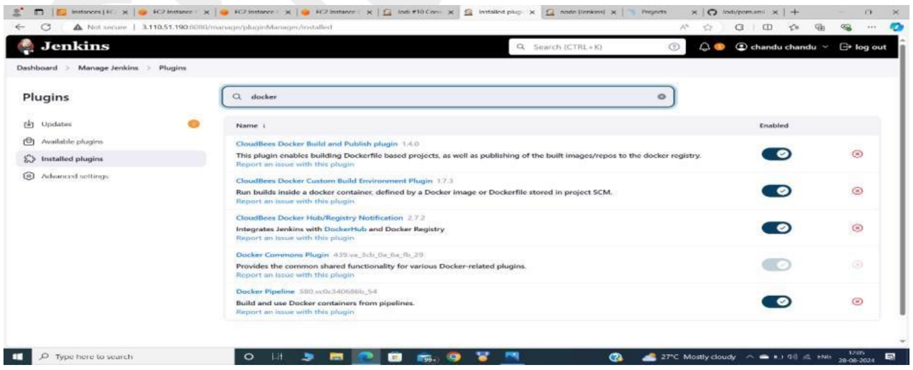
BY USING DECLARATIVE PIPELINE 1.JENKINS CAN INTEGRATE WITH DOCKER 2. PUSH THE IMAGE TO DOCKER HUB WITH NODE SERVER 3.DEPLOY THE IMAGE ON TOMCAT

Perseqsites:

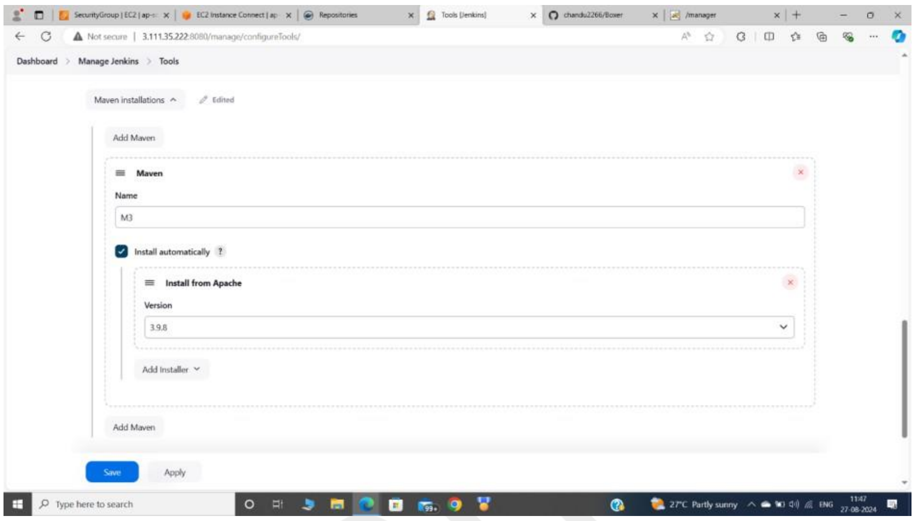
- We need a launch 2 instances.
  1. For Jenkins.
  2. For Node.
- Install plugins in Jenkins.



Name ↓	Enabled
CloudBees Docker Build and Publish plugin 1.4.0 This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry. Report an issue with this plugin	✓
CloudBees Docker Custom Build Environment Plugin 1.7.3 Run builds inside a docker container, defined by a Docker image or Dockerfile stored in project SCM. Report an issue with this plugin	✓
CloudBees Docker Hub/Registry Notification 2.7.2 Integrates Jenkins with DockerHub and Docker Registry Report an issue with this plugin	✓
Docker Commons Plugin 439.va_3cb_0a_6a_fb_29 Provides the common shared functionality for various Docker-related plugins. Report an issue with this plugin	✓
Docker Pipeline 580.vc0c340686b_54 Build and use Docker containers from pipelines. Report an issue with this plugin	✓

- First we can start Jenkins instance select a new job(eg name:boxer) and select a pipeline option.
- Now we can setup the node for the job.
- **Dashboard>manage Jenkins>nodes>**

- Create a new node and add the required details here.
- Now we integrate with docker and maven by using declarative pipeline.
- For maven first we need to set up a maven configuration in tools.


Dashboard > Manage Jenkins > Tools

Maven installations

Add Maven

Maven

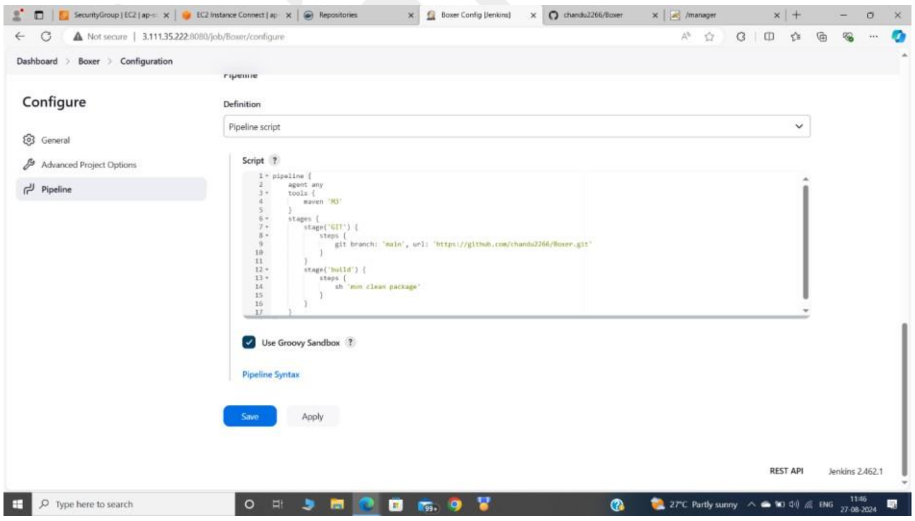
Name: M3

[✓] Install automatically

Install from Apache

Version: 3.9.8

- Now we can write a pipeline for the git maven and node.


