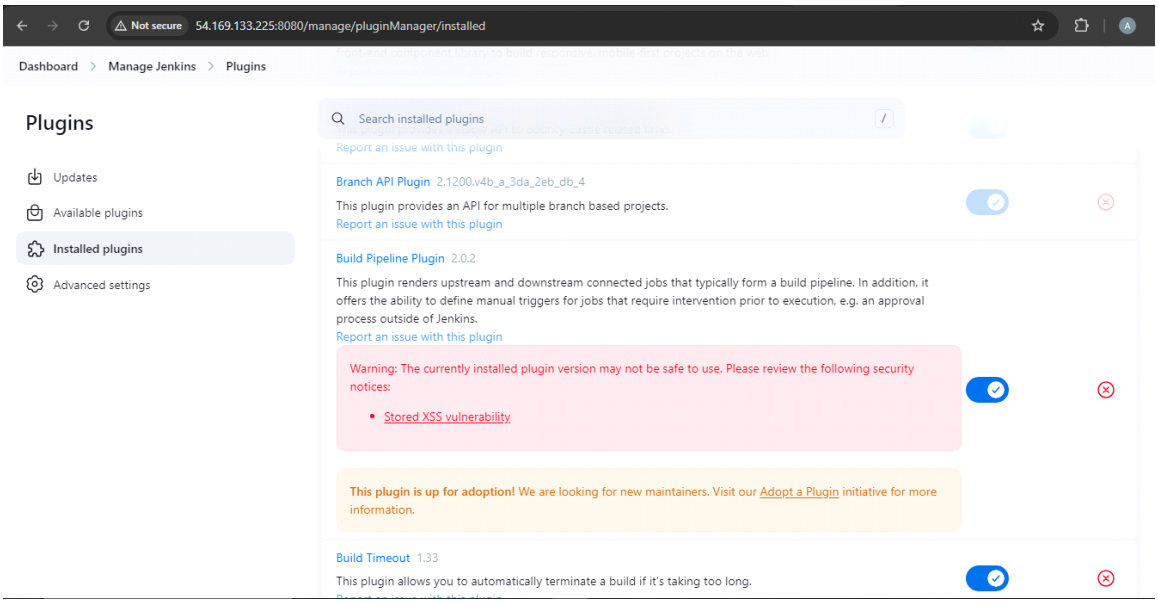
- in payload URL give the jenkins ip address /github-webhook/
- and click on save



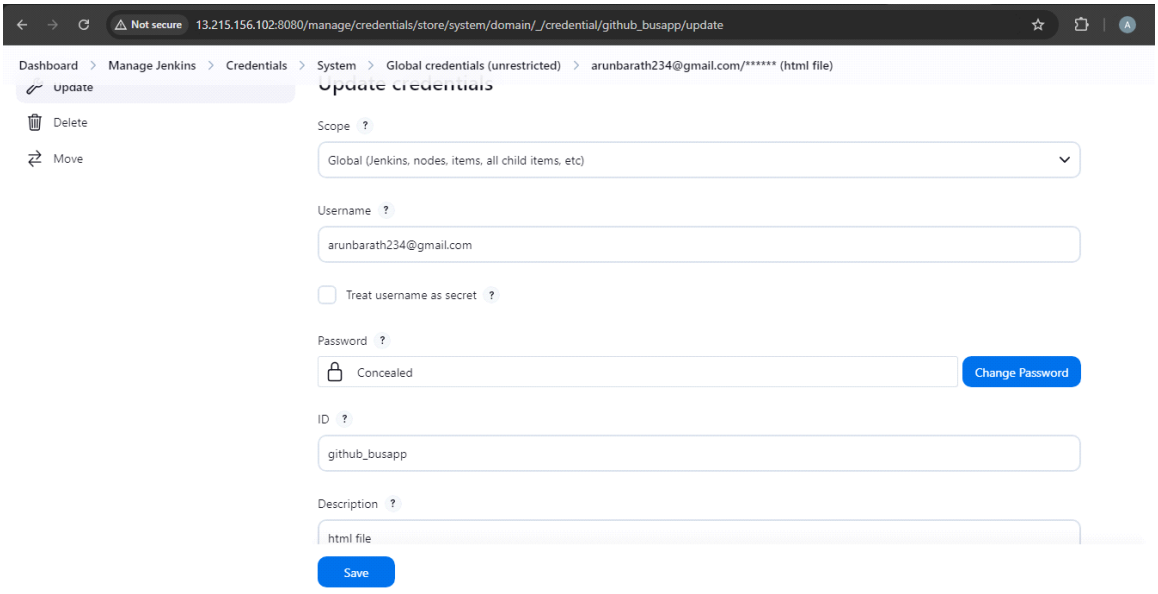
- we can see the right mark so the webhook was configured properly



