DevOps Project

Problem Statement:

Create an end to end CI/CD pipeline in AWS platform using Jenkins as the orchestration tool, Github as the SCM, Maven as the Build tool, Deploy in a docker instance and create a Docker image, Store the docker image in ECR, Achieve Kubernetes deployment using the ECR image. Build a sample java web app using maven

Approach:

Requirements:

- ✓ CI/CD pipeline System.
- ✓ Git &Github- local version control system.
- ✓ GitHub - As Distributed version control system.
- ✓ Jenkins - Continuous Integration tool.
- ✓ AWS ECR-Container Registry
- ✓ AWS EKS-Container Orchestration
- ✓ Maven - As a Build Tool.
- ✓ Docker -Containerization
- ✓ Kubernetes - As Container Management Tool

Step-1:

➢ Setup CI/CD with GitHub, Jenkins, Maven & Tomcat.

➢ Setup Jenkins

➢ Setup & Configure Maven, Git.

➢ Setup Tomcat Server.

➢ Integrating GitHub, Maven, Tomcat Server with Jenkins

➢ Create a CI and CD Job.

➢ Test the Deployment

Step-2:

➢ Setup CI/CD with GitHub, Jenkins, Maven & Docker.

➢ Setting up the docker Environment.

➢ Create an Image and Container on Docker Host.

➢ Integrate Docker Host with Jenkins.

➢ Create CI/CD Job on Jenkins to build and deploy on container.

yum

Step-3:

➢ Build and Deploy on Container.

➢ CI/CD with GitHub, Jenkins, Maven & Kubernetes.

➢ Setting up the Kubernetes (EKS).

➢ Write groovy file in jenkins to build the pipeline in declarative format.

➢ CI/CD Job to build code on Jenkins & Deploy it on Kubernetes.

Step-4:

➢ Deploy artifacts on the Kubernetes

➢ Write codes in the artifacts of docker and Kubernetes which we want to run.

➢ Now build the code in Jenkins.

➢ Check in Kubernetes the pods are getting created or not.

➢ Now copy the service IP and paste it in the browser and check the output.

Solution:

Git-Clone:

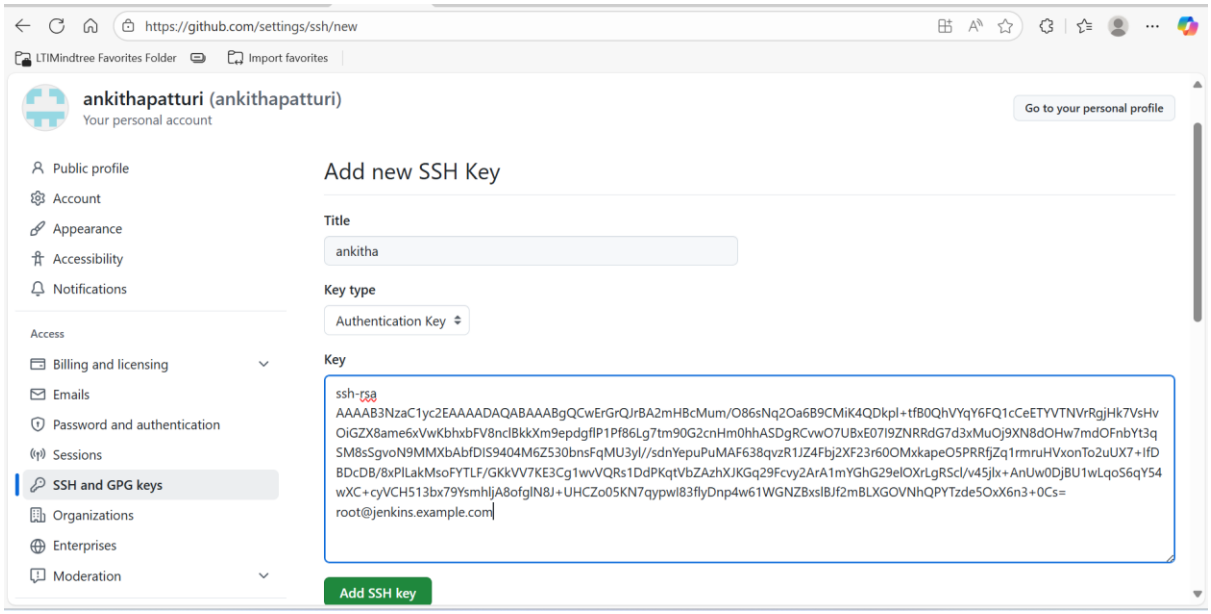
```
Connection to ec2-54-167-5-171.compute-1.amazonaws.com closed.
PS C:\Users\10844392\Downloads> ssh -i "project-key.pem" ec2-user@ec2-54-167-5-171.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Oct 17 10:08:10 2025 from 103.244.155.144
[ec2-user@jenkins ~]$ sudo su -
Last login: Fri Oct 17 10:08:13 UTC 2025 on pts/1
[root@jenkins ~]# git clone https://github.com/sanjayguruji/shourjo-10743365.git
Cloning into 'shourjo-10743365'...
remote: Enumerating objects: 84, done.
remote: Counting objects: 100% (84/84), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 84 (delta 17), reused 29 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (84/84), 15.96 KiB | 7.98 MiB/s, done.
Resolving deltas: 100% (17/17), done.
[root@jenkins ~]#
```

Create an Ec2 instance and install git in the terminal.