Dashboard > Boxer > Configuration

Definition: Pipeline script

Script

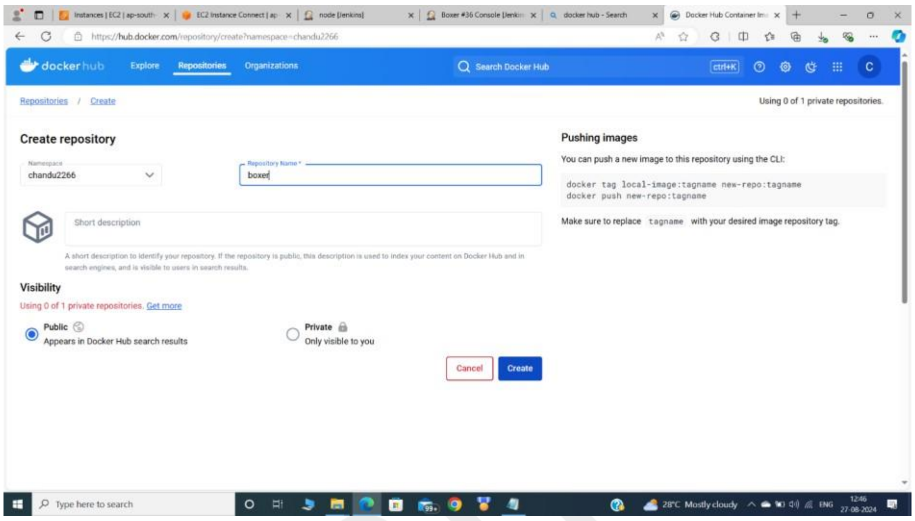
```
pipeline {
    agent any
    tools {
        maven 'M3'
    }
    stages {
        stage('GIT') {
            steps {
                git branch: 'main', url: 'https://github.com/chandu2266/Boxer.git'
            }
        }
        stage('build') {
            steps {
                sh 'mvn clean package'
            }
        }
    }
```

[✓] Use Groovy Sandbox

Note: **the full scipt pipeline below the document**.

- Click on save and apply.

- Click on build now.
- Now we can set up the docker.
- Firstly we can go to the docker hub and create the one repository.



Using 0 of 1 private repositories.

Create repository

Namespace: chandu2266

Repository Name *: boxer

Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 0 of 1 private repositories. Get more

[✓ Public] Appears in Docker Hub search results

Private Only visible to you

Cancel Create

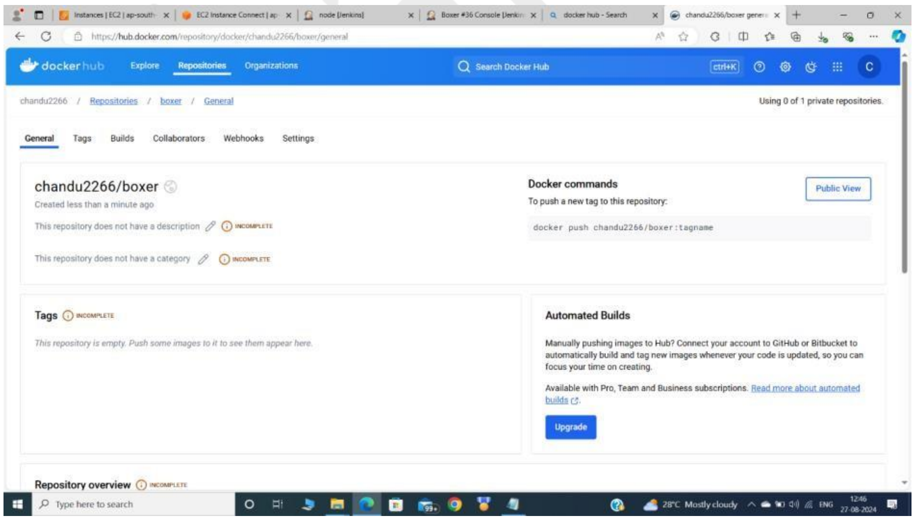
Pushing images

You can push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to replace `tagname` with your desired image repository tag.

- After creating a repository now click on the our profile.



Using 0 of 1 private repositories.

chandu2266/boxer

Created less than a minute ago

This repository does not have a description INCOMPLETE

This repository does not have a category INCOMPLETE

Docker commands

To push a new tag to this repository:

```
docker push chandu2266/boxer:tagname
```

Tags INCOMPLETE

This repository is empty. Push some images to it to see them appear here.

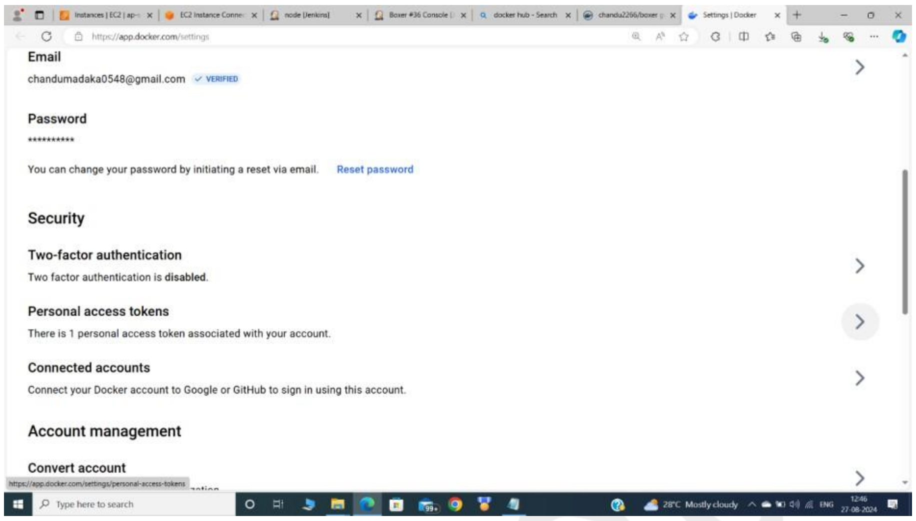
Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

Repository overview INCOMPLETE

- Next click on the personal access tokens and next click on the generate new token.


https://app.docker.com/settings

Email

chandumadaka0548@gmail.com ✓ VERIFIED

Password

****

You can change your password by initiating a reset via email. Reset password

Security

Two-factor authentication

Two factor authentication is **disabled**.