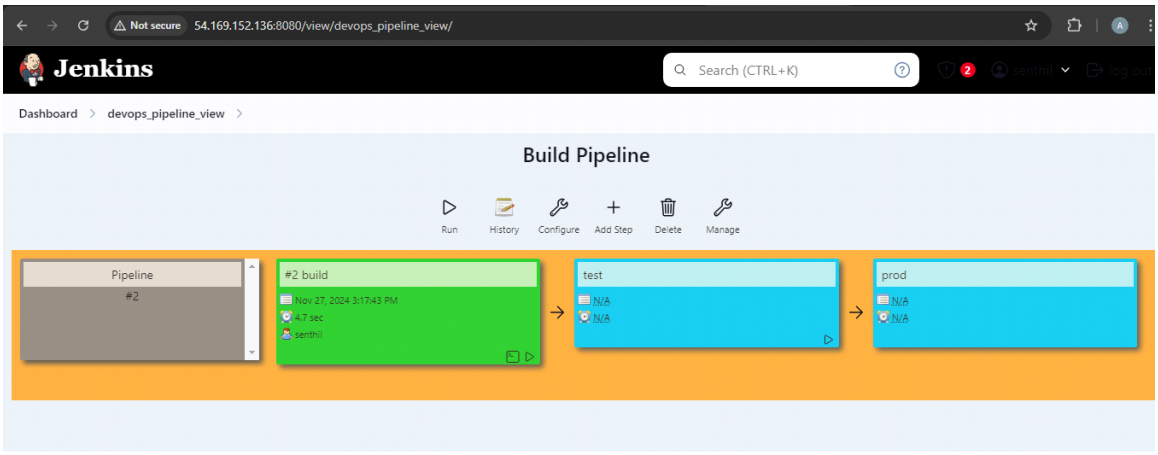
- under the manage jenkins --> plugins



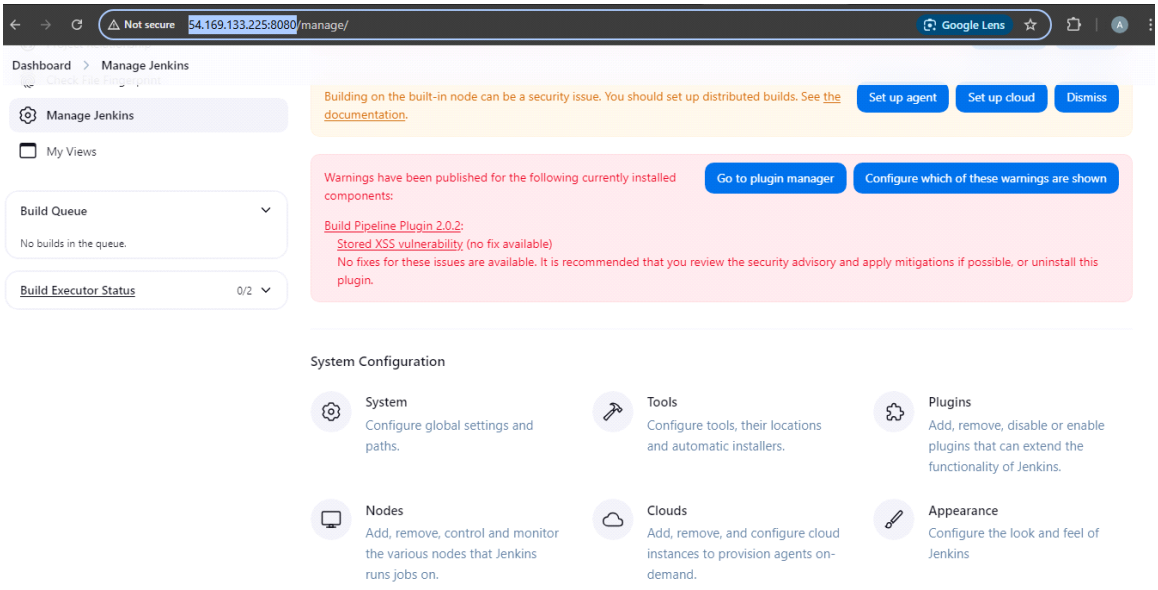
- install the plugin in --> build pipeline



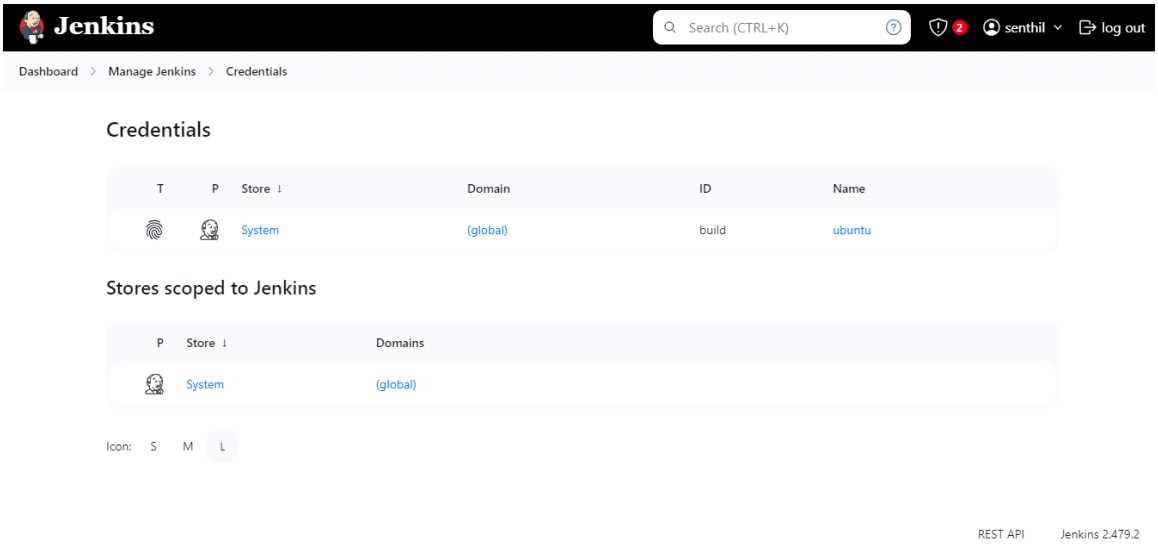
- using this built view we can see the builts in graphical representation



