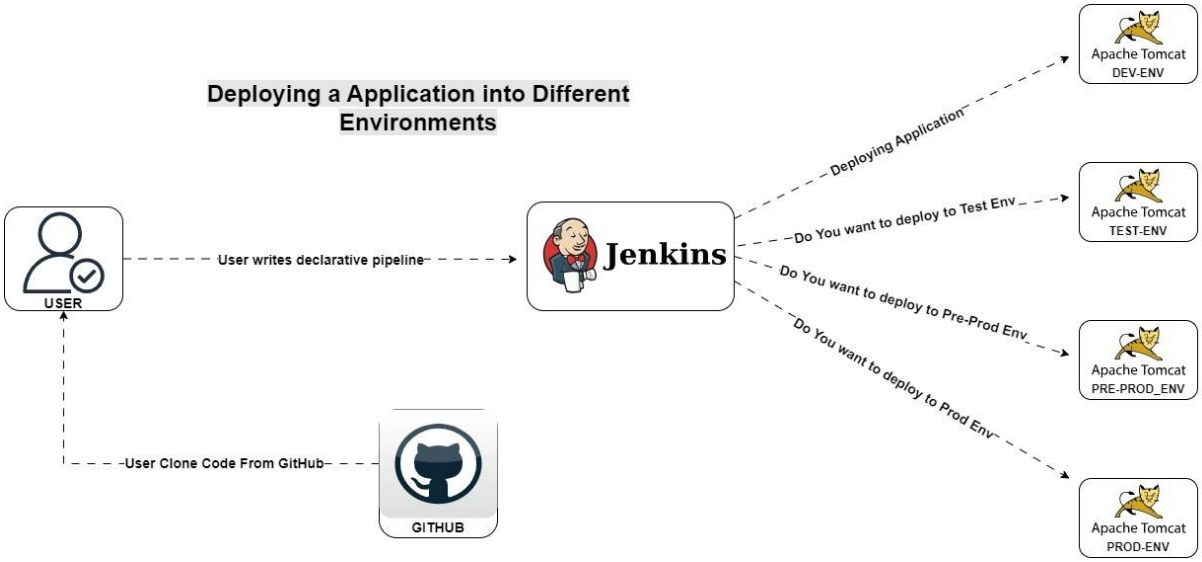
PROJECT – 1

COURSE: DEVOPS

MODULE: Deploying an Application into Different Environments.

TRAINER: Mr. MADHUKAR



Name: K.Udaya Bhanu

Mobile: 8688161889

Email: udayabhanu072@gmail.com

1.Create AWS instance and connect to web server:

- First enter into AWS console login page.
- Login into AWS account.
- Select Ec2 service in AWS account.
- Create new instance by click on launch instance.
- Give name, select ubuntu and keypair name click on launch instance.

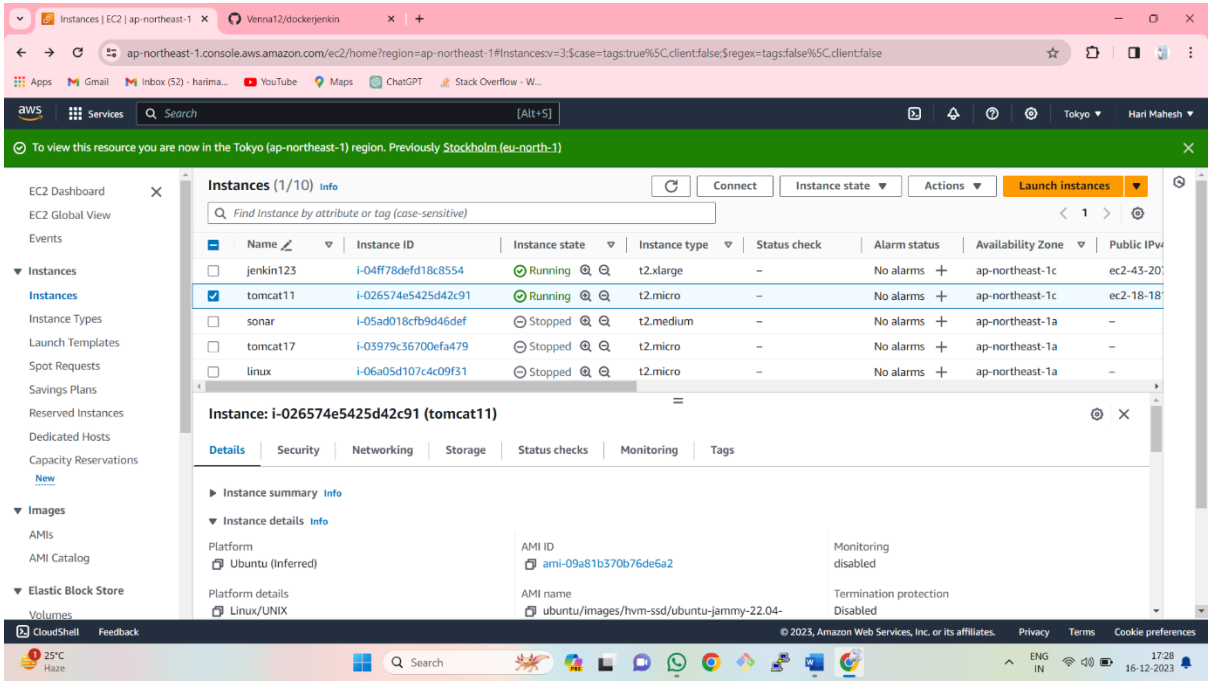
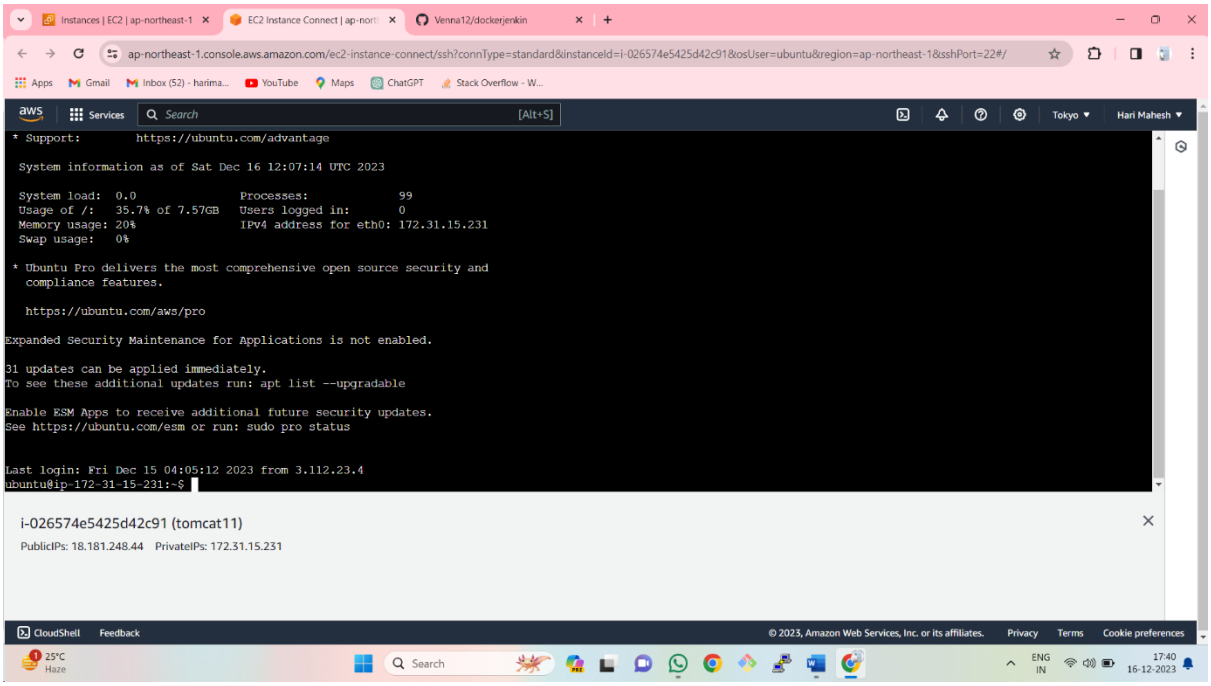


Fig.1 After completing launch instance.

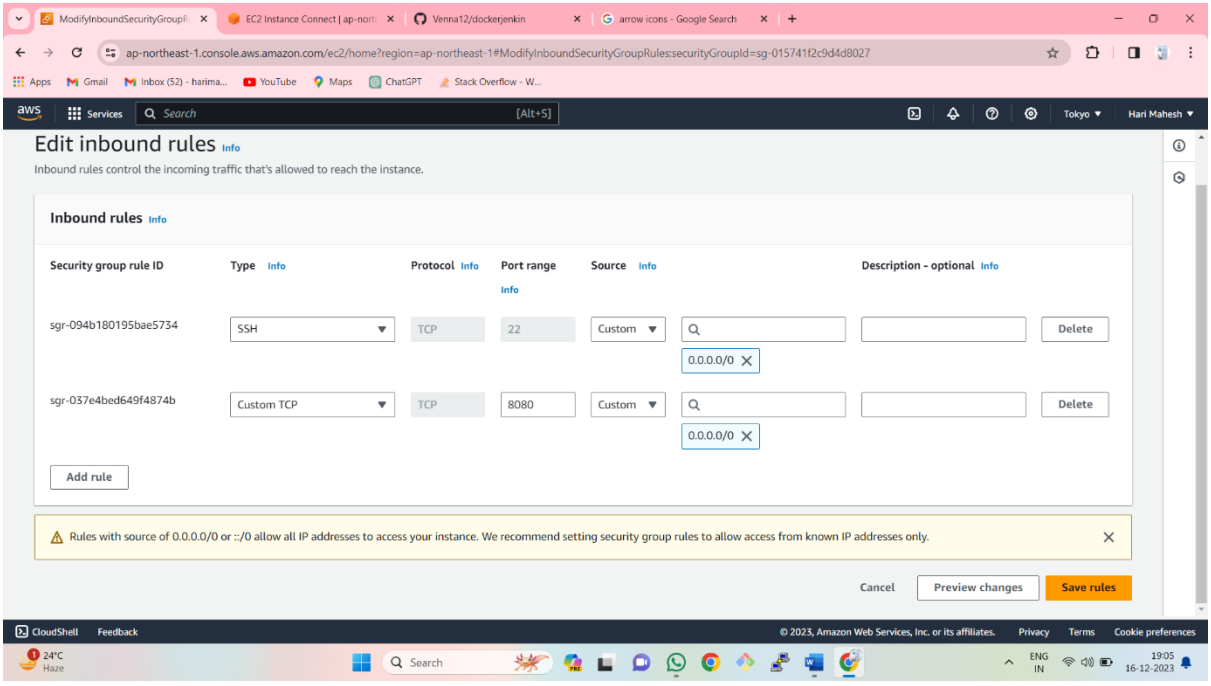
- After launch instance click on connect then it enters into web server.