Personal access tokens

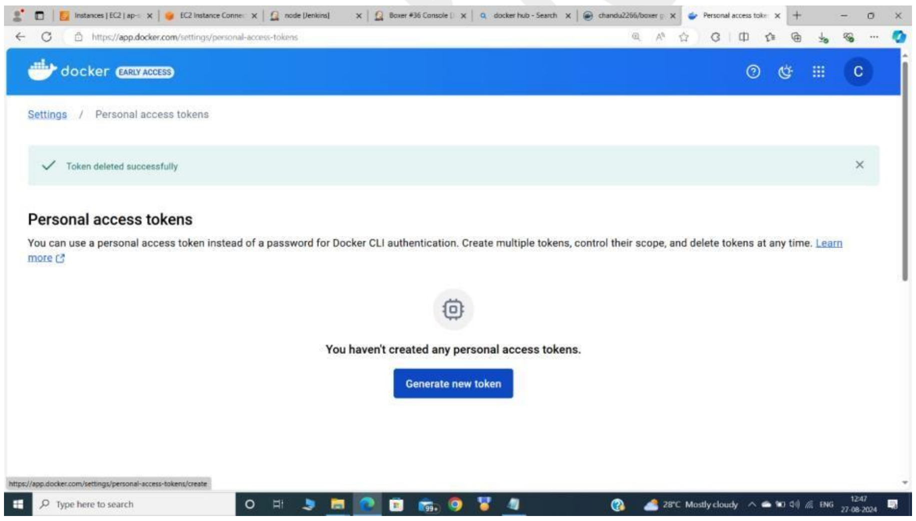
There is 1 personal access token associated with your account.

Connected accounts

Connect your Docker account to Google or GitHub to sign in using this account.

Account management

Convert account


https://app.docker.com/settings/personal-access-tokens

Settings / Personal access tokens

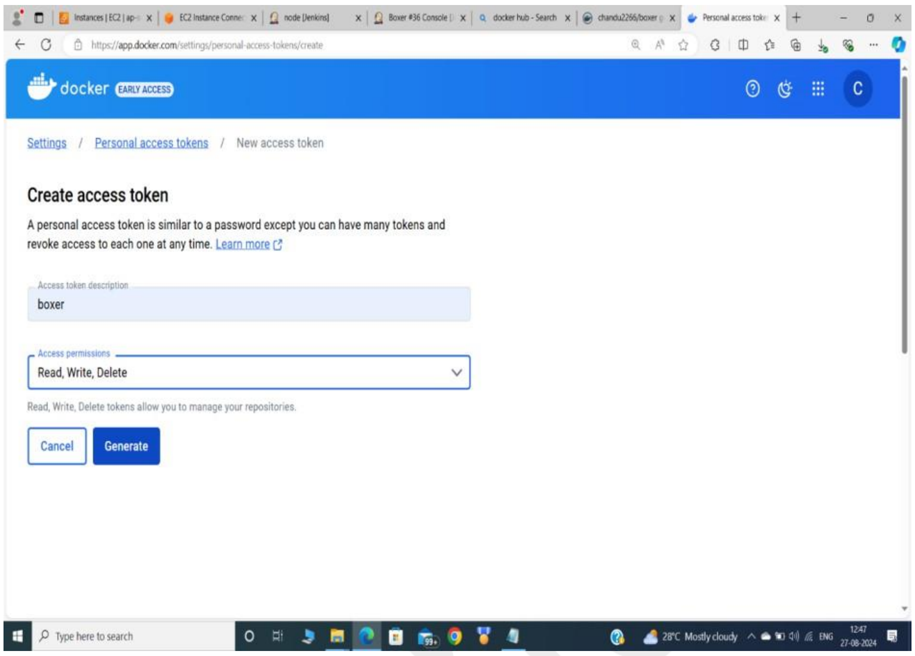
✓ Token deleted successfully

Personal access tokens

You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. Learn more

You haven't created any personal access tokens.

- Select the access permission **read,write,delete**.
- And click on the generate.



Settings / Personal access tokens / New access token

Create access token

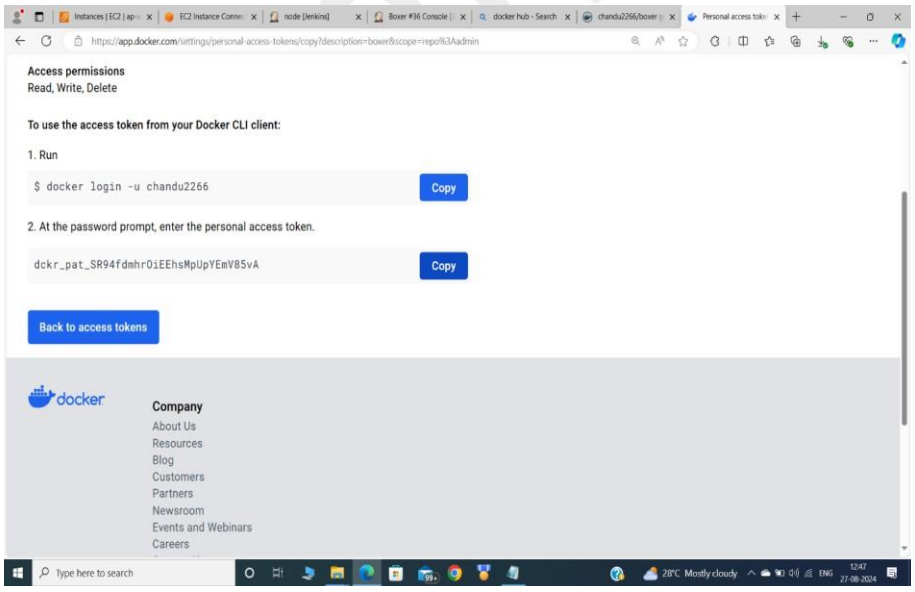
A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. Learn more

Access token description: boxer

Access permissions: Read, Write, Delete

Read, Write, Delete tokens allow you to manage your repositories.

➢ The tocken will generate as shown in the below.



Access permissions
Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run

```
$ docker login -u chandu2266
```

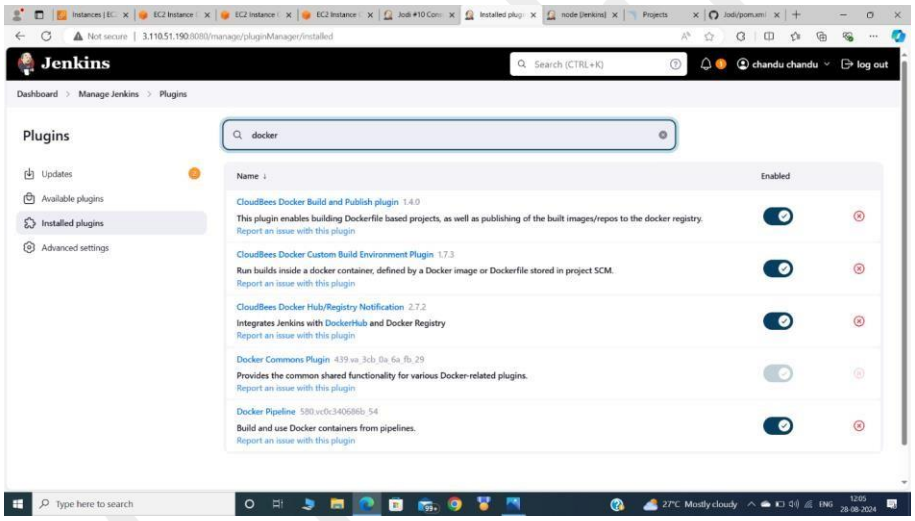
2. At the password prompt, enter the personal access token.

```
dckr_pat_SR94fdmhrOiEEhsMpUpYEmV85vA
```

➢ Now copy the token for the add as a credentials in Jenkins.