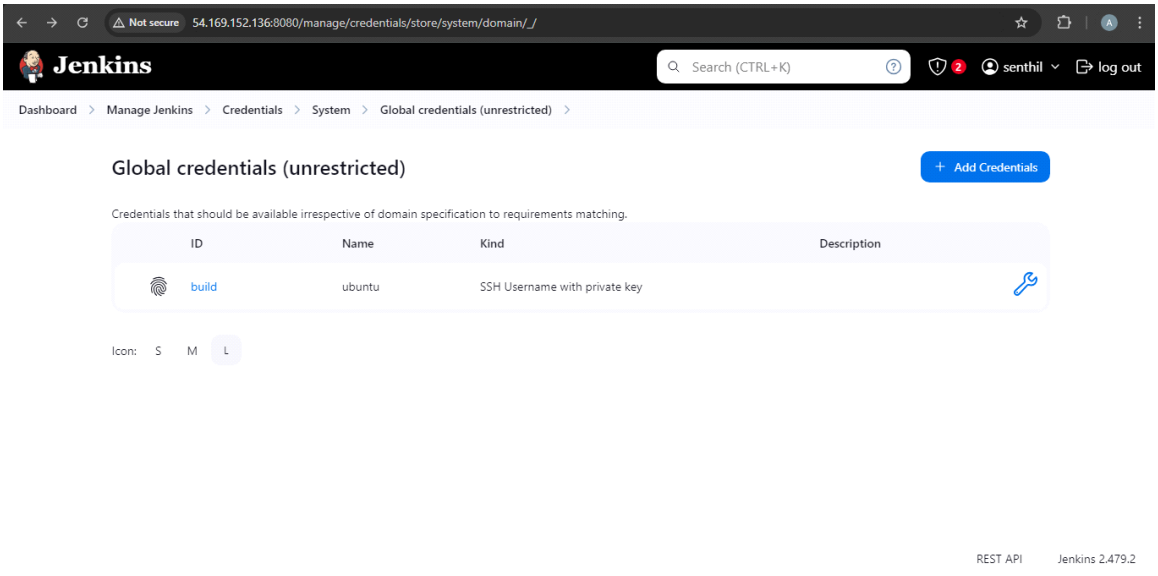
- When the file is pushed to git hub the builts are automatically getting triggered