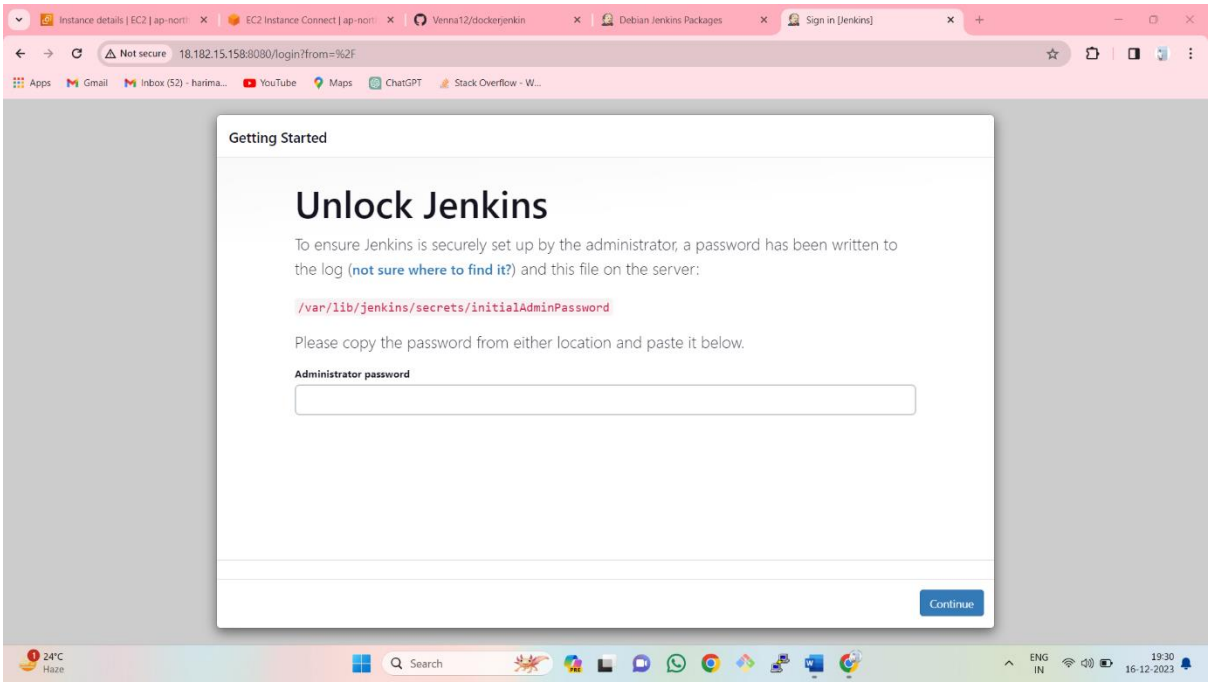
- After enter into server the window looks like above.
- To convert root user, enter sudo -i.

2.Installation of java, maven and Jenkins:

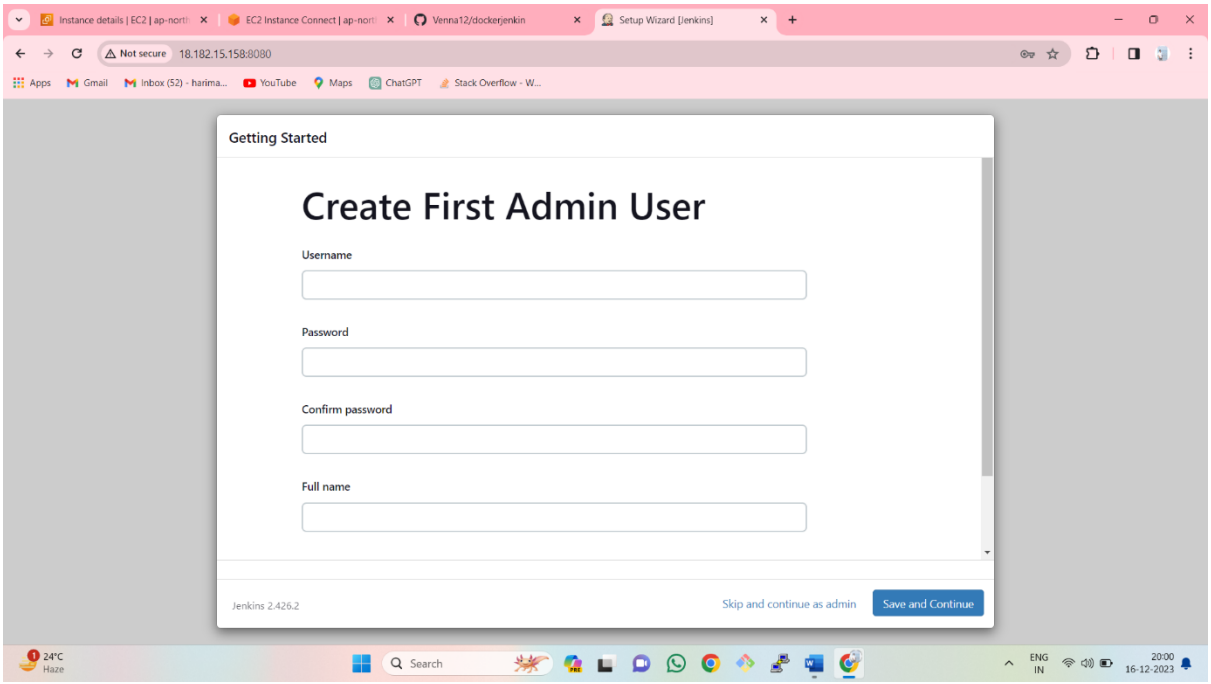
- Install java and maven for installing Jenkins using below commands;
  - apt update -y
  - apt install default-jdk -y
  - apt install maven -y
- After completing installation of java and maven, install Jenkins.
- Open new tab and search download Jenkins.
- Click on Jenkins download and deployment.
- Select ubuntu/debian.
- Then its shows jenkins debian packages.
- Copy from that package below commands;
  - sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
    https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
  - echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  - https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null
  - sudo apt-get update
  - sudo apt-get install jenkins
- Enter above commands then jenkins installed successful.
- To check weather jenkins are created or not, command:
  systemctl status jenkins.
- Go to instance page and change security settings →inbound actions → add rule →port number=8080 then save rule.



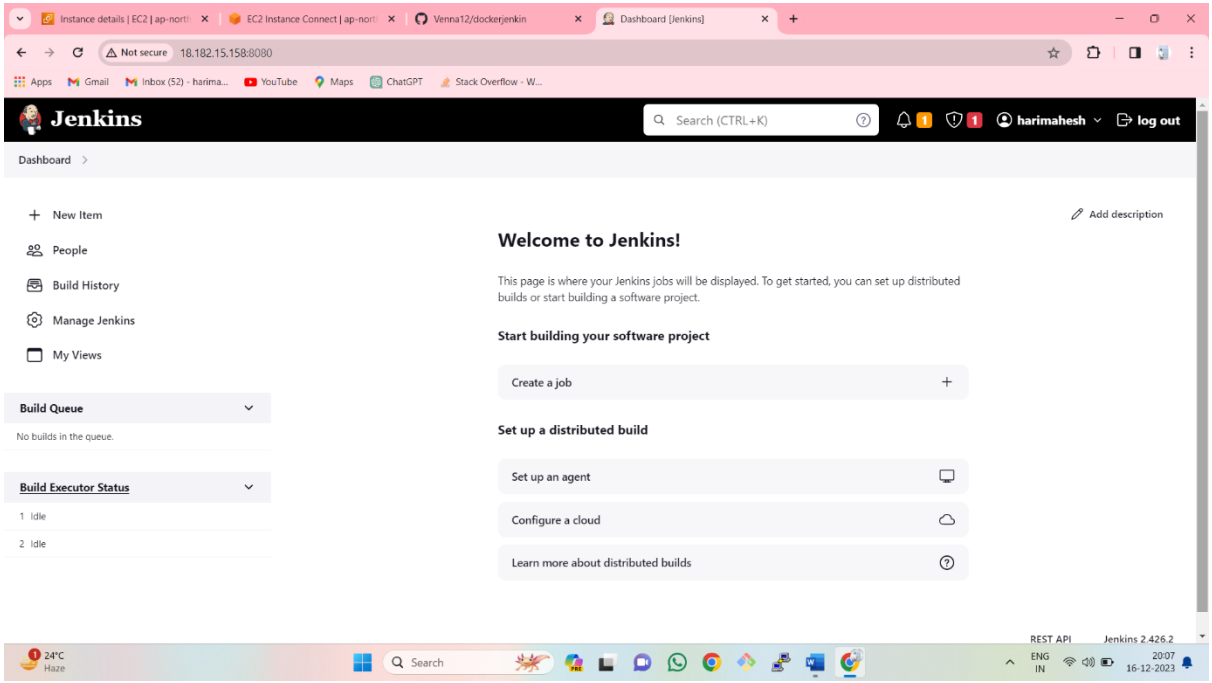
- Copy public IP address of launch instance and paste in new tab and add :8080 like this 18.181.248.44:8080.
- Then it asks administration password and shows password location like below shows:
  /var/lib/jenkins/secrets/initialAdminPassword



- After coping the location /var/lib/jenkins/secrets/initialAdminPassword
- Enter cat /var/lib/jenkins/secrets/initialAdminPassword to see password.
- To copy password from the server and enter in the unlock jenkins administration password.



- Now create username and password for jenkins dashboard.
- After completing creation dashboard looks like below fig.