➢ Now come to the Jenkins server go to the **dashboard>manage Jenkins>credentials**

- **Click on global and click add credentials.**
- **Kind is username and password.**
- **Username: chandu2266 (dockerhub username).**
- **Password: paste the token.**
- Give id and description(dockerhub-token).
- Now install the plugins related to docker hub.
- **Dashboard>manage Jenkins>plugins**.
- Go to the available plugins and install the plugins as shown in the below fig.

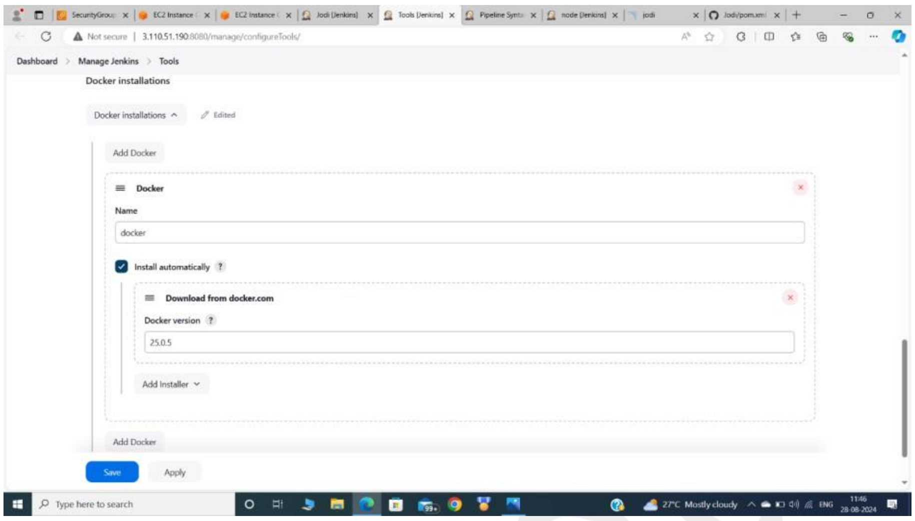

Dashboard > Manage Jenkins > Plugins

Plugins

Search: docker

Name ↓	Enabled
CloudBees Docker Build and Publish plugin 1.4.0 This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry. Report an issue with this plugin	✓
CloudBees Docker Custom Build Environment Plugin 1.7.3 Run builds inside a docker container, defined by a Docker image or Dockerfile stored in project SCM. Report an issue with this plugin	✓
CloudBees Docker Hub/Registry Notification 2.7.2 Integrates Jenkins with DockerHub and Docker Registry Report an issue with this plugin	✓
Docker Commons Plugin 439.va_3cb_0a_6a_fb_29 Provides the common shared functionality for various Docker-related plugins. Report an issue with this plugin	✓
Docker Pipeline 580.vc0c340686b_54 Build and use Docker containers from pipelines. Report an issue with this plugin	✓

- For plugins activation restart the Jenkins.
- Now go to the **dashboard>manage Jenkins>system**.
- Here we have a one option docker open it clik on add installer enter details as shown in the below.



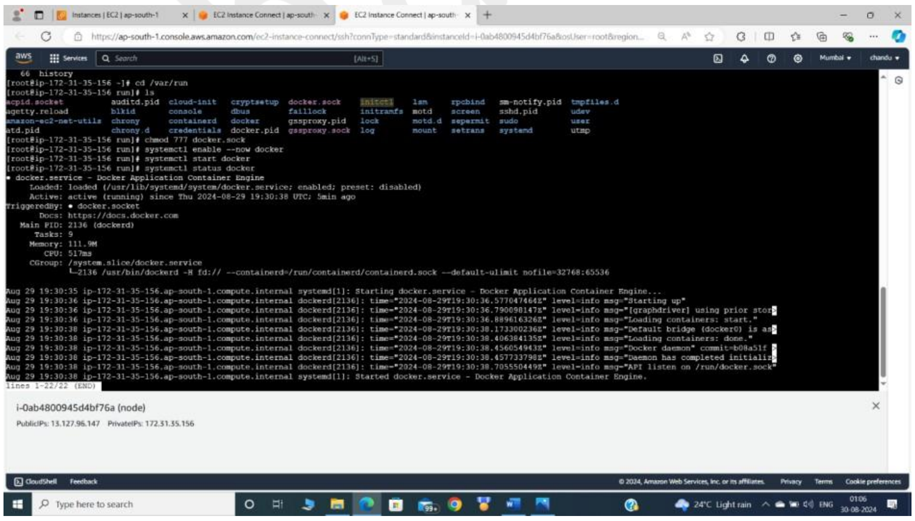
Name: docker

[✓] Install automatically

Download from docker.com

Docker version: 25.0.5

- Save and apply.
- Now go the node sever run the commands.
- **Yum install git maven docker -y**
- **cd /var/run/**
- **chmod 777 docker.sock**
- **systemctl enable --now docker**
- **systemctl start docker**
- **systemctl status docker**