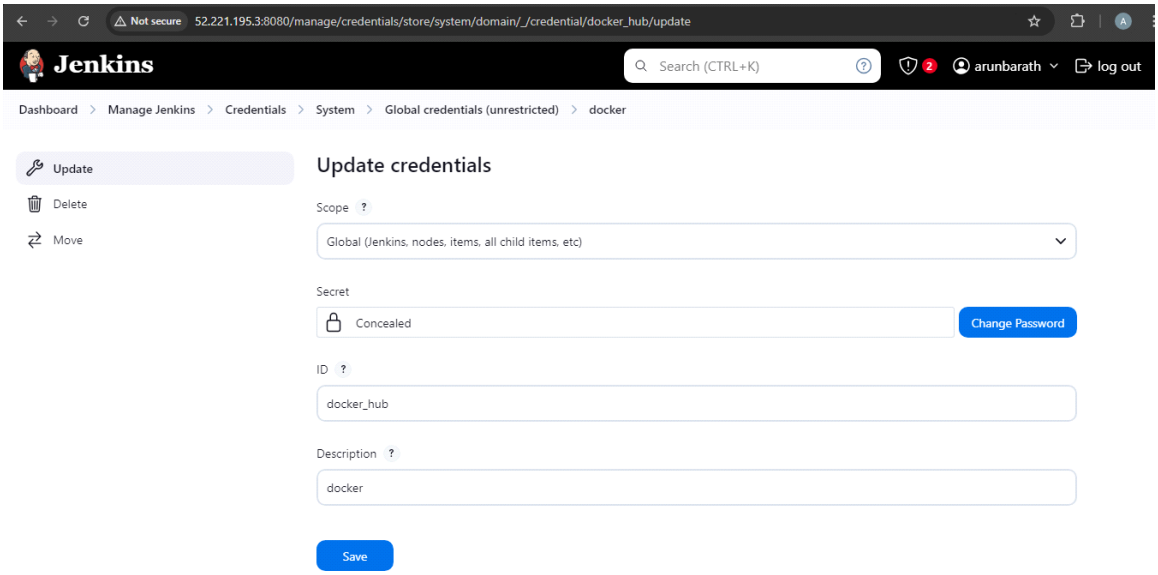
- under mange jenkins --> credentials



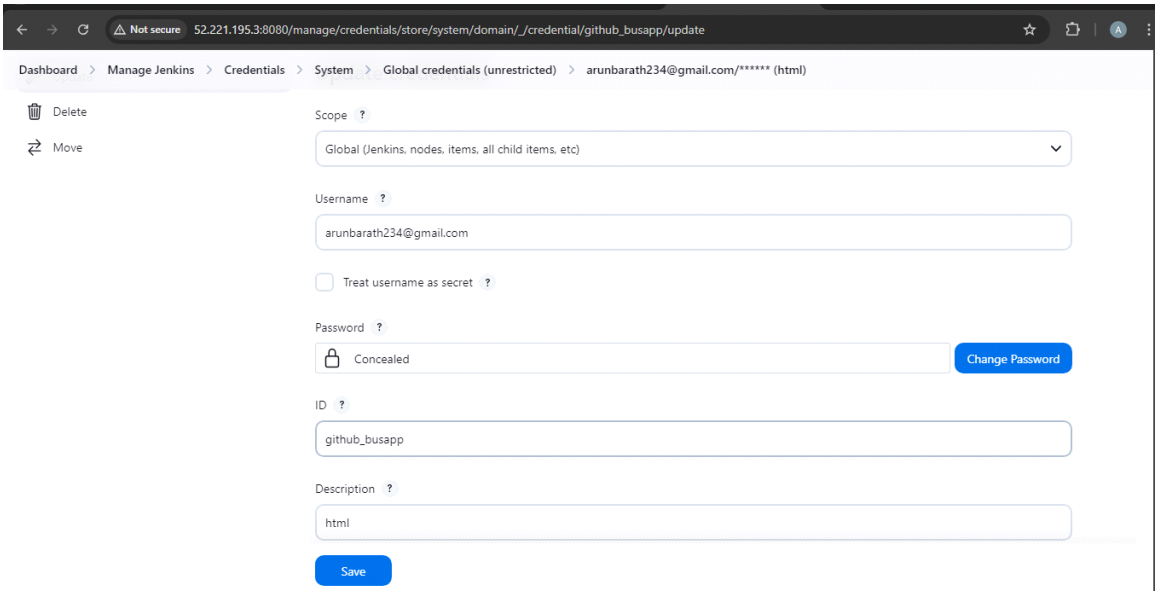
- Click on the global



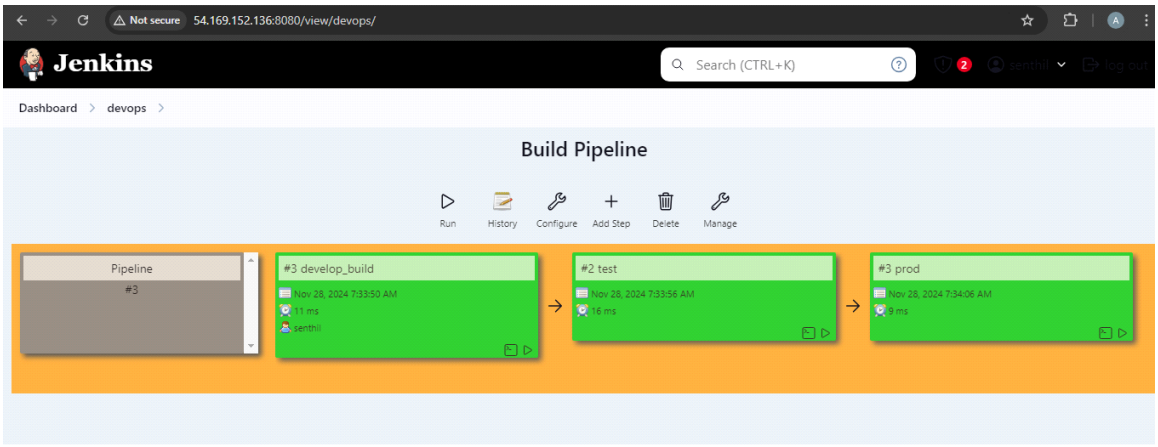
- Add credentials for creating the docker credential