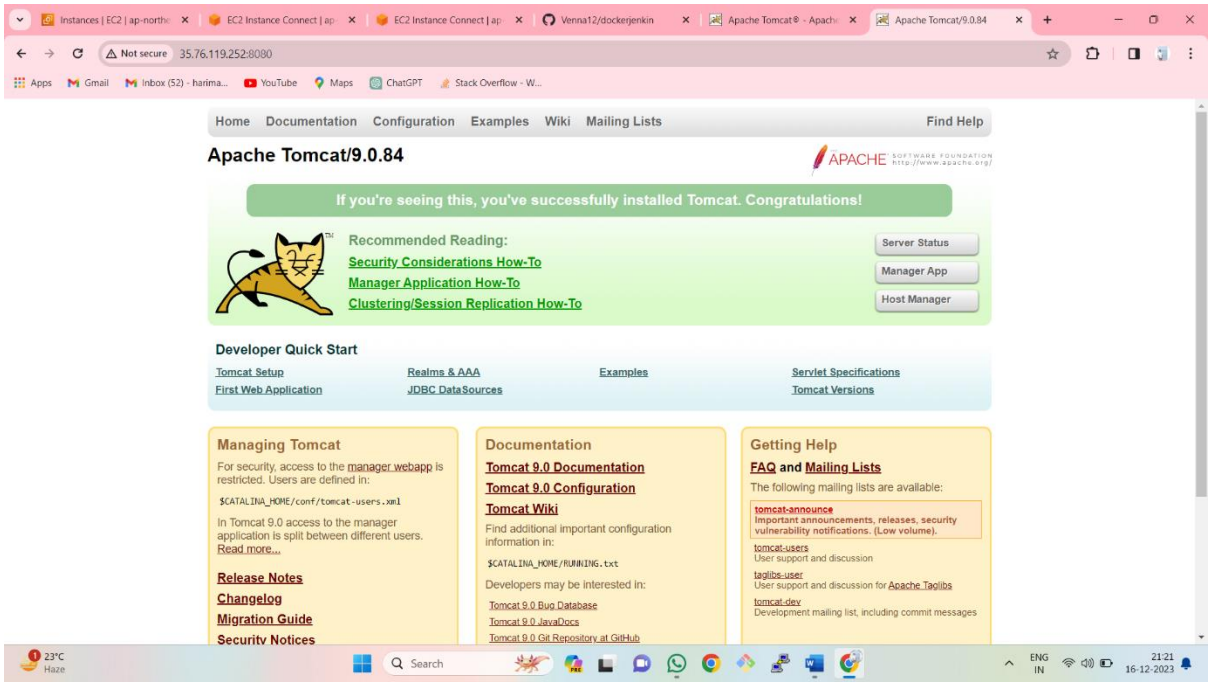


Jenkins dashboard

3. Installation of tomcat:

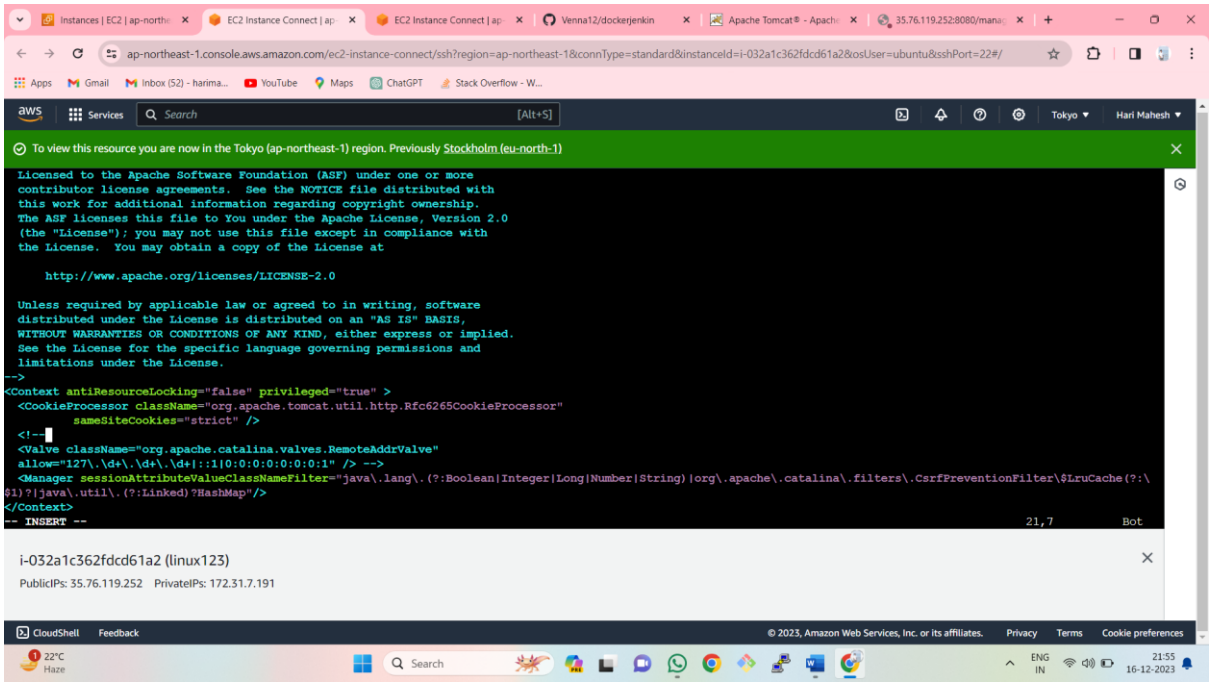
- Again, take instance or take old instance.
- First install java commands:
  - apt update -y
  - apt install default-jdk -y
- Now install tomcat enter below commands:
  - Search tomcat download in new tab
  - Copy link address of tar file.
  - https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.84/bin/apache-tomcat-9.0.84.tar.gz
  - Paste in Ec2 instance servers.
  - Enter wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.84/bin/apache-tomcat-9.0.84.tar.gz
  - This is tar file so untar that → tar -xvzf (file name)
  - To rename that file → mv (file name)
  - cd tomcat
  - ls
  - cd bin

- ls
- ./startup.sh
- Now tomcat is started.
- Add rule in security settings of Ec2 instance →inbound actions →add rule →port number 8080→ save rules.
- Copy public IP address of Ec2 instance paste in new tab add :8080
- Now tomcat dashboard appears looks like below fig.

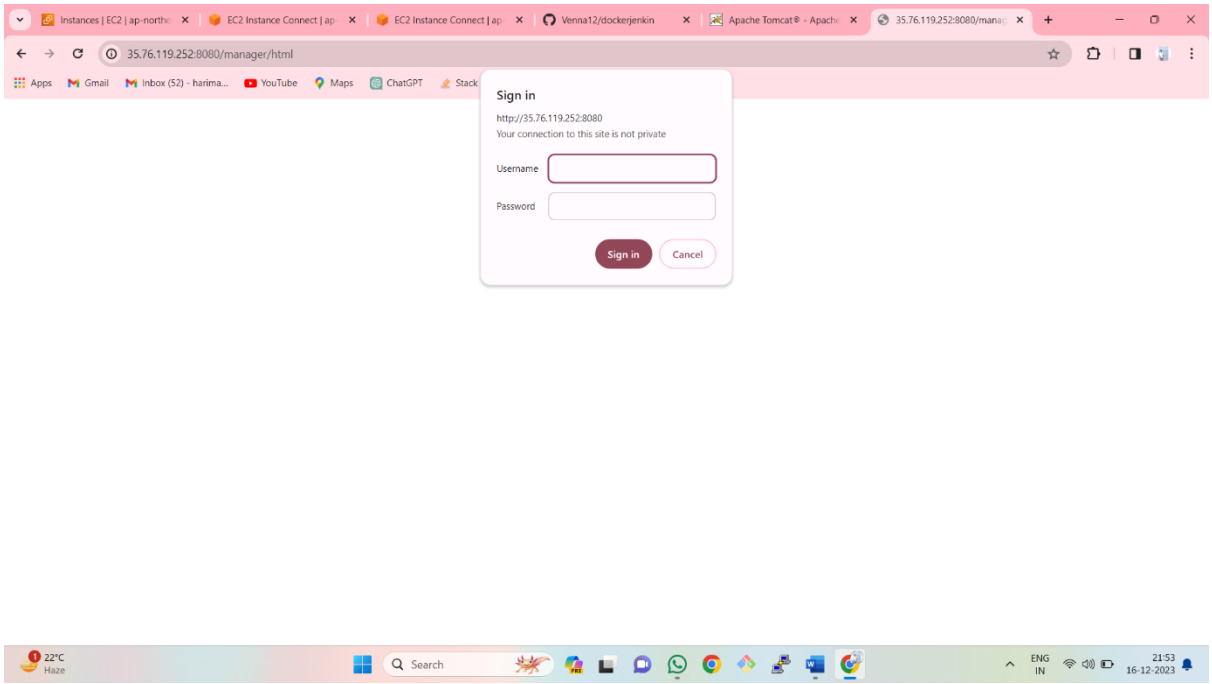


- But not open manager app then add 2two comments in host-manager and manager in Ec2 server.
- cd tomcat
- ls
- cd webapps
- ls
- cd host-manager
- ls
- cd META -INF/
- ls
- vi context.xml
- comment 2 lines <!—
  -->
- then save →esc+shift+: wq enter

- shown below:

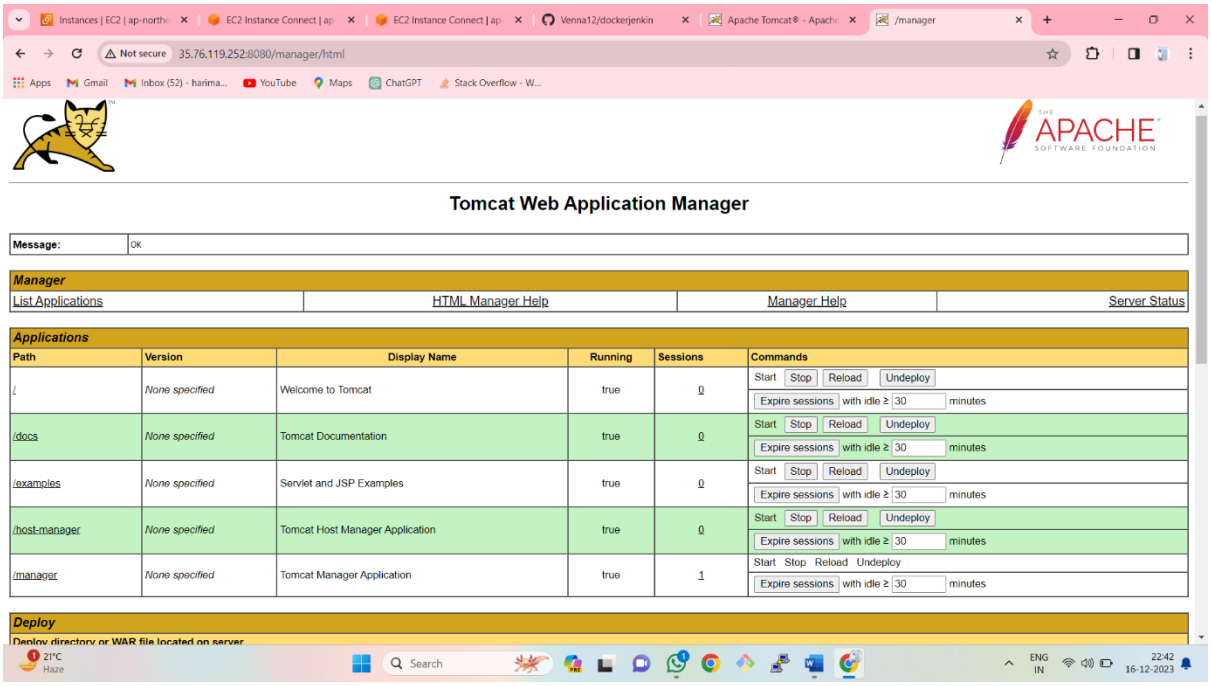


- Now manager app open and ask login looks like below fig:



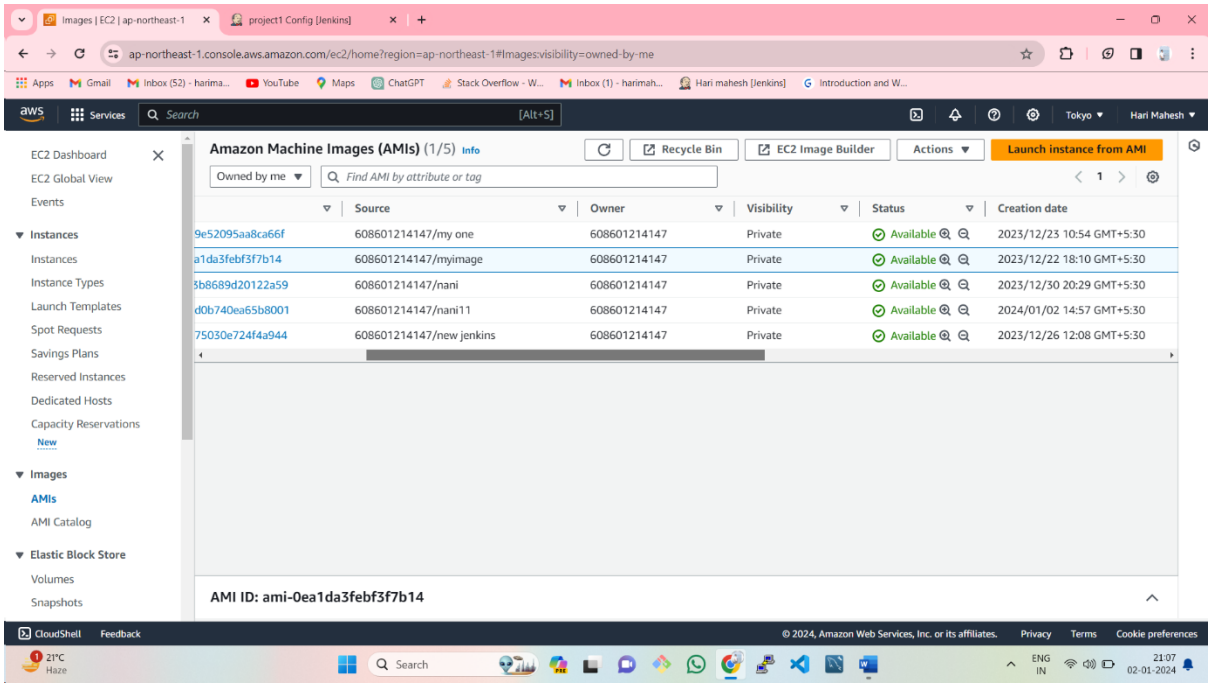
- For this username and password again go to Ec2 instance and do some changes in tomcat-user.xml

- cd tomcat
- ls
- cd conf
- ls
- vi tomcat-user.xml
- add some password related content
  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <role rolename="manager-jmx"/>
  <role rolename="manager-status"/>
  <user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>
  <user username="deployer" password="deployer" roles="manager-script"/>
  <user username="mahesh" password="Mahesh@2344" roles="manager-gui"/>
- Then save esc+shift+: wq
- Now go to tomcat and enter username and password you enter in tomcat-user.xml
- Then tomcat web application manager opens looks like below fig.



4. USING AMAZON MACHINE IMAGE (AMI) TO CREATE DIFFERENT TOMCAT ENVIRONMENTS:

- Go to ec2 dashboard and select instance which have tomcat installed on it.
- Click on actions then select image and templates.
- Give name as your wish like hari, then click on create image.
- Now go to ec2 dashboard click on AMI in images.
- It takes some time to available and after available select image and click on launch instance from AMI.
- Window looks like below fig.



Launch instance from AMI

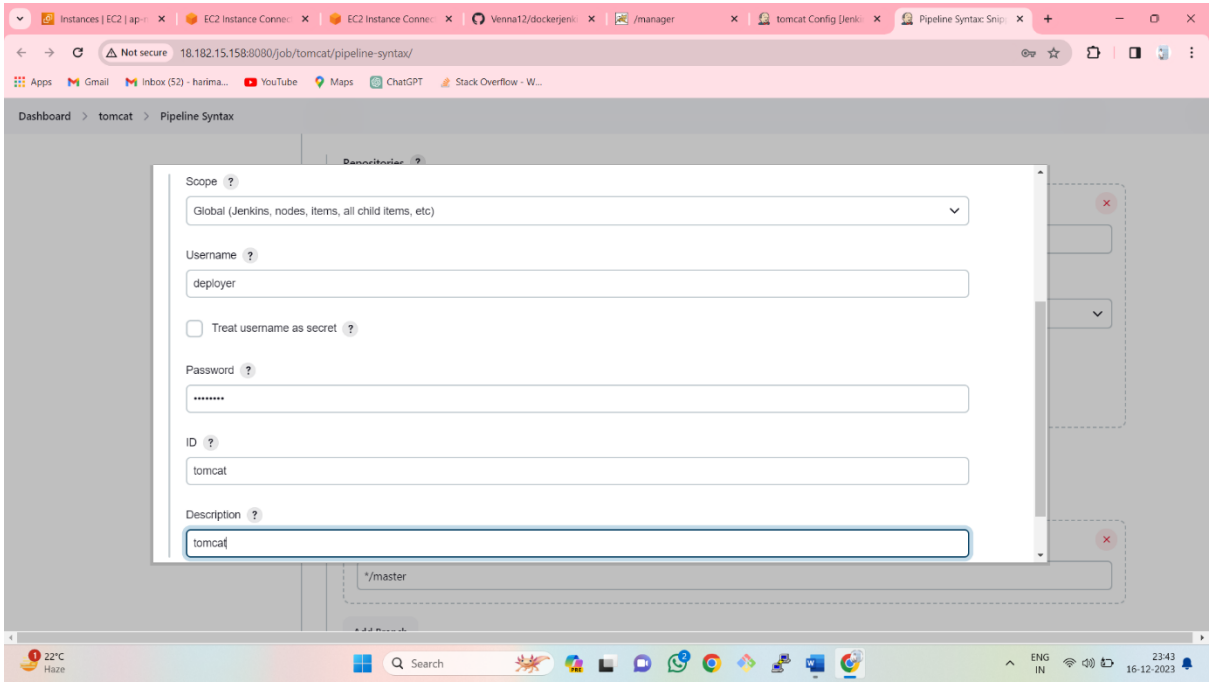
- Then give name, select keypair and add network settings add rule and click on launch instance.
- Now instance created with tomcat installation.
- Now create same like this using AMI to launch again 2 instances for tomcat.
- Open 4 tomcat servers and go to bin and restart the application using
    - cd /tomcat/bin
    - ./startup.sh
- Now do this all the four tomcat servers to login.

5.create pipeline to Deploying an Application into Different Environments:

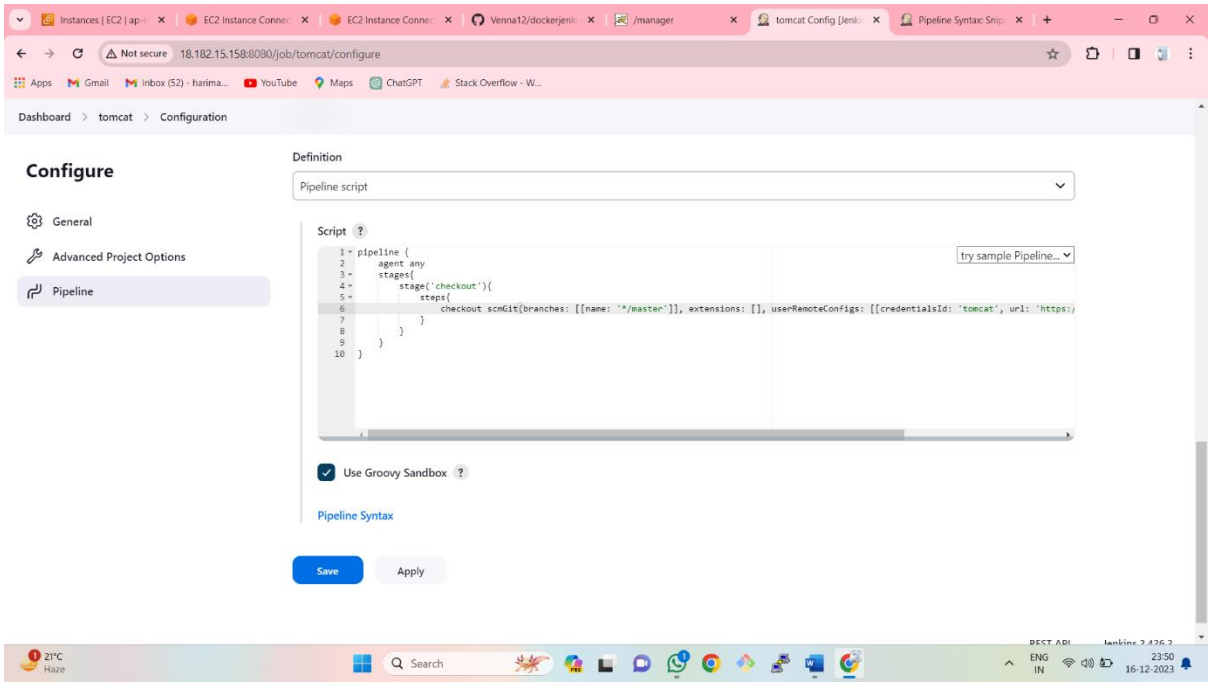
- Open Jenkins dashboard and click on new item, select pipeline then click on ok.
- Now configure pipeline script.
- Write pipeline with input message.

Stage-1: checkout

- In the above figure select checkout: checkout for version control in sample steps.
- Give git hub code URL and add Credentials →Jenkins→give username, password, Id, description like below figure.
- Now click on generate pipeline syntax and copy and add in script writing.

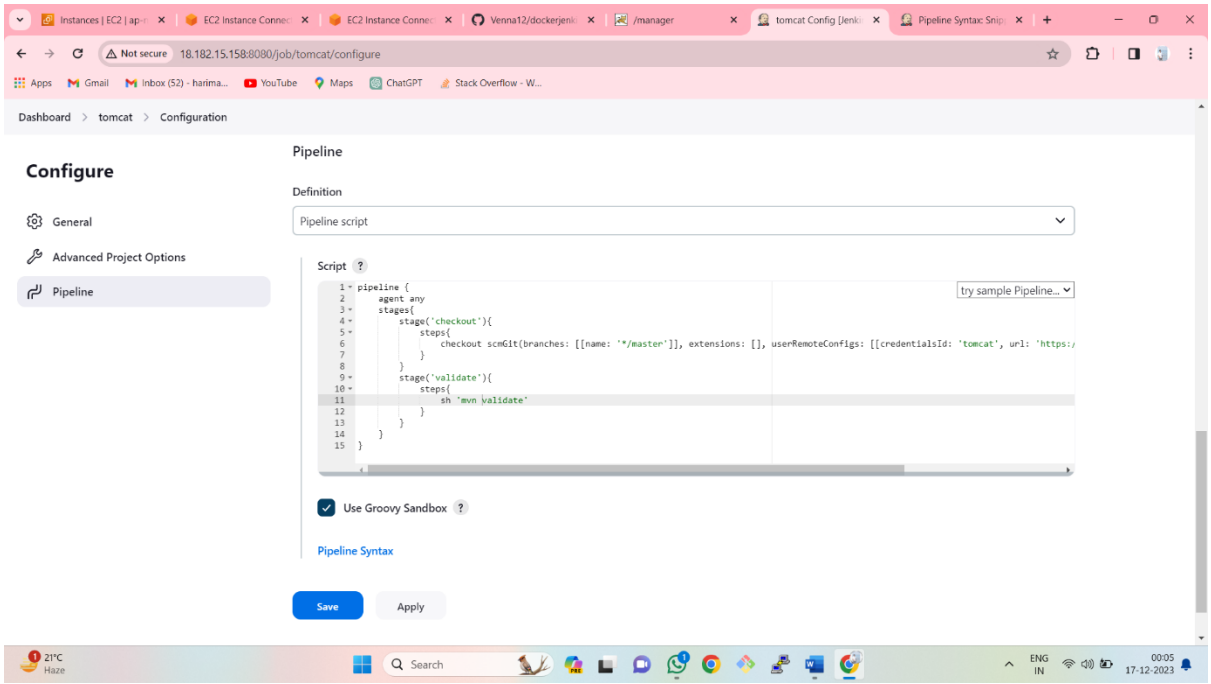


- checkout scmGit(branches: [[name: '*/master']], extensions: [], userRemoteConfigs: [[credentialsId: 'tomcat', url: 'https://github.com/harimahesh2344/taxi-booking.git']]).
- Pipeline script write stages like shown in below fig.



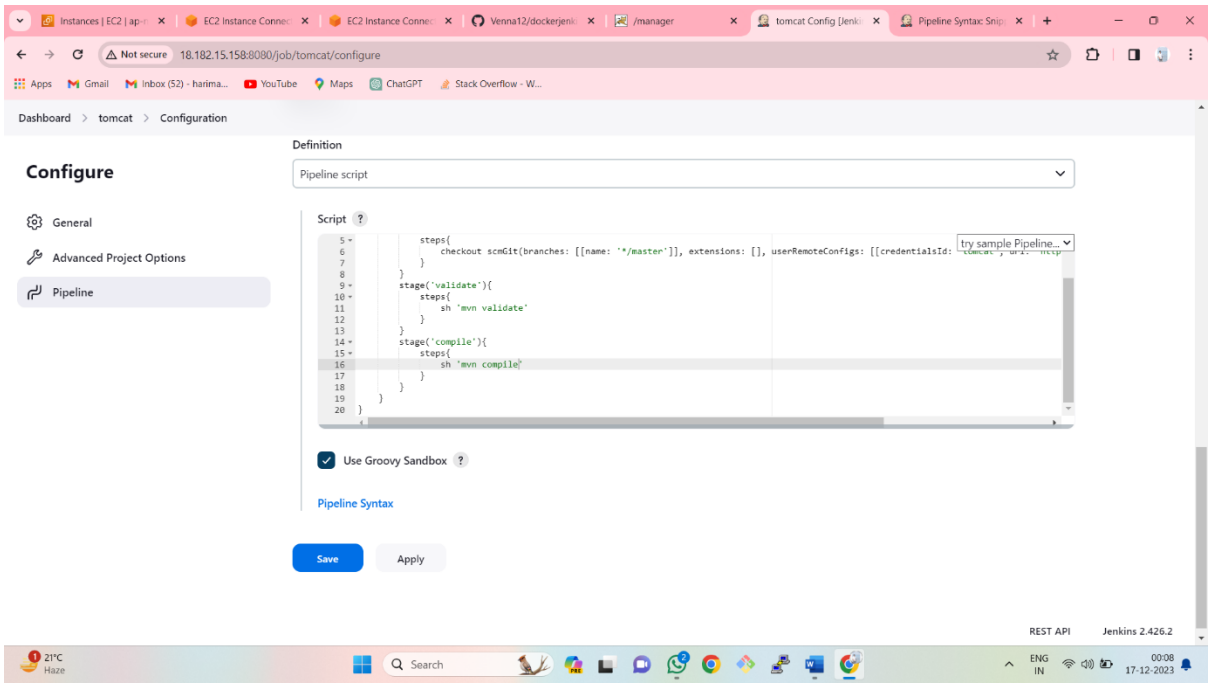
Stage-2: validate

- Write validate stage shown in below:



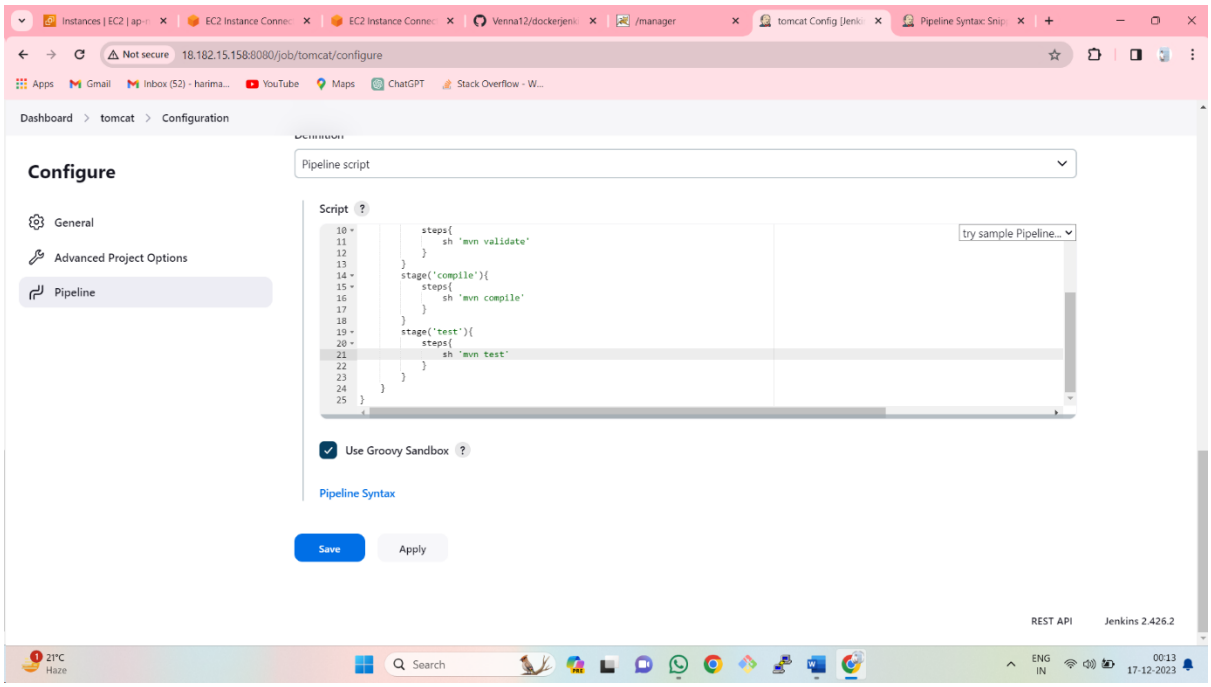
Stage-3: compile

- Write compile stage shown in below fig.



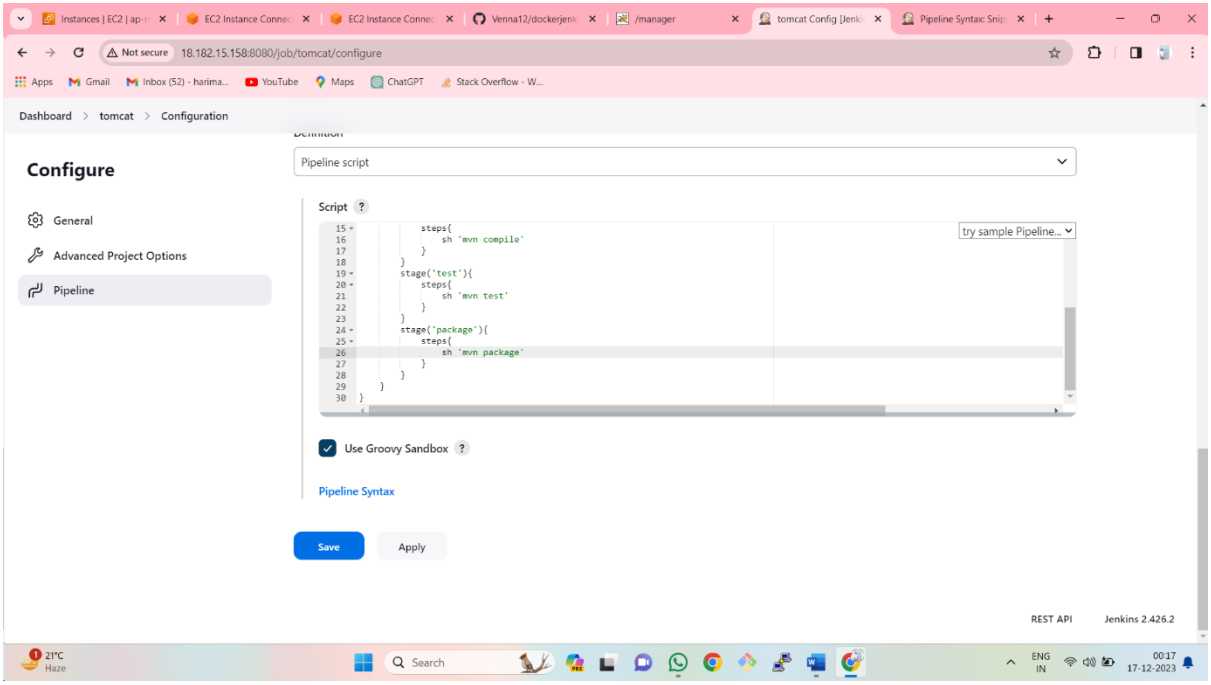
Stage-4: Test

- Write test stage shown in below fig.



Stage-5: Package

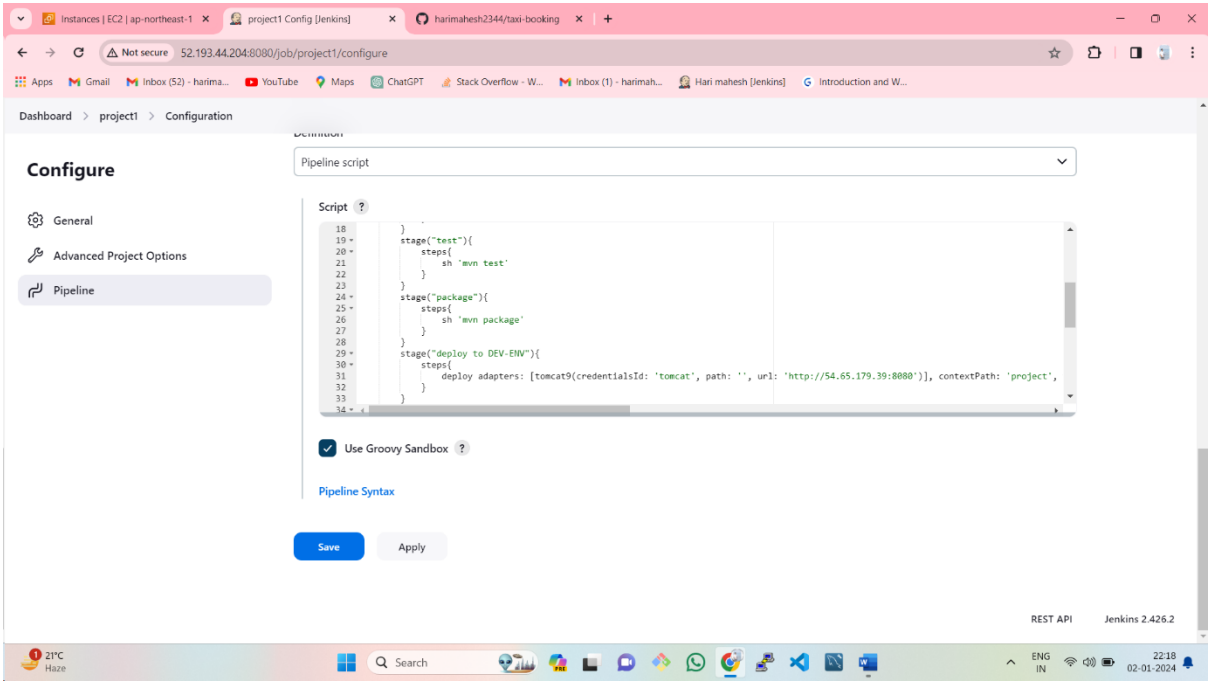
- Write package stage shown in below fig.



- Then apply and save.

Stage-6: Deploying application to DEV-ENV:

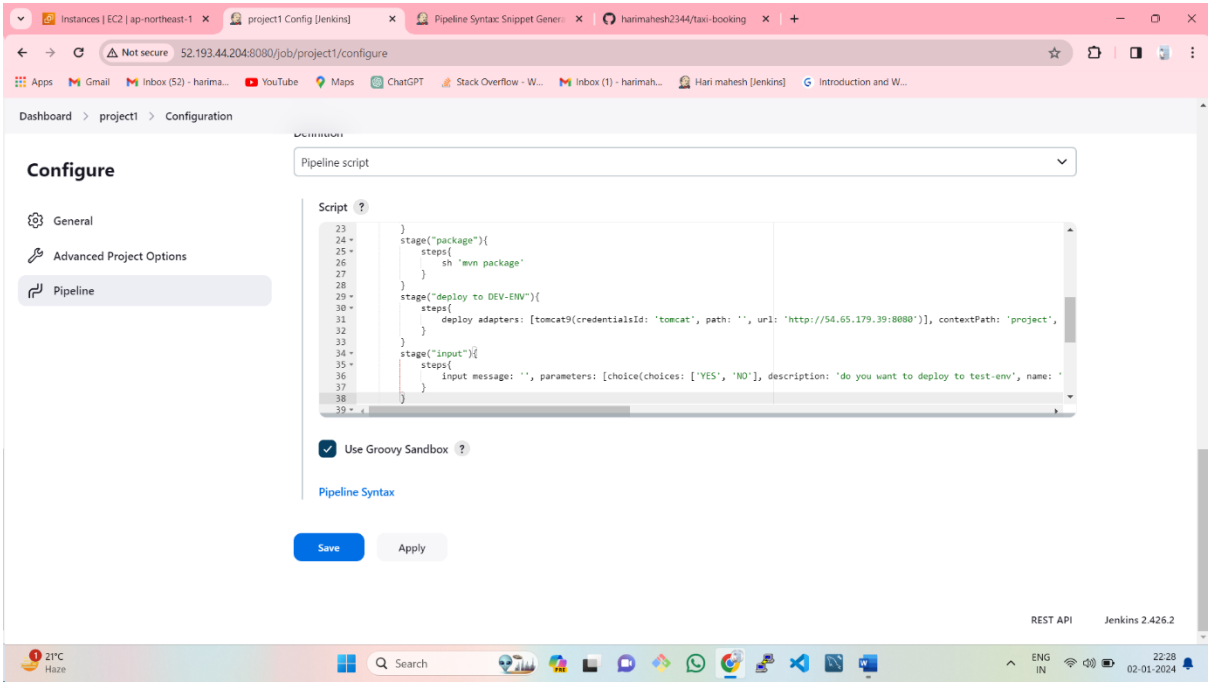
- First add plugin deploy to container.
- Go to manage jenkins→click on plugin→click available plugin→search deploy to container and install it.



- Now restart Jenkins in Ec2 instance command→systemctl restart jenkins.
- Now click on pipeline syntax and select deploy to container in sample step.
- Enter **/*.war in war/ear files.
- Give context path name
- Add container→select tomcat 9.x Remote.
- Add Credentials → select jenkins → give username, password, Id, description.
- Then click on generate syntax
- Copy and add in the script like shown in above fig.

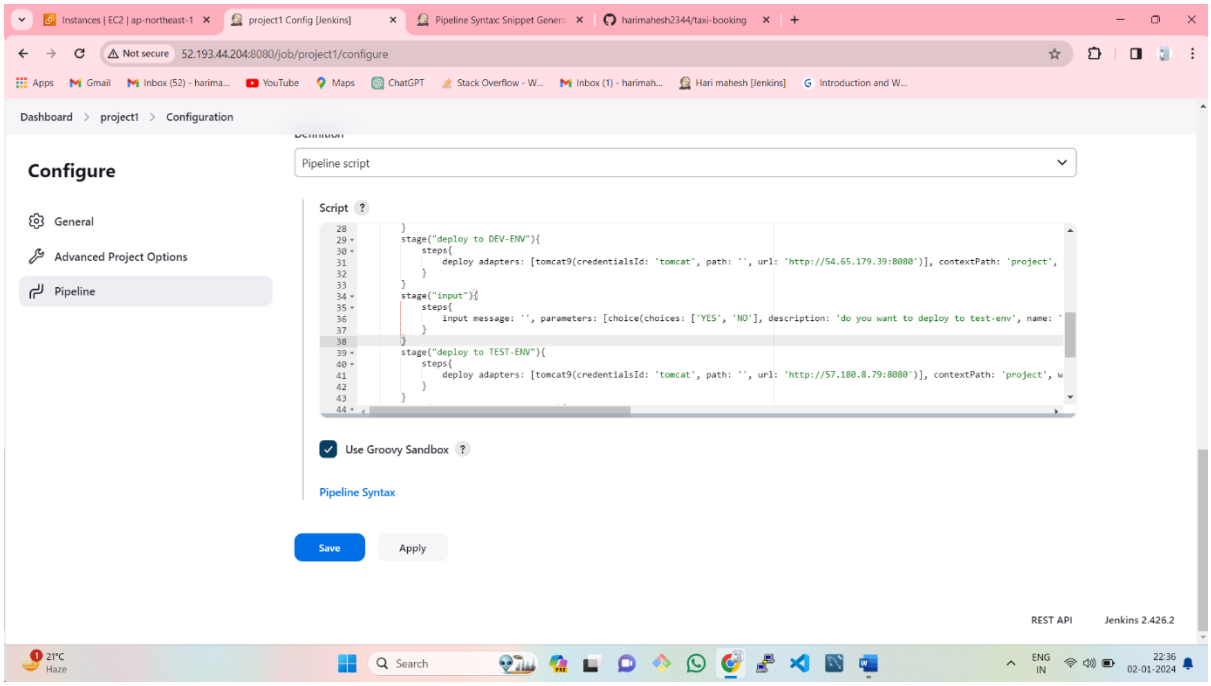
Stage-7: input

- To ask do you want to deploy to dev, test, pre-prod, prod environments.
- Go to pipeline syntax and select input: wait for interactive input then click on advanced.
- Then add parameter like choice.
- Now add names of three environments individually and give choice like yes or no and add description.
- Now click on generate pipeline syntax and copy then paste in pipeline script of stage input.
- Script for input looks like below fig.



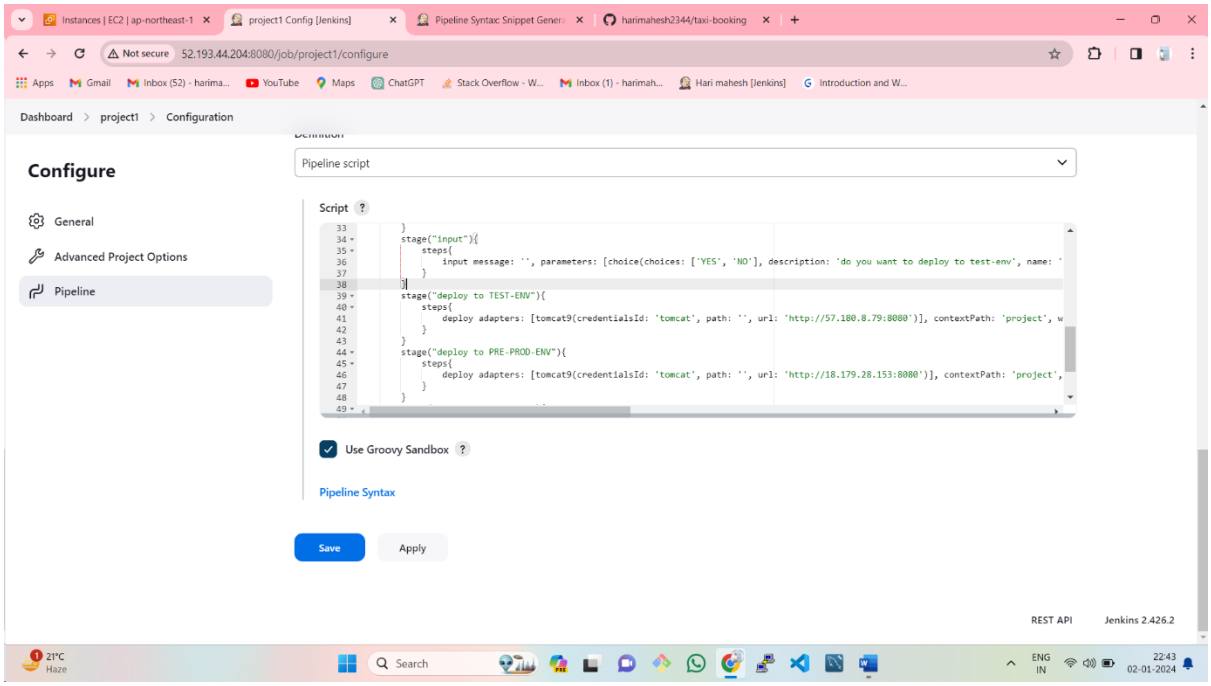
Stage-8: Deploying application to TEST-ENV:

- Now click on pipeline syntax and select deploy to container in sample step.
- Enter **/*.war in war/ear files.
- Give context path name
- Add container→select tomcat 9.x Remote.
- Add Credentials → select jenkins → give username, password, Id, description.
- Then click on generate syntax
- Copy and add in the script like shown in below fig.



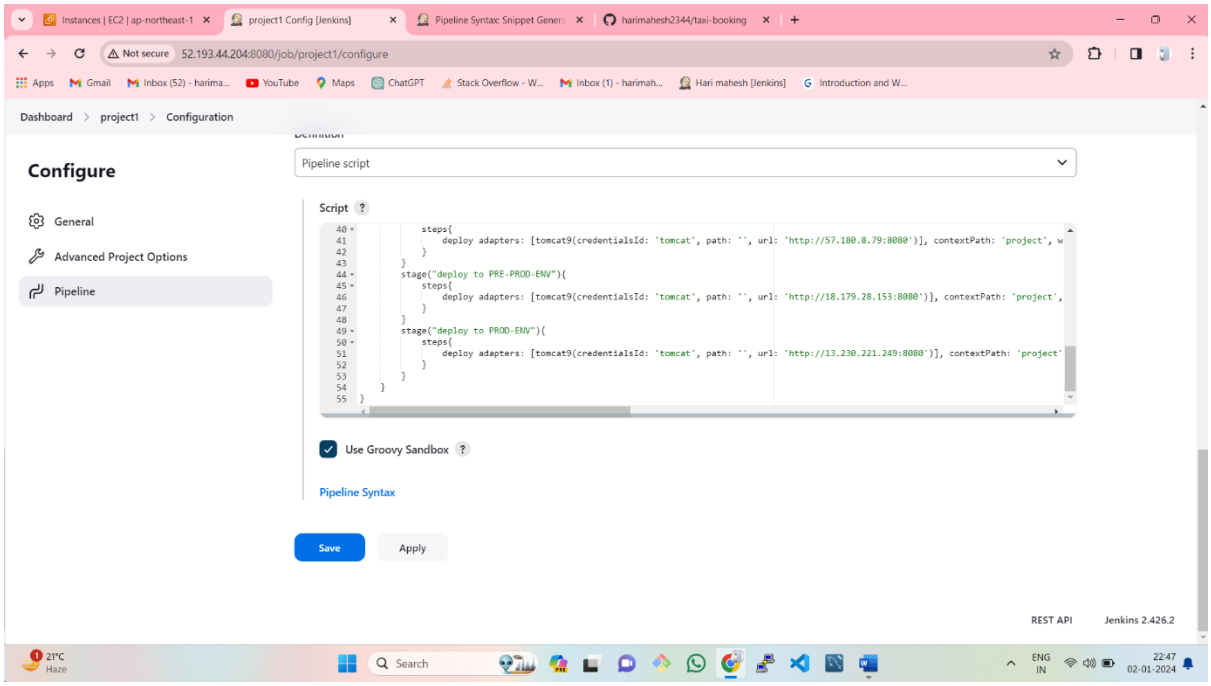
Stage-9: Deploying application to PRE-PROD-ENV:

- Now click on pipeline syntax and select deploy to container in sample step.
- Enter **/*.war in war/ear files.
- Give context path name
- Add container→select tomcat 9.x Remote.
- Add Credentials → select jenkins → give username, password, Id, description.
- Then click on generate syntax
- Copy and add in the script like shown in below fig.



Stage-10: Deploying application to PROD-ENV:

- Now click on pipeline syntax and select deploy to container in sample step.
- Enter **/*.war in war/ear files.
- Give context path name
- Add container→select tomcat 9.x Remote.
- Add Credentials → select jenkins → give username, password, Id, description.
- Then click on generate syntax
- Copy and add in the script like shown in below fig.

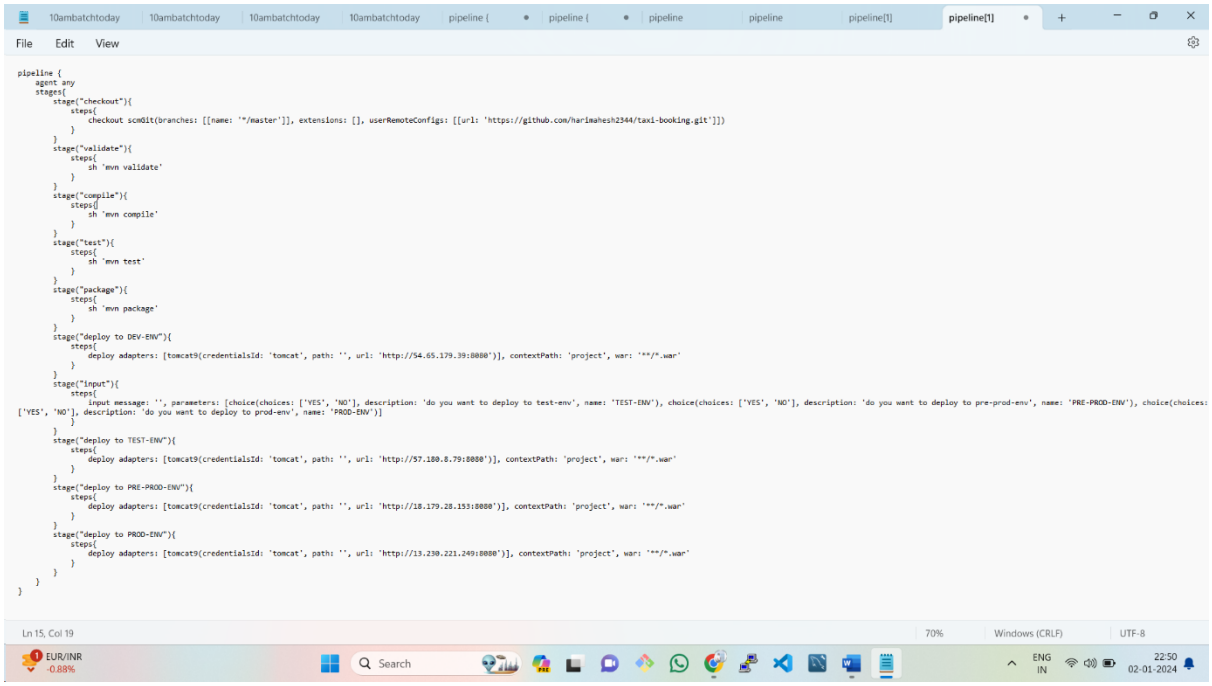


Entire pipeline script:

```
pipeline {
    agent any
    stages{
        stage("checkout"){
            steps{
                checkout scmGit(branches: [[name: '*/master']], extensions: [], 
userRemoteConfigs: [[url: 'https://github.com/harimahesh2344/taxi-booking.git']])
            }
        }
        stage("validate"){
            steps{
                sh 'mvn validate'
            }
        }
        stage("compile"){
            steps{
                sh 'mvn compile'
            }
```