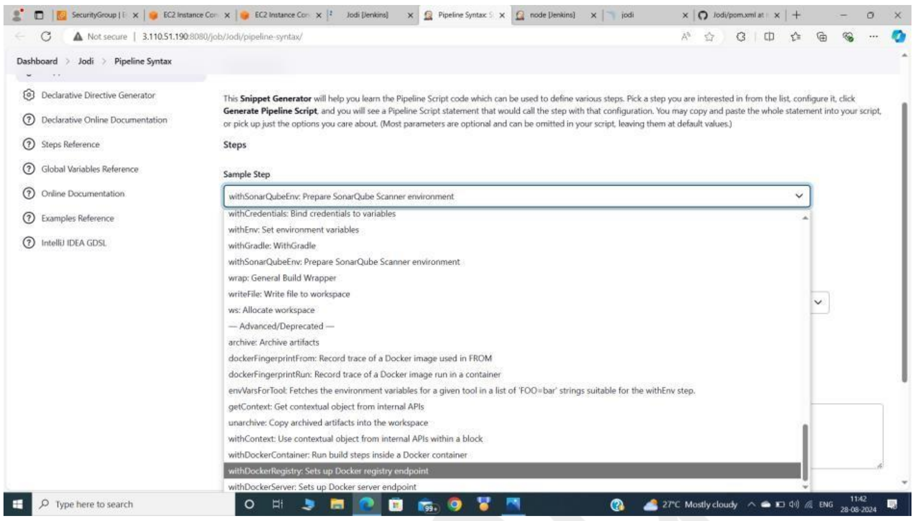
```
  66  history
[root@ip-172-31-35-156 ~]# cd /var/run
[root@ip-172-31-35-156 run]# ls
acpid.socket          auditd.pid  cloud-init   cryptsetup  docker.sock    initctl    lsm     rpcbind   sm-notify.pid  tmpfiles.d
agetty.reload         blkid       console      dbus        faillock       initramfs  motd    screen    sshd.pid       udev
amazon-ec2-net-utils  chrony      containerd   docker      gssproxy.pid   lock       motd.d  sepermit  sudo           user
atd.pid               chrony.d    credentials  docker.pid  gssproxy.sock  log        mount   setrans   systemd        utmp
[root@ip-172-31-35-156 run]# chmod 777 docker.sock
[root@ip-172-31-35-156 run]# systemctl enable --now docker
[root@ip-172-31-35-156 run]# systemctl start docker
[root@ip-172-31-35-156 run]# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
     Active: active (running) since Thu 2024-08-29 19:30:38 UTC; 5min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2136 (dockerd)
      Tasks: 9
     Memory: 111.9M
        CPU: 517ms
     CGroup: /system.slice/docker.service
             └─2136 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Aug 29 19:30:35 ip-172-31-35-156.ap-south-1.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Aug 29 19:30:36 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:36.577047464Z" level=info msg="Starting up"
Aug 29 19:30:36 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:36.790098147Z" level=info msg="[graphdriver] using prior stor>
Aug 29 19:30:36 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:36.889616326Z" level=info msg="Loading containers: start."
Aug 29 19:30:38 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:38.173300236Z" level=info msg="Default bridge (docker0) is as>
Aug 29 19:30:38 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:38.406384135Z" level=info msg="Loading containers: done."
Aug 29 19:30:38 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:38.456054943Z" level=info msg="Docker daemon" commit=b08a51f >
Aug 29 19:30:38 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:38.457733798Z" level=info msg="Daemon has completed initializ>
Aug 29 19:30:38 ip-172-31-35-156.ap-south-1.compute.internal dockerd[2136]: time="2024-08-29T19:30:38.705550449Z" level=info msg="API listen on /run/docker.sock"
Aug 29 19:30:38 ip-172-31-35-156.ap-south-1.compute.internal systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (END)
```

i-0ab4800945d4bf76a (node)

PublicIPs: 13.127.96.147 PrivateIPs: 172.31.35.156

- Now come to the Jenkins server click on dashboard and configure the job (boxer).
- Click on pipeline syntax.

- Select a pluing as shown in the screenshot and enter a details.
- The pluin is withdockerRegestry:Sets up Docker registry endpoint.



Dashboard > Jodi > Pipeline Syntax

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, 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

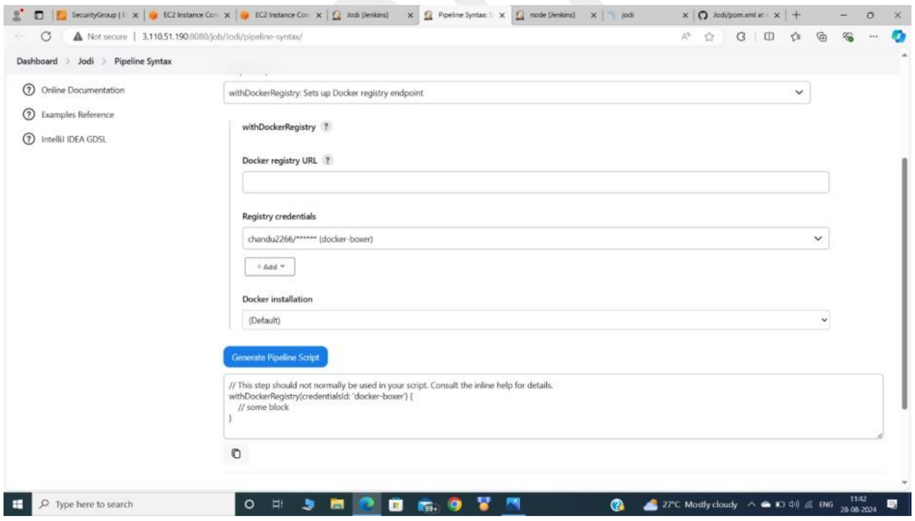
Sample Step

withSonarQubeEnv: Prepare SonarQube Scanner environment

- withCredentials: Bind credentials to variables
- withEnv: Set environment variables
- withGradle: WithGradle
- withSonarQubeEnv: Prepare SonarQube Scanner environment
- wrap: General Build Wrapper
- writeFile: Write file to workspace
- ws: Allocate workspace
- — Advanced/Deprecated —
- archive: Archive artifacts
- dockerFingerprintFrom: Record trace of a Docker image used in FROM
- dockerFingerprintRun: Record trace of a Docker image run in a container
- envVarsForTool: Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
- getContext: Get contextual object from internal APIs
- unarchive: Copy archived artifacts into the workspace
- withContext: Use contextual object from internal APIs within a block
- withDockerContainer: Run build steps inside a Docker container
- withDockerRegistry: Sets up Docker registry endpoint
- withDockerServer: Sets up Docker server endpoint

Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

- Add the details as shown in the below and click on generate script .
- Copy the script add in the new stage of the docker pipeline.