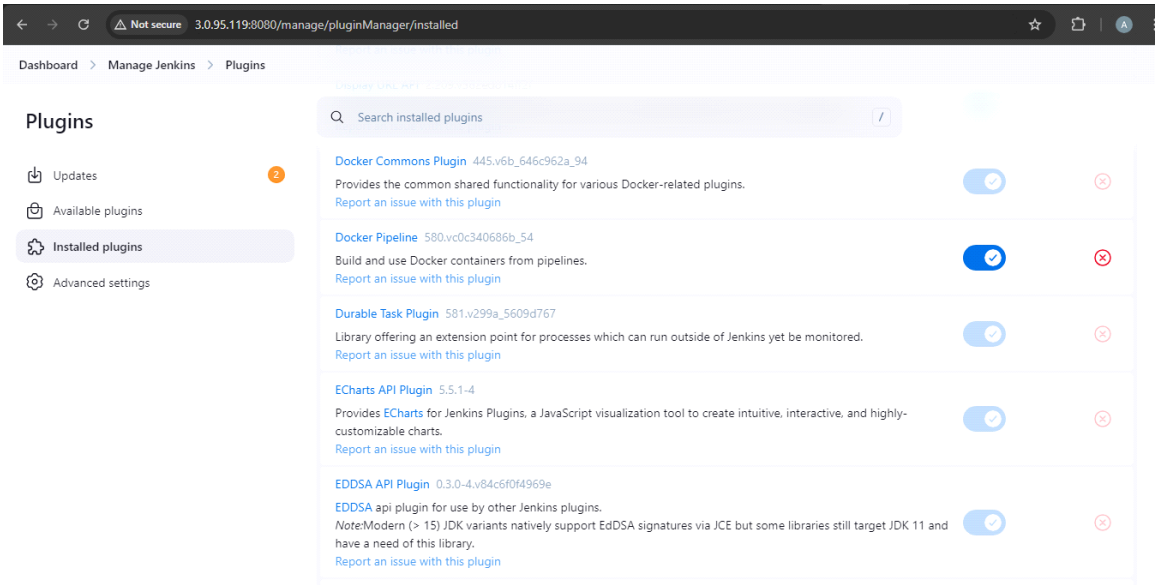
- created the docker credentials



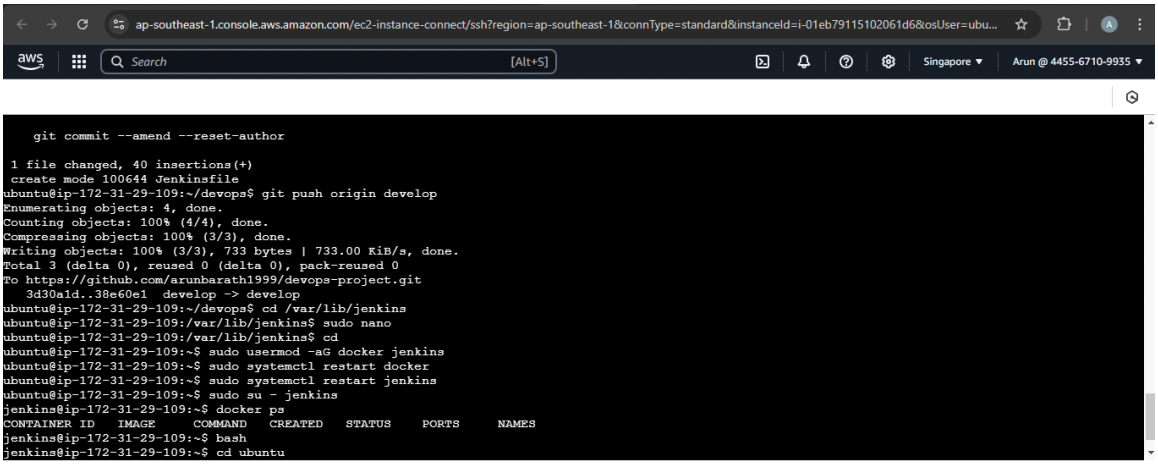
- created the github credentials

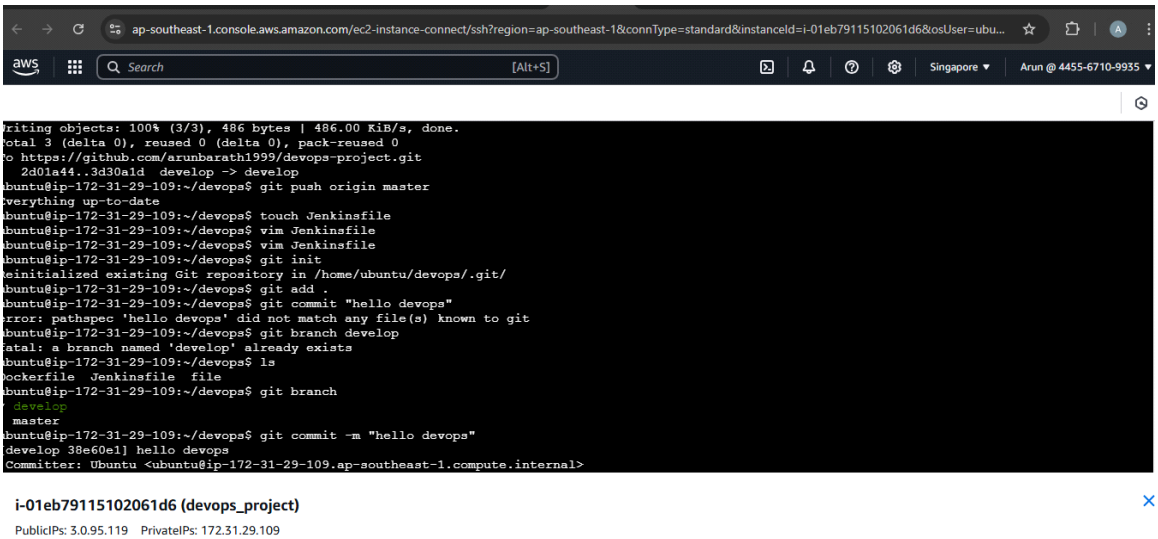


- we can observe the jobs are getting built automatically when we push the commit to github