Cloning the repo and pushing the repo into new repository should be done as a part of this project.

```
[root@jenkins shourjo-10743365]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:N504GKdQ/LLBkt6nPbY2U5GXFNDhEVMN9cvqLjTavjA root@jenkins.example.com
The key's randomart image is:
+---[RSA 3072]----+
|       ..   .oBB+|
|       ..    .ooo|
|      .o...  o...|
|      o.+=.oo.+ .|
|     . oS+= oo o |
|      . o..=. .  |
|         E+...   |
|        ..Xo.    |
|         ooB+o   |
+----[SHA256]-----+
[root@jenkins shourjo-10743365]# cat /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCwErGrQJrBA2mHBcMum/O86sNq2Oa6B9CMiK4QDkpl+tfB0QhVYqY6FQ1cCeETYVTNVrRgjHk7VsHvOiGZX8ame6xVwKbhx
bFV8nclBkkXm9epdgflP1Pf86Lg7tm90G2cnHm0hhASDgRCvwO7UBxE07I9ZNRRdG7d3xMuOj9XN8dOHw7mdOFnbYt3qSM8sSgvoN9MMXbAbfDIS9404M6Z530bnsFqMU3yl/
/sdnYepuPuMAF638qvzR1JZ4Fbj2XF23r60OMxkapeO5PRRfjZq1rmruHVxonTo2uUX7+IfDBDcDB/8xPlLakMsoFYTLF/GKkVV7KE3Cg1wvVQRs1DdPKqtVbZAzhXJKGq29F
cvy2ArA1mYGhG29elOXrLgRScl/v45jlx+AnUw0DjBU1wLqoS6qY54wXC+cyVCH513bx79YsmhljA8ofglN8J+UHCZo05KN7qypwI83flyDnp4w61WGNZBxslBJf2mBLXGOVN
hQPYTzde5OxX6n3+0Cs= root@jenkins.example.com
[root@jenkins shourjo-10743365]# git add .
[root@jenkins shourjo-10743365]# git push origin main --force
fatal: 'git@github.com' does not appear to be a git repository
fatal: Could not read from remote repository.
```

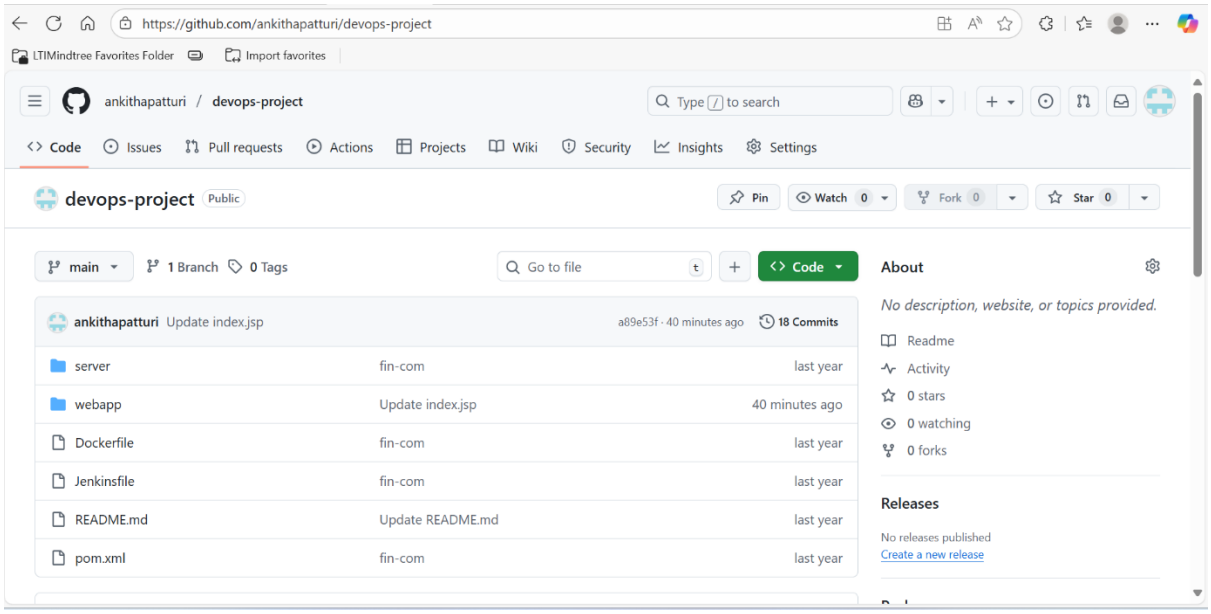
Create and generate SSH and add this key into repository to establish connection.



SSH key is generated for connection with GitHub for pushing it my GitHub.

```
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
[root@jenkins shourjo-10743365]# git init
Reinitialized existing Git repository in /root/shourjo-10743365/.git/
[root@jenkins shourjo-10743365]# git remote add origin git@github.com:ankithapatturi/devop-project.git
error: remote origin already exists.
[root@jenkins shourjo-10743365]# git push -u origin main
fatal: 'git@github.com' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
[root@jenkins shourjo-10743365]# git remote remove origin
[root@jenkins shourjo-10743365]#  git remote add origin git@github.com:ankithapatturi/devop-project.git
[root@jenkins shourjo-10743365]#  git push -u origin main
The authenticity of host 'github.com (140.82.112.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enumerating objects: 84, done.
Counting objects: 100% (84/84), done.
Delta compression using up to 2 threads
Compressing objects: 100% (36/36), done.
Writing objects: 100% (84/84), 15.96 KiB | 15.96 MiB/s, done.
Total 84 (delta 17), reused 84 (delta 17), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (17/17), done.
To github.com:ankithapatturi/devop-project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
[root@jenkins shourjo-10743365]#
```