Dashboard > Jodi > Pipeline Syntax

withDockerRegistry: Sets up Docker registry endpoint

withDockerRegistry

Docker registry URL

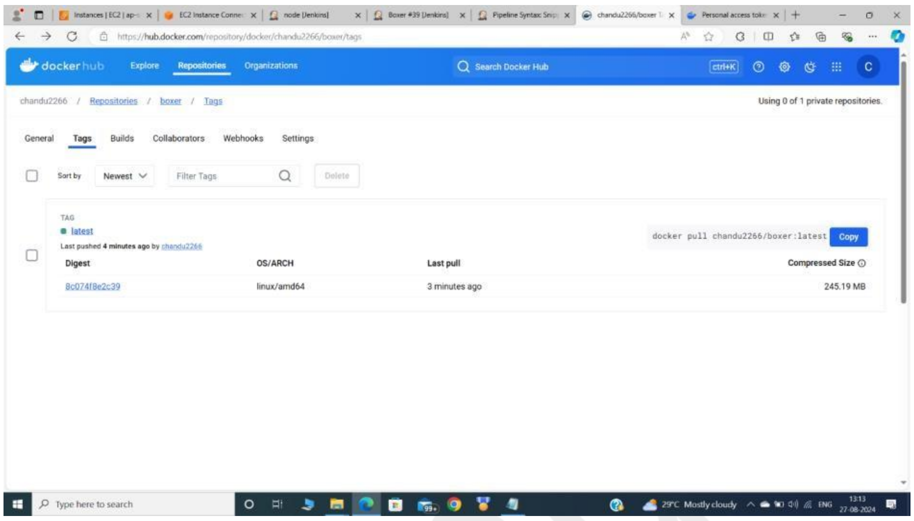
Registry credentials: chandu2266/****** (docker-boxer)

Docker installation: (Default)

```
// This step should not normally be used in your script. Consult the inline help for details.
withDockerRegistry(credentialsId: 'docker-boxer') {
    // some block
}
```

- Now we can add two sh command in script under docker stage
- **Sh ‘docker built -t chandu2266/boxer .’**
- **Sh ‘docker push chandu2266/boxer’**
- Now save and apply click on build now.
- The build will success.

- The output will be shown and we can check in docker hub the image will shown in the dockerhub.



https://hub.docker.com/repository/docker/chandu2266/boxer/tags

chandu2266 / Repositories / boxer / Tags

Using 0 of 1 private repositories.

TAG
latest
Last pushed 4 minutes ago by chandu2266

```
docker pull chandu2266/boxer:latest
```

Digest	OS/ARCH	Last pull	Compressed Size
8c074f8e2c39	linux/amd64	3 minutes ago	245.19 MB

- Now click on latest or id the image will shown.
- Now we can deploy the tomcat our image for that follow the script.
- Now come to the Jenkins and configure the job.
- Add a stage between maven and docker push. As shown in the image.

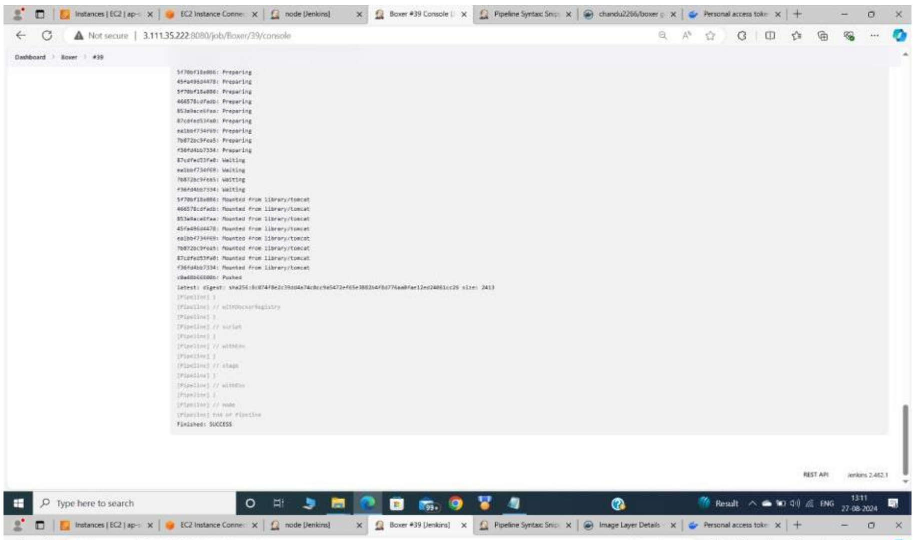
```
stage('build') {
        steps {
            sh 'mvn clean package'
        }
    }
    stage ('container') {
        steps {
            sh 'docker rm -f boxer'
            sh 'docker rmi -f chandu2266/boxer'
        }
    }
    stage('docker build and push') {
      steps {
          script {
            withDockerRegistry(credentialsId: 'docker-boxer') {
                sh "docker build -t chandu2266/boxer ."
                sh "docker push chandu2266/boxer"
            }
          }
        }
    }
```

- And also add the one more stage for tomcat deployment.

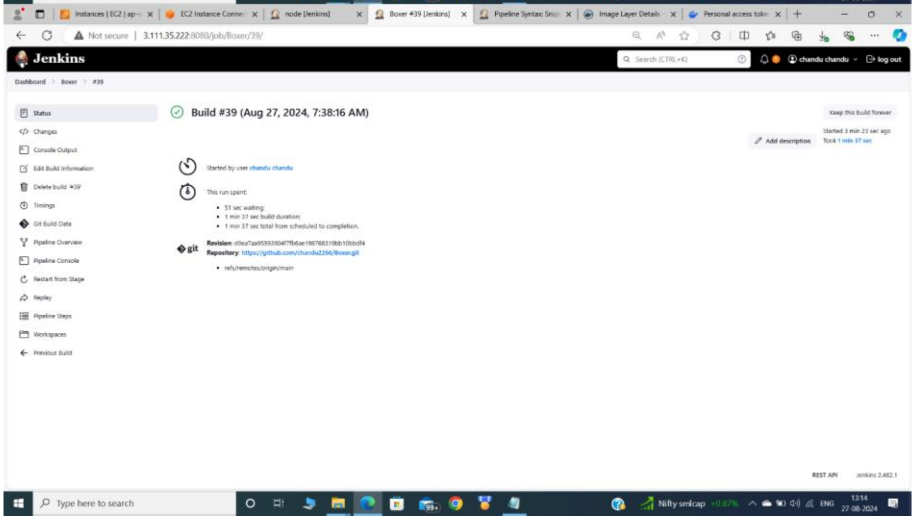
```
stage ('docker run') {
        steps {
```

```
                    sh 'docker run -d -p 8081:8080 --name boxer chandu2266/boxer'
                }
            }
        }
    }
```