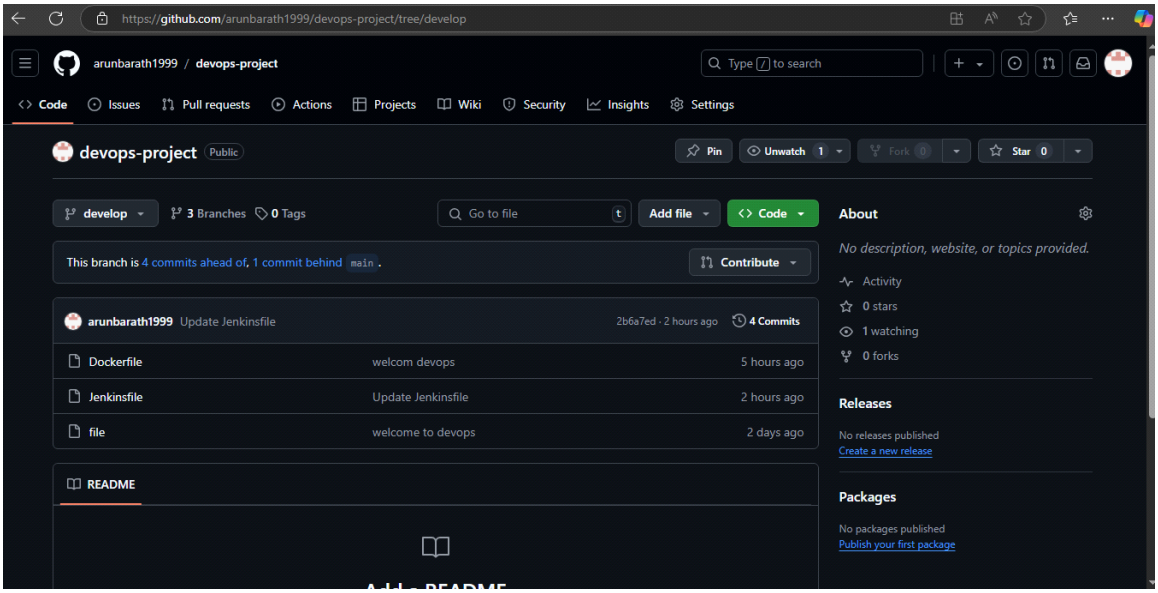


- Installed plugins docker pipeline in jenkins

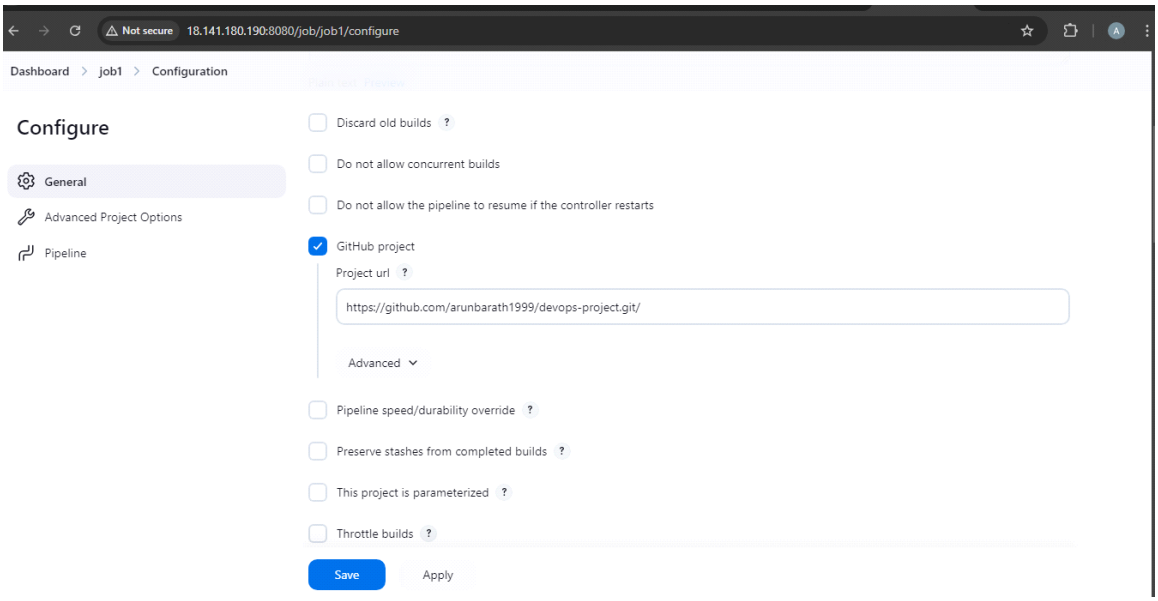




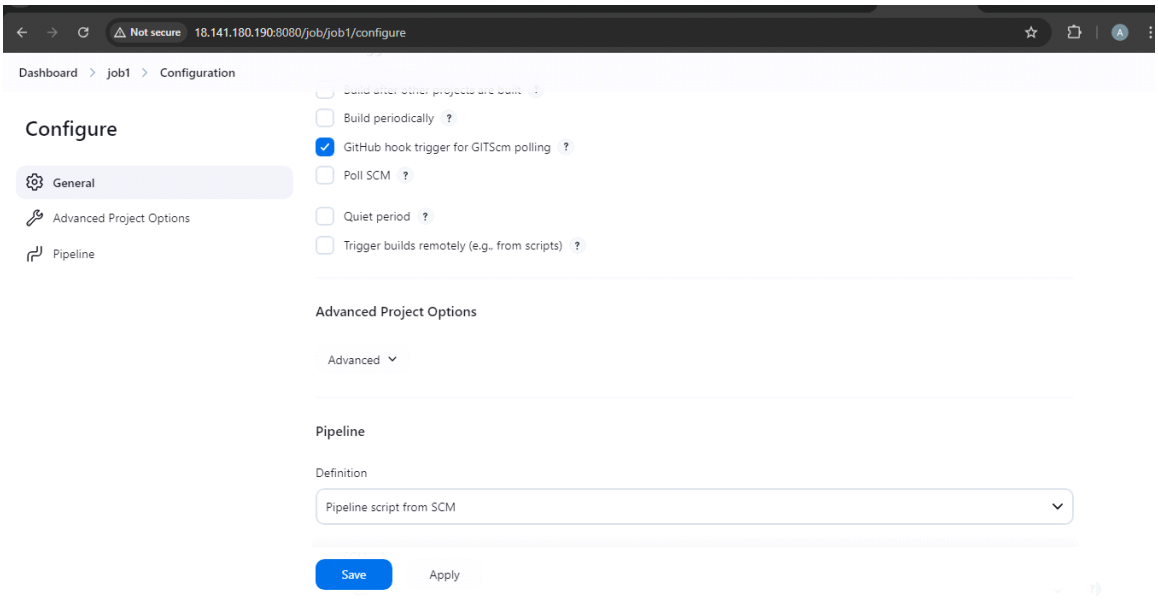
- push to the Dockerfile and Jenkins in Github repository