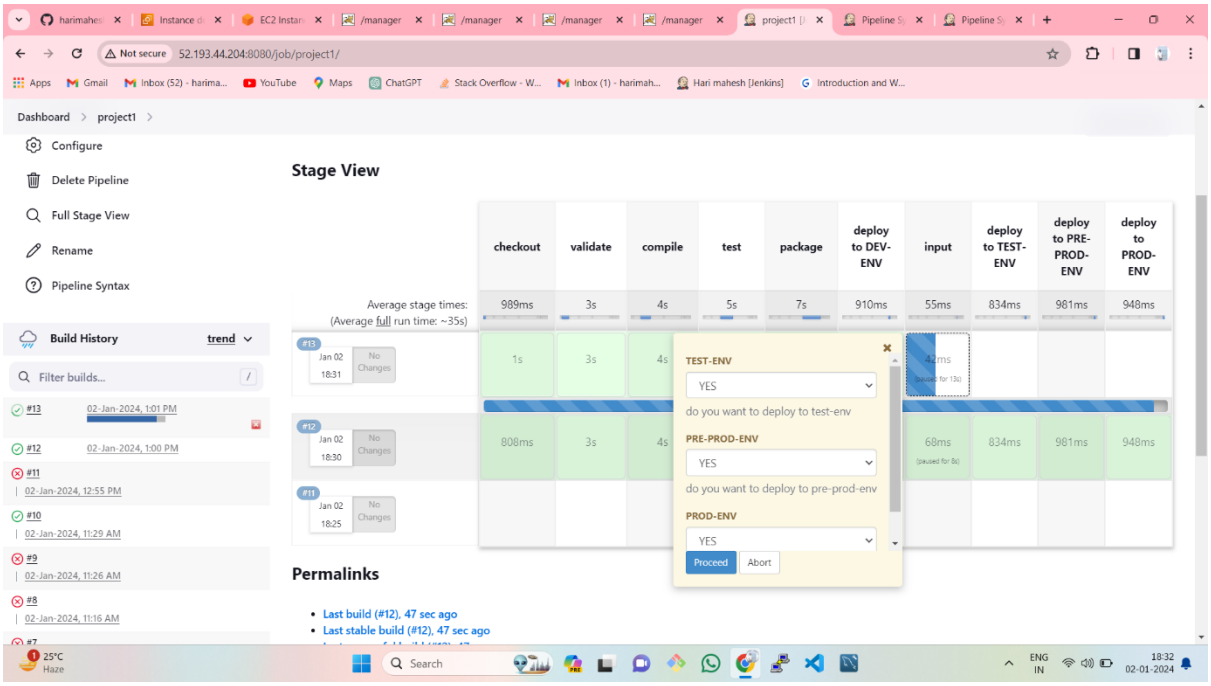
```
    }
    stage("test"){
        steps{
            sh 'mvn test'
        }
    }
    stage("package"){
        steps{
            sh 'mvn package'
        }
    }
    stage("deploy to DEV-ENV"){
        steps{
            deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url: 'http://54.65.179.39:8080')], contextPath: 'project', war: '**/*.war'
        }
    }
    stage("input"){
        steps{
            input message: '', parameters: [choice(choices: ['YES', 'NO'], description: 'do you want to deploy to test-env', name: 'TEST-ENV'), choice(choices: ['YES', 'NO'], description: 'do you want to deploy to pre-prod-env', name: 'PRE-PROD-ENV'), choice(choices: ['YES', 'NO'], description: 'do you want to deploy to prod-env', name: 'PROD-ENV')]
        }
    }
    stage("deploy to TEST-ENV"){
        steps{
            deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url: 'http://57.180.8.79:8080')], contextPath: 'project', war: '**/*.war'
        }
    }
    stage("deploy to PRE-PROD-ENV"){
        steps{
```

```
            deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url:
'http://18.179.28.153:8080')], contextPath: 'project', war: '**/*.war'
        }
    }
    stage("deploy to PROD-ENV"){
        steps{
            deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url:
'http://13.230.221.249:8080')], contextPath: 'project', war: '**/*.war'
        }
    }
  }
}
```



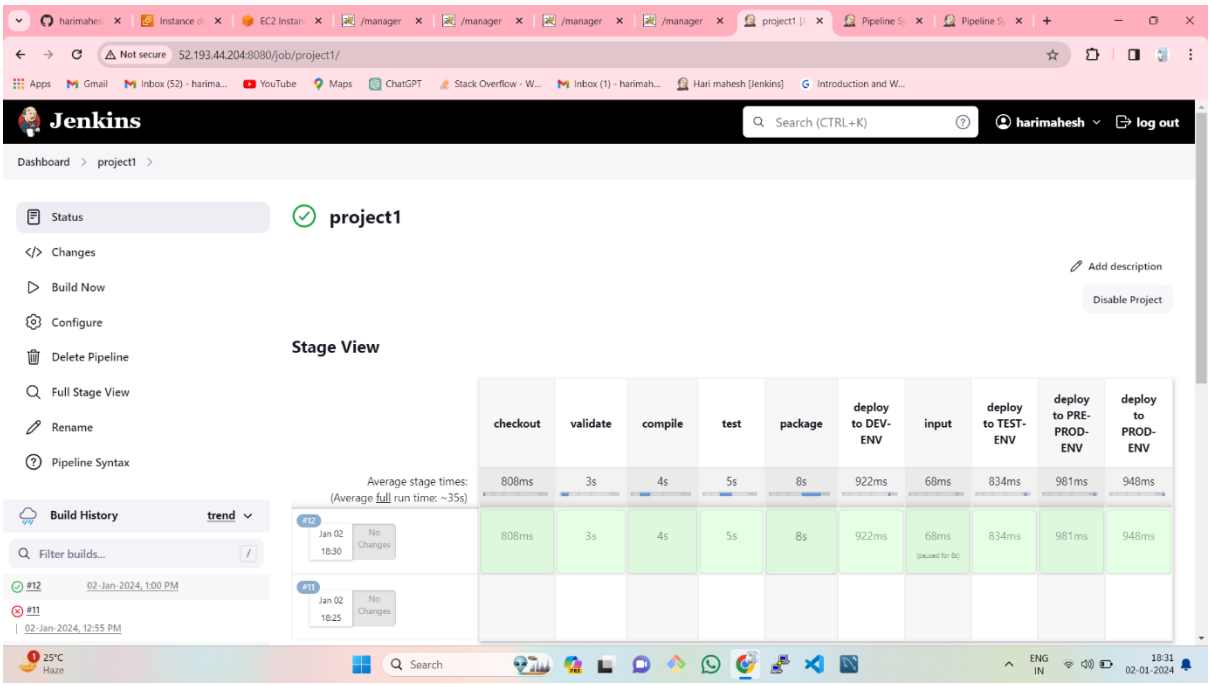
Entire pipeline

- Now click on build now.
- The entire pipeline is running.
- After completing dev stage, it asks do you want deploy or abort because we using input message.
- Now full stage view of pipeline is looks like below fig.



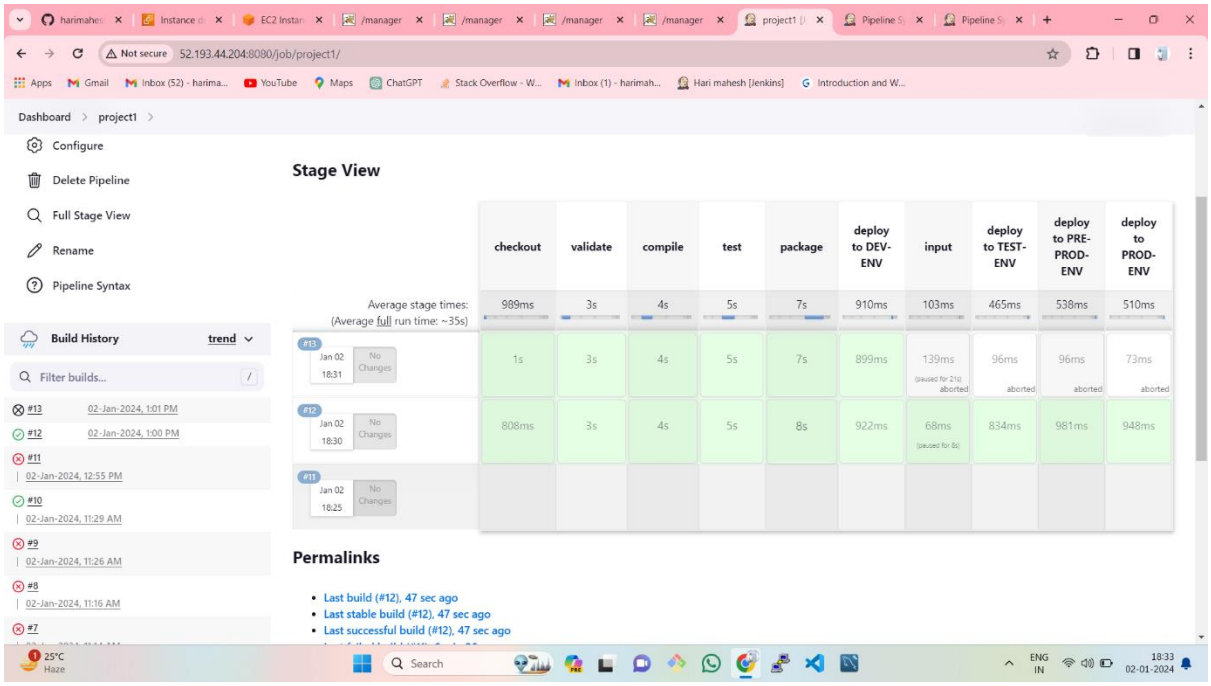
Ask for proceed or abort

- If you select yes and proceed then pipeline success
- Looks like below fig.



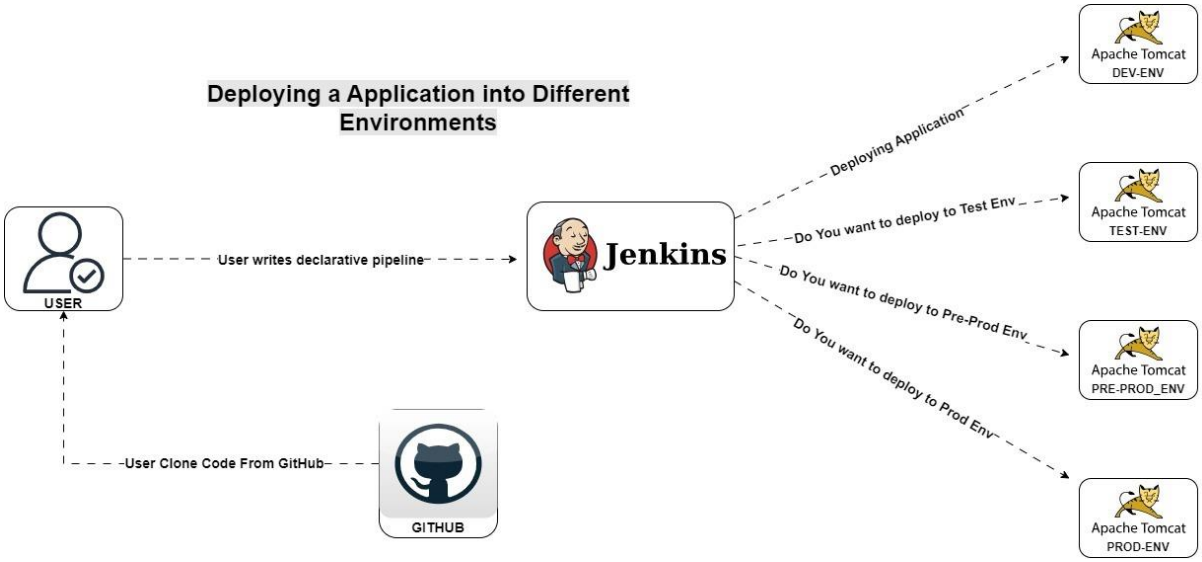
Full stage view of successful pipeline

- If you select abort the pipeline goes to aborted.
- Looks like below fig.



Pipeline aborted

- Now open four application servers and refresh.
- Application is successfully deployed in four servers at a time.



Deploying an Application into Different Environments is successfully deployed.