- ➢ Save and apply.
- ➢ Click on build now.
- ➢ The build success as shown in the below.



```
5f70bf18a086: Preparing
45fa496d4478: Preparing
5f70bf18a086: Preparing
466578cdfadb: Preparing
853a0ace6fea: Preparing
87cdfed53fe0: Preparing
ea1bbf734f69: Preparing
7b872bc9fea5: Preparing
f36fd4bb7334: Preparing
87cdfed53fe0: Waiting
ea1bbf734f69: Waiting
7b872bc9fea5: Waiting
f36fd4bb7334: Waiting
5f70bf18a086: Mounted from library/tomcat
466578cdfadb: Mounted from library/tomcat
853a0ace6fea: Mounted from library/tomcat
45fa496d4478: Mounted from library/tomcat
ea1bbf734f69: Mounted from library/tomcat
7b872bc9fea5: Mounted from library/tomcat
87cdfed53fe0: Mounted from library/tomcat
f36fd4bb7334: Mounted from library/tomcat
c0a48b66800b: Pushed
latest: digest: sha256:[illegible] size: 2413
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Build #39 (Aug 27, 2024, 7:38:16 AM)

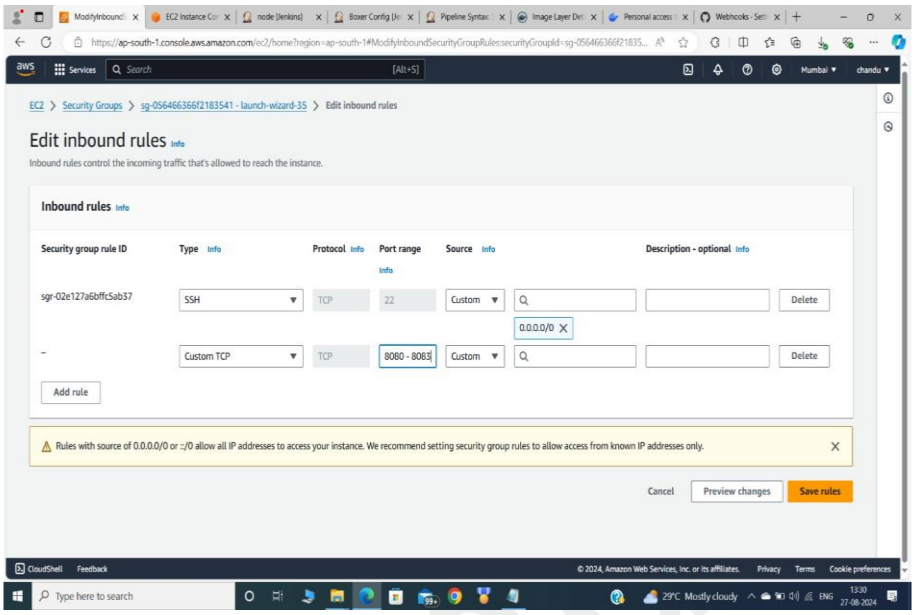
Started by user chandu chandu

This run spent:
- 51 sec waiting;
- 1 min 37 sec build duration;
- 1 min 37 sec total from scheduled to completion.

Revision: d9ea7aa95593904f7fb6ae198768319bb10bbdf4
Repository: https://github.com/chandu2266/Boxer.git
- refs/remotes/origin/main

Started 3 min 23 sec ago
Took 1 min 37 sec

- ➢ Now go the node server sequritygroups add the portnumbers.



EC2 > Security Groups > sg-056466366f2183541 - launch-wizard-35 > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

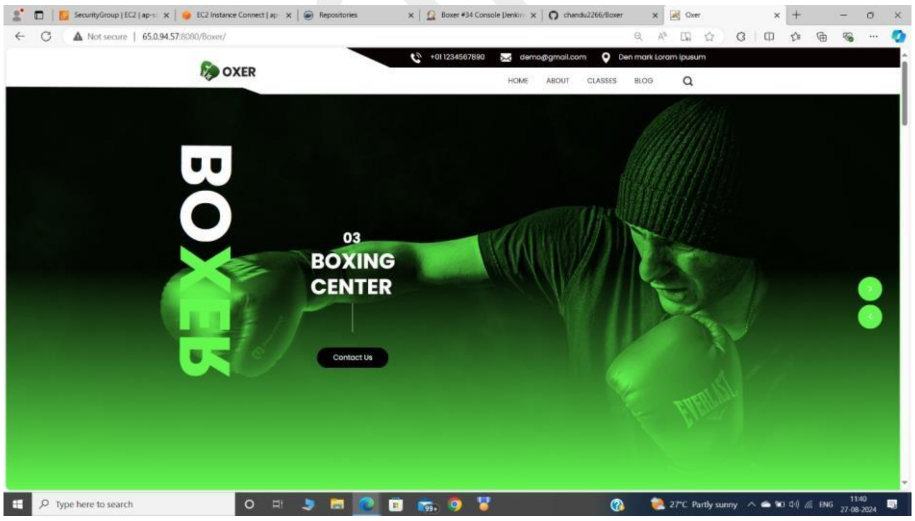
Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-02e127a6bffc5ab37	SSH	TCP	22	Custom	0.0.0.0/0	
-	Custom TCP	TCP	8080 - 8083	Custom		

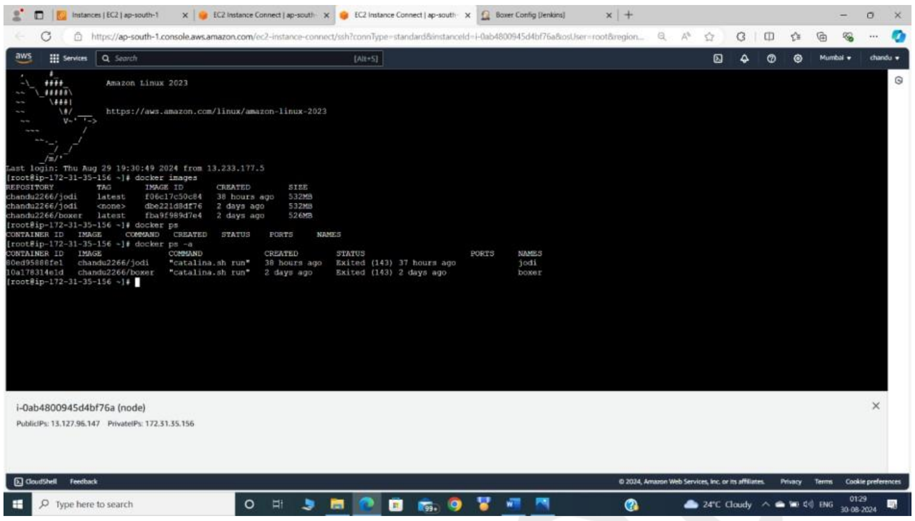
Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

➢ Copy the node **publicip:8081**

➢ The image will be deploy as shown in the image.



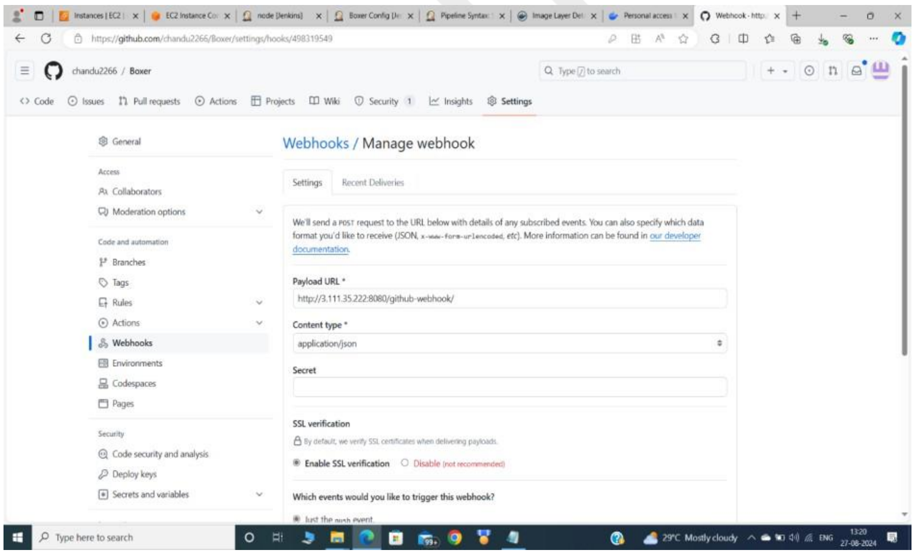
➢ We can check in node the image will shown are not in the node server.