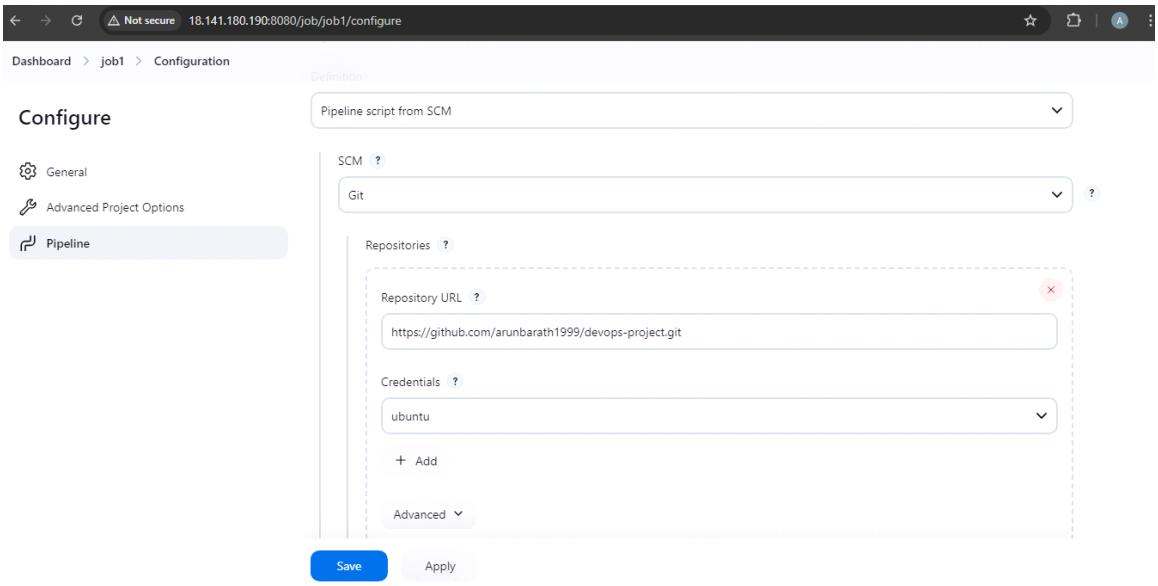
- created the Dockerfile and Jenkins in Github repository



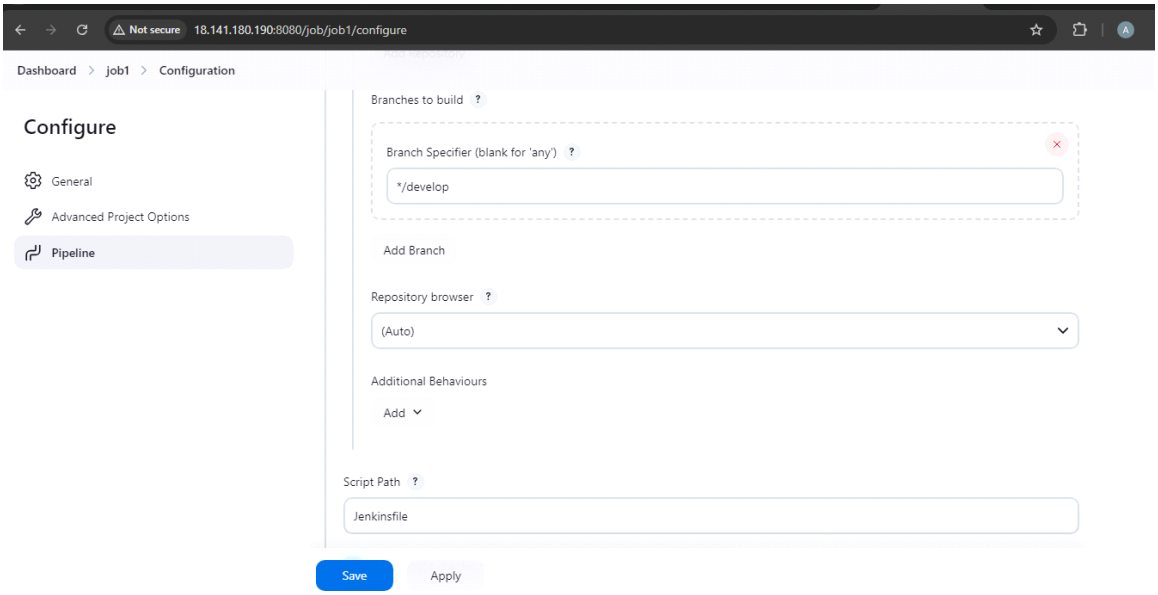
- created a pipeline job1 in jenkins
- under Github project past the Github repository URL



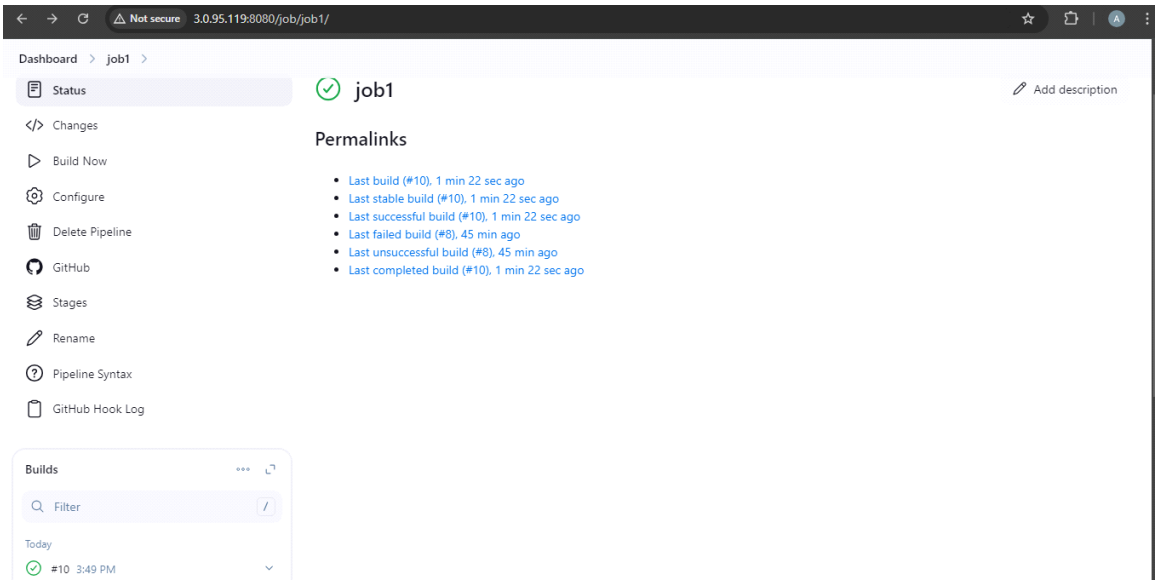
- select the Github hook trigger for Gitscm polling