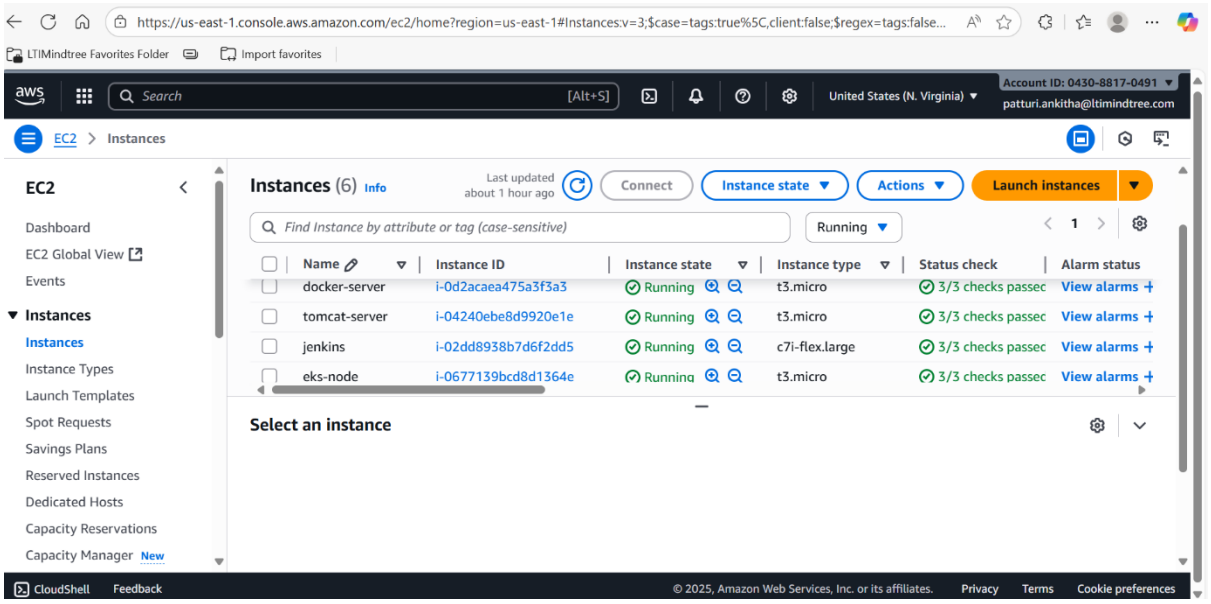
Added some of the commands to push the codes into the git.



All the files are present in my Github Repository.

Create the instances for our project requirements.

Jenkins&Maven:



Created the instances required.

Let’s connect those instances to terminal as per requirements.

Connect Jenkins instance to the terminal to install and enable Jenkins.

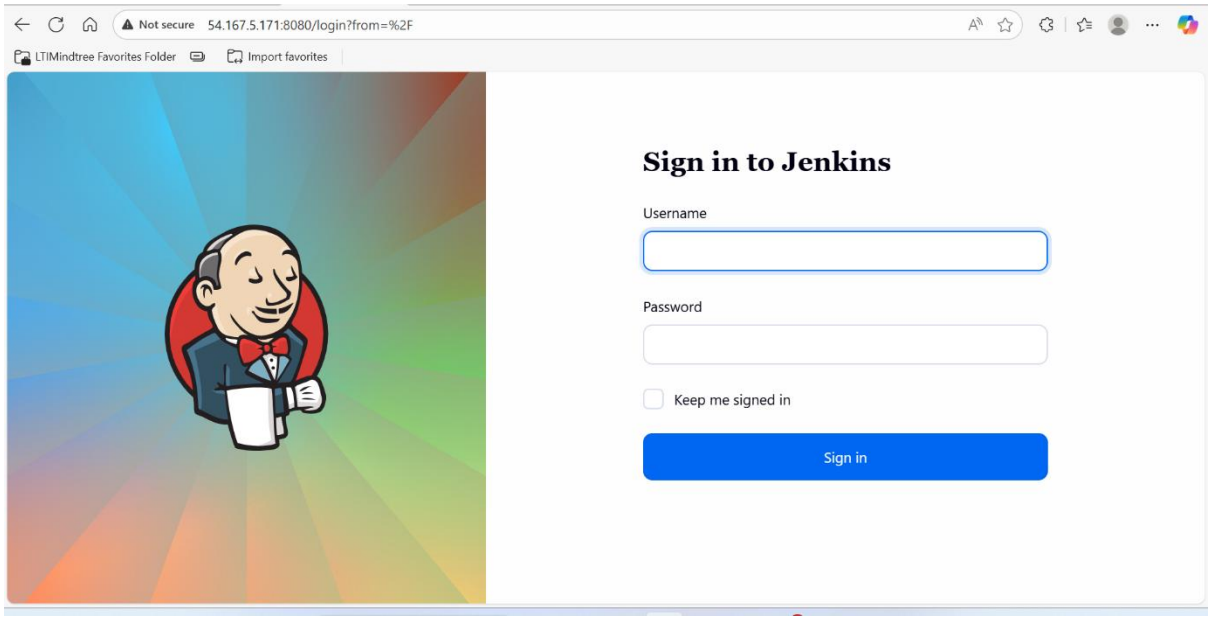
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\10844392\Downloads> ssh -i "project-key.pem" ec2-user@ec2-54-167-5-171.compute-1.amazonaws.com
The authenticity of host 'ec2-54-167-5-171.compute-1.amazonaws.com (54.167.5.171)' can't be established.
ED25519 key fingerprint is SHA256:Uc0fTWEe2540w6U1fu/esLz4Nwtx8yhJRh/wE3ObCH4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-167-5-171.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-16-116 ~]$ sudo su -
[root@ip-172-31-16-116 ~]# hostnamectl set-hostname jenkins.example.com
[root@ip-172-31-16-116 ~]# bash
[root@jenkins ~]# sudo yum update -y
Last metadata expiration check: 0:00:42 ago on Fri Oct 17 03:42:07 2025.
Dependencies resolved.
Nothing to do.
Complete!
[root@jenkins ~]# sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2025-10-17 03:43:29--  https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.38.133, 2a04:4e42:78::645
```

Install Jenkins in the terminal to start and enable Jenkins.

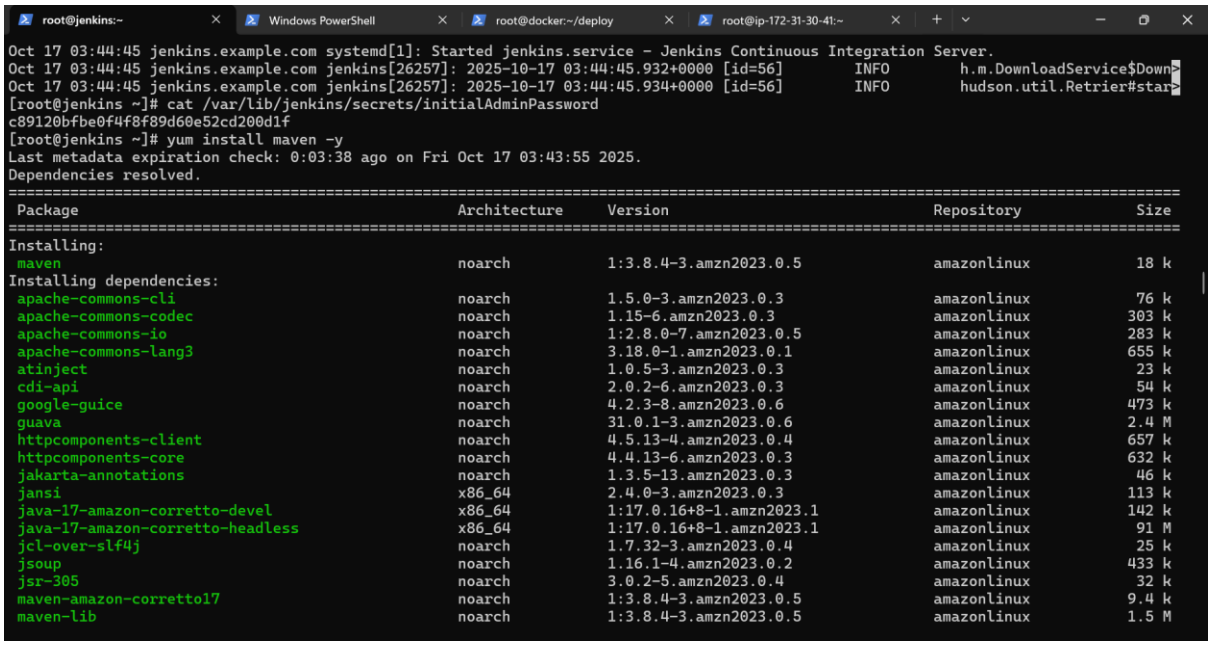
After installation of Jenkins paste the public id of Jenkins along with port 8080 to start Jenkins in the browser.

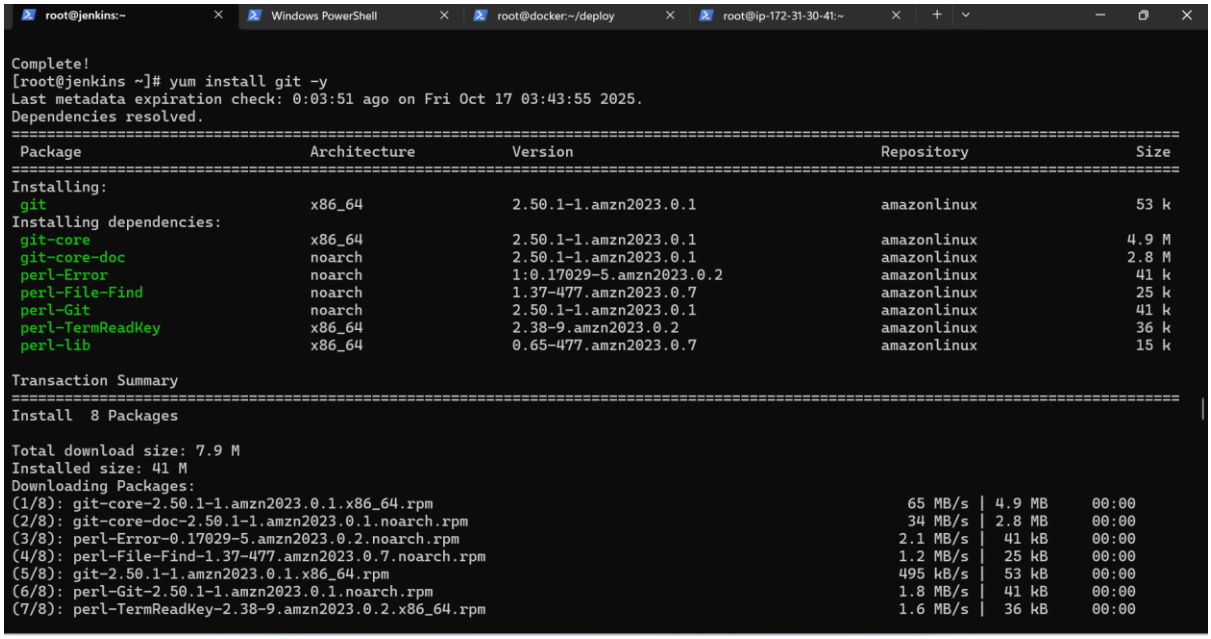
Generate password of Jenkins by copying the path of Jenkins in that page.

Sign-in to Jenkins by giving some credentials like username ,password and email, etc.

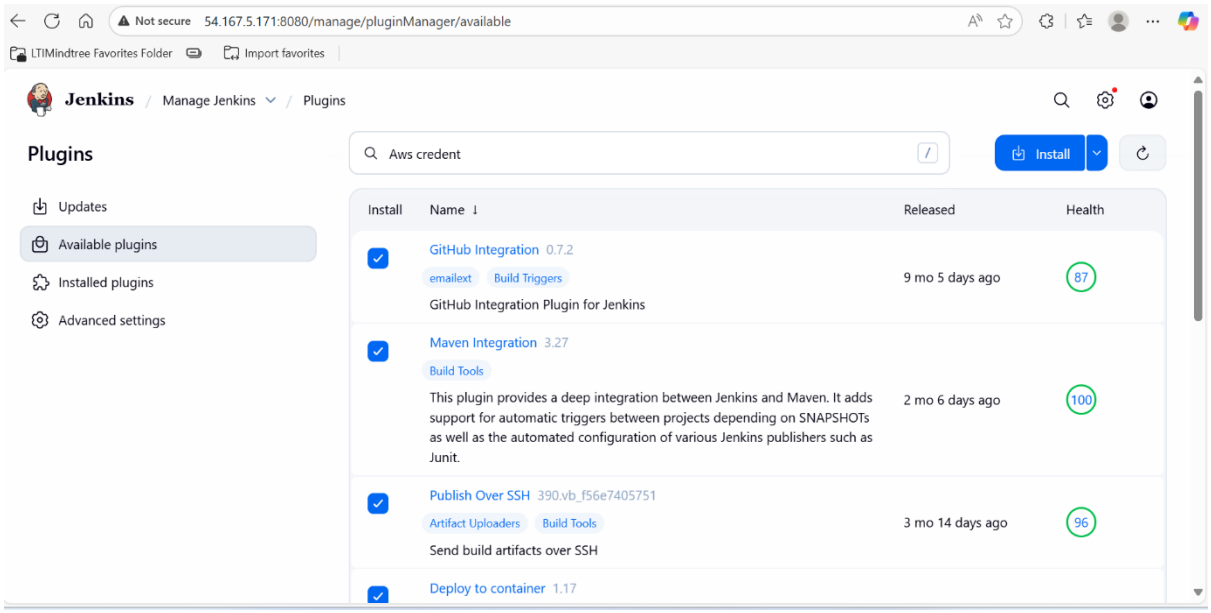


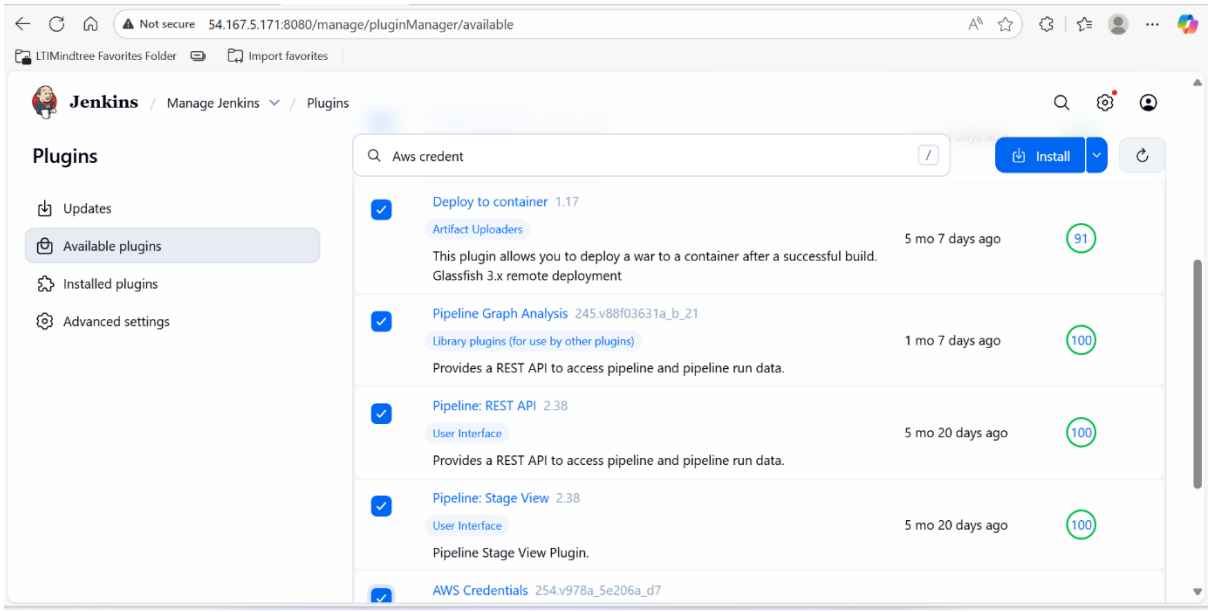
Install maven and git in Jenkins terminal.

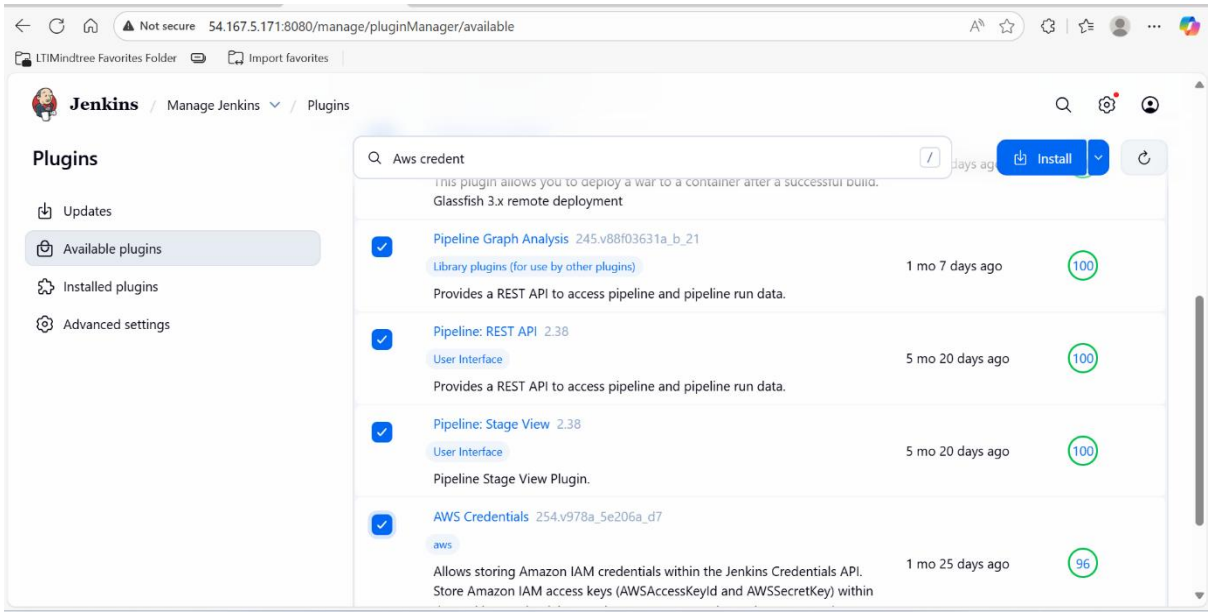


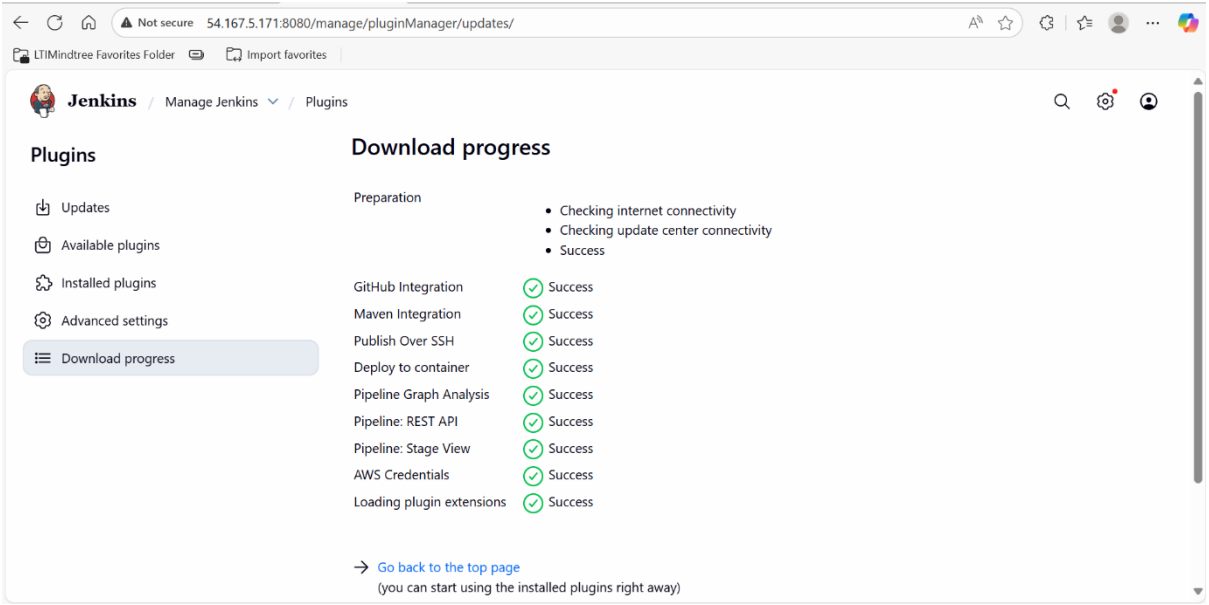


Add required plugins in tools of the Jenkins.



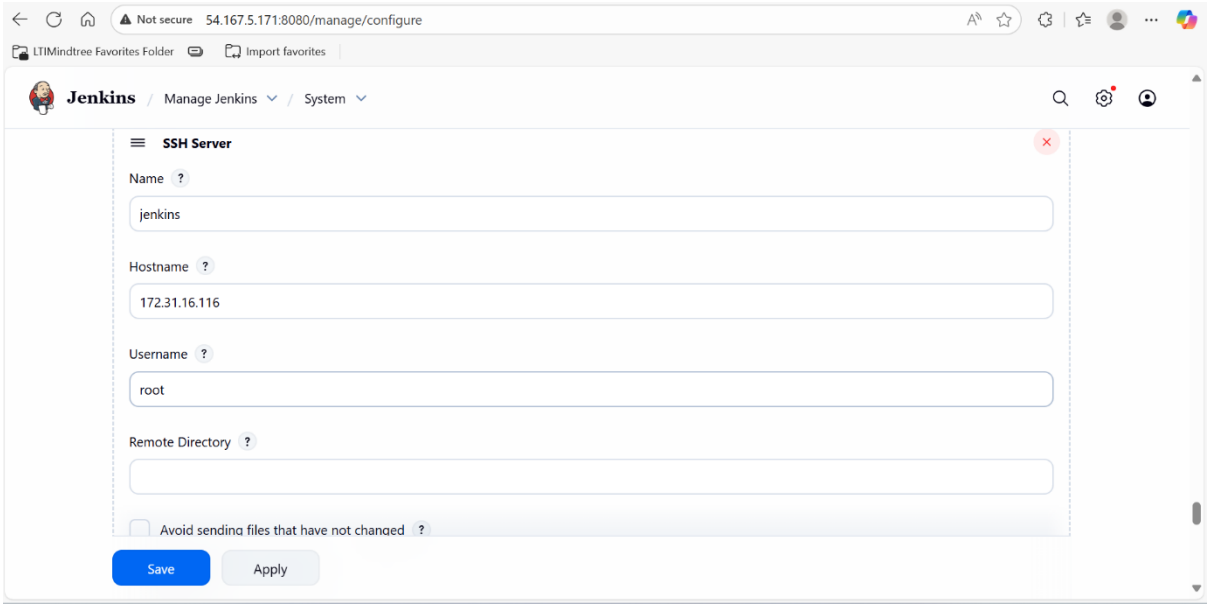

Not secure 54.167.5.171:8080/manage/pluginManager/available
LTIMindtree Favorites Folder
Import favorites
Jenkins / Manage Jenkins / Plugins
Plugins
Updates
Available plugins
Installed plugins
Advanced settings
Aws credent
Install
Deploy to container 1.17
Artifact Uploaders
This plugin allows you to deploy a war to a container after a successful build.
Glassfish 3.x remote deployment
5 mo 7 days ago
91
Pipeline Graph Analysis 245.v88f03631a_b_21
Library plugins (for use by other plugins)
Provides a REST API to access pipeline and pipeline run data.
1 mo 7 days ago
100
Pipeline: REST API 2.38
User Interface
Provides a REST API to access pipeline and pipeline run data.
5 mo 20 days ago
100
Pipeline: Stage View 2.38
User Interface
Pipeline Stage View Plugin.
5 mo 20 days ago
100
AWS Credentials 254.v978a_5e206a_d7


Not secure 54.167.5.171:8080/manage/pluginManager/available
LTIMindtree Favorites Folder
Import favorites
Jenkins / Manage Jenkins / Plugins
Plugins
Updates
Available plugins
Installed plugins
Advanced settings
Aws credent
Install
Glassfish 3.x remote deployment
Pipeline Graph Analysis 245.v88f03631a_b_21
Library plugins (for use by other plugins)
Provides a REST API to access pipeline and pipeline run data.
1 mo 7 days ago
100
Pipeline: REST API 2.38
User Interface
Provides a REST API to access pipeline and pipeline run data.
5 mo 20 days ago
100
Pipeline: Stage View 2.38
User Interface
Pipeline Stage View Plugin.
5 mo 20 days ago
100
AWS Credentials 254.v978a_5e206a_d7
aws
Allows storing Amazon IAM credentials within the Jenkins Credentials API.
Store Amazon IAM access keys (AWSAccessKeyId and AWSSecretKey) within
1 mo 25 days ago
96

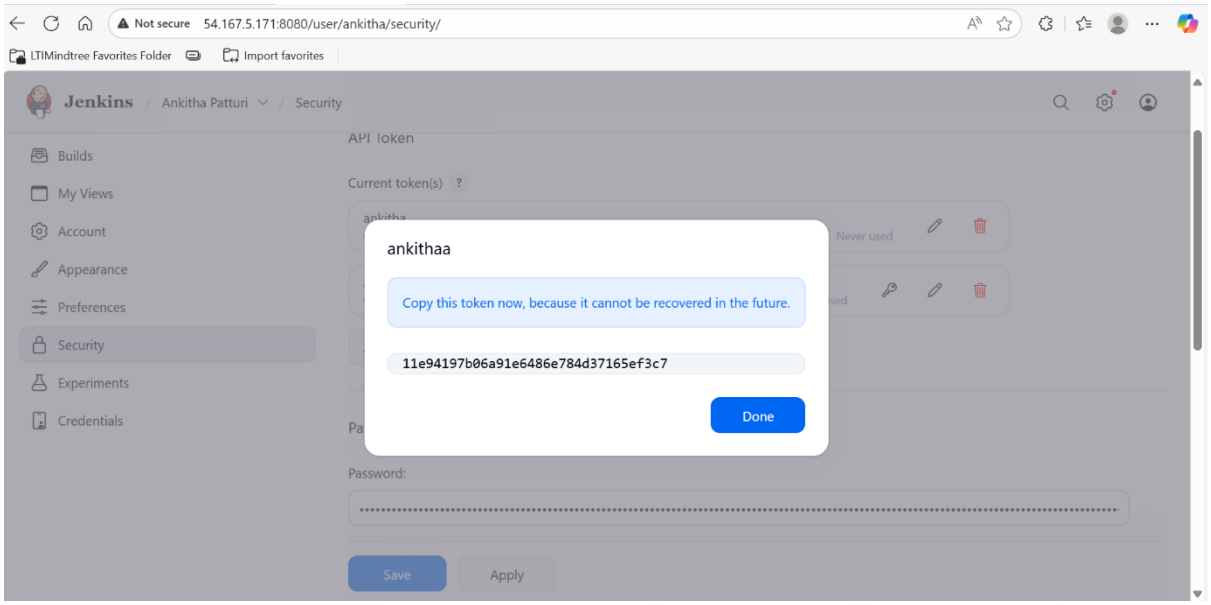


Plugins are necessary to add support for source code, build tools and deployment.

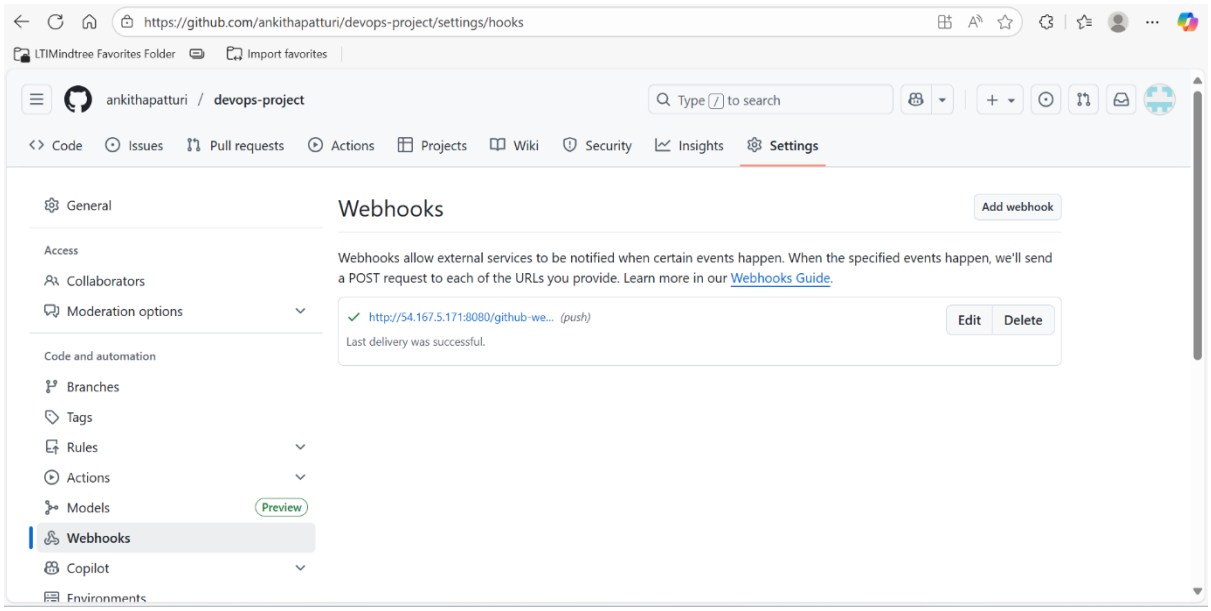
We need to add Jenkins data in SSH server of Jenkins.



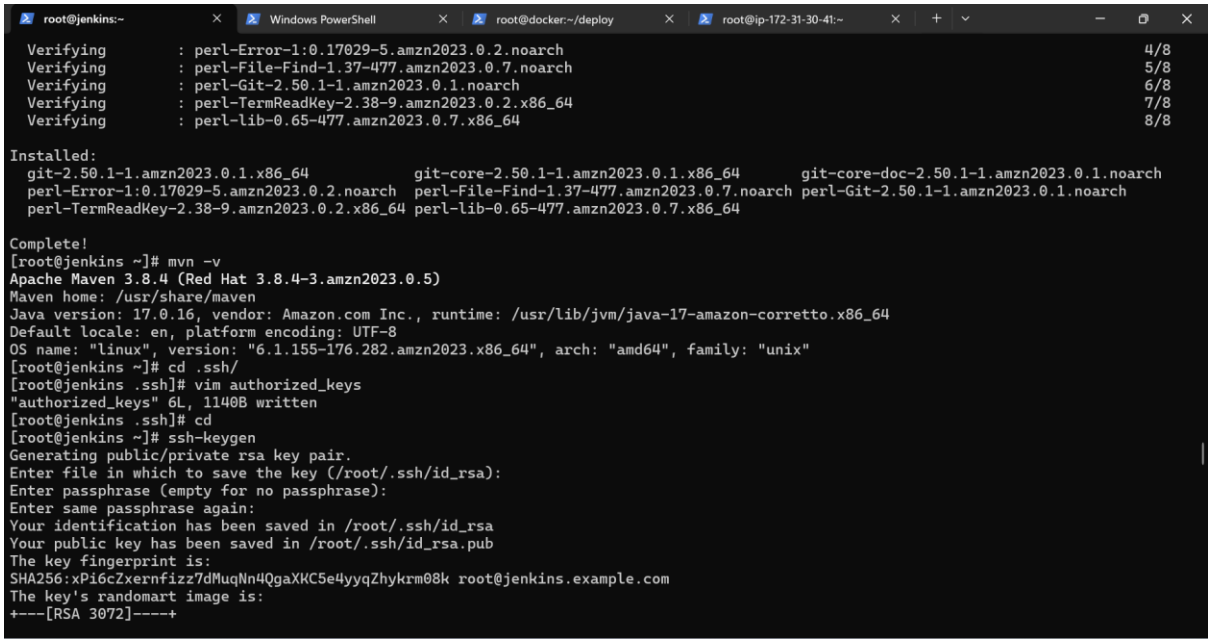
Add a webhook in github and for adding webhook we need to generate API token in Jenkins.