```
Last login: Thu Aug 29 19:30:49 2024 from 13.233.177.5
[root@ip-172-31-35-156 ~]# docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
chandu2266/jodi     latest    f06c17c50c84   38 hours ago     532MB
chandu2266/jodi     <none>    dbe221d8df76   2 days ago       532MB
chandu2266/boxer    latest    fba9f989d7e4   2 days ago       526MB
[root@ip-172-31-35-156 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-35-156 ~]# docker ps -a
CONTAINER ID   IMAGE              COMMAND             CREATED        STATUS                      PORTS     NAMES
80ed95888fe1   chandu2266/jodi    "catalina.sh run"   38 hours ago   Exited (143) 37 hours ago             jodi
10a178314e1d   chandu2266/boxer   "catalina.sh run"   2 days ago     Exited (143) 2 days ago               boxer
[root@ip-172-31-35-156 ~]#
```

i-0ab4800945d4bf76a (node)
PublicIPs: 13.127.96.147 PrivateIPs: 172.31.35.156

- We can add a webhook for any changes done in the index.xml file the update will automatically done.



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

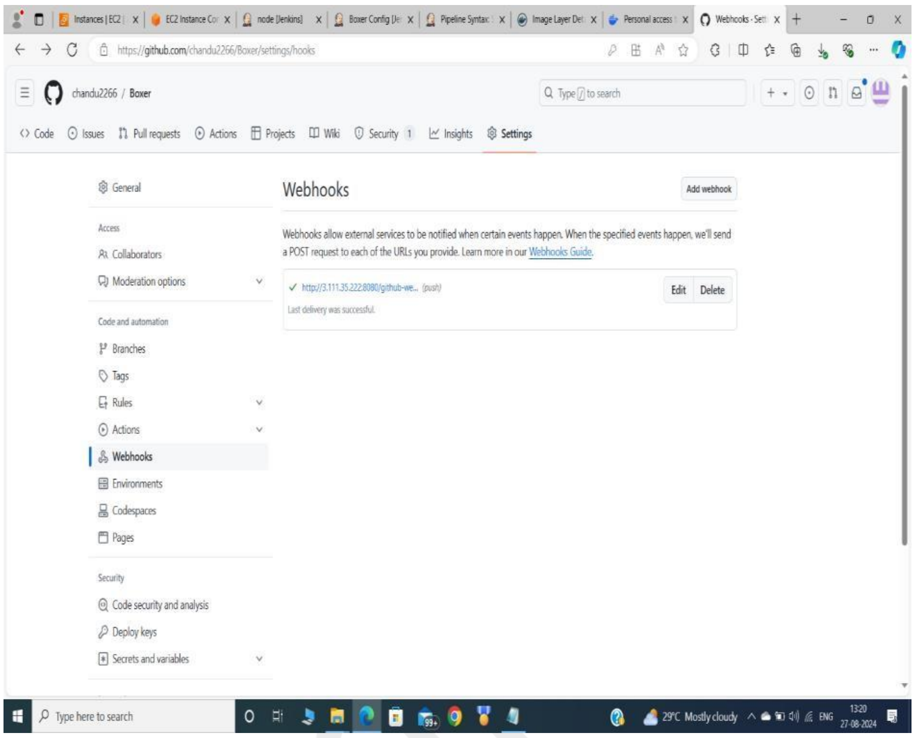
Payload URL *: http://3.111.35.222:8080/github-webhook/

Content type *: application/json

Secret:

SSL verification
By default, we verify SSL certificates when delivering payloads.
[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?
[✓ Just the push event]



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://3.111.35.222:8080/github-we... (push)

Last delivery was successful.

The complte pipeline script for the docker project-1

```
pipeline {

   agent { label 'dev' }

   tools {

       maven 'M3'

   }

   stages {

       stage('GIT') {

           steps {

               git branch: 'main', url: 'https://github.com/chandu2266/Boxer.git'
```

```
    }

  }

  stage('build') {

    steps {

        sh 'mvn clean package'

    }

  }

  stage ('container') {

    steps {

        sh 'docker rm -f boxer'

        sh 'docker rmi -f chandu2266/boxer'

    }

  }

  stage('docker build and push') {

   steps {

       script {

        withDockerRegistry(credentialsId: 'docker-boxer') {

            sh "docker build -t chandu2266/boxer ."

            sh "docker push chandu2266/boxer"

        }

       }

    }
```

```
        }

        stage ('docker run') {

          steps {

            sh 'docker run -d -p 8081:8080 --name boxer chandu2266/boxer'

          }

        }

    }

}
```