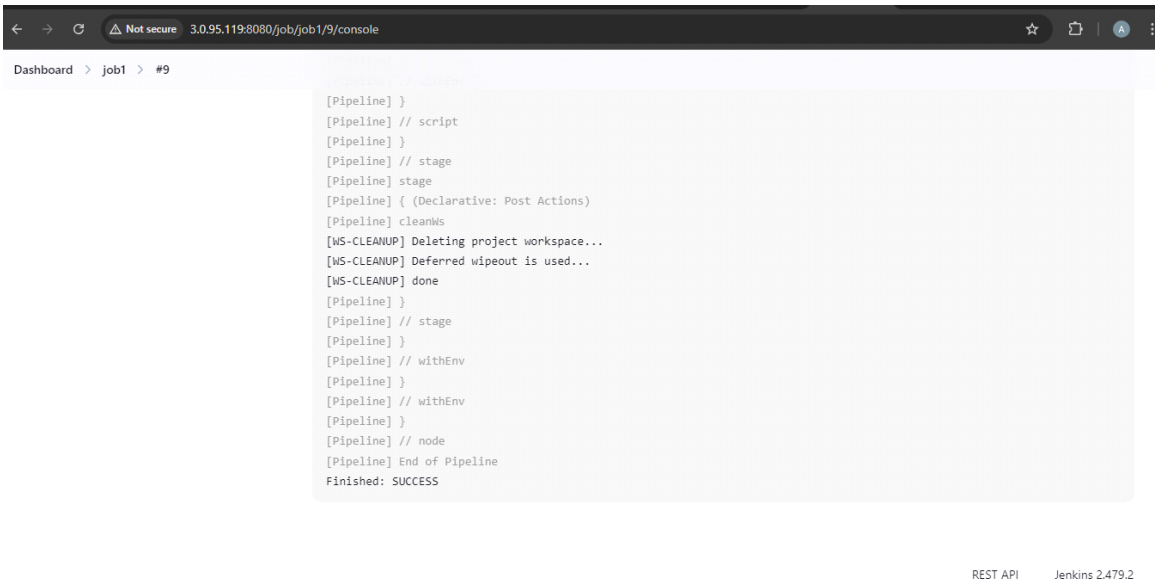


- under pipeline select the pipeline script for SCM
- under script path give the jenkins which we created in the github repository

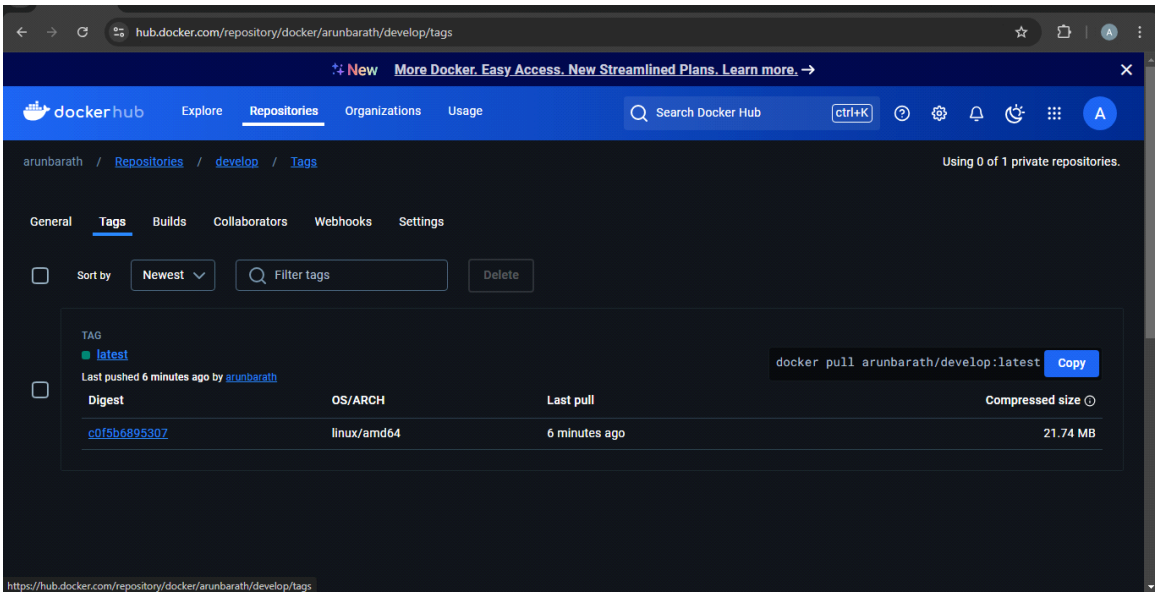


- under SCM select the git
- past the Git repository URL past the repository URL
- save the job1





- Now built the job1
- the job1 was successful built



- Finally Dockerfile was pushed to Docker hub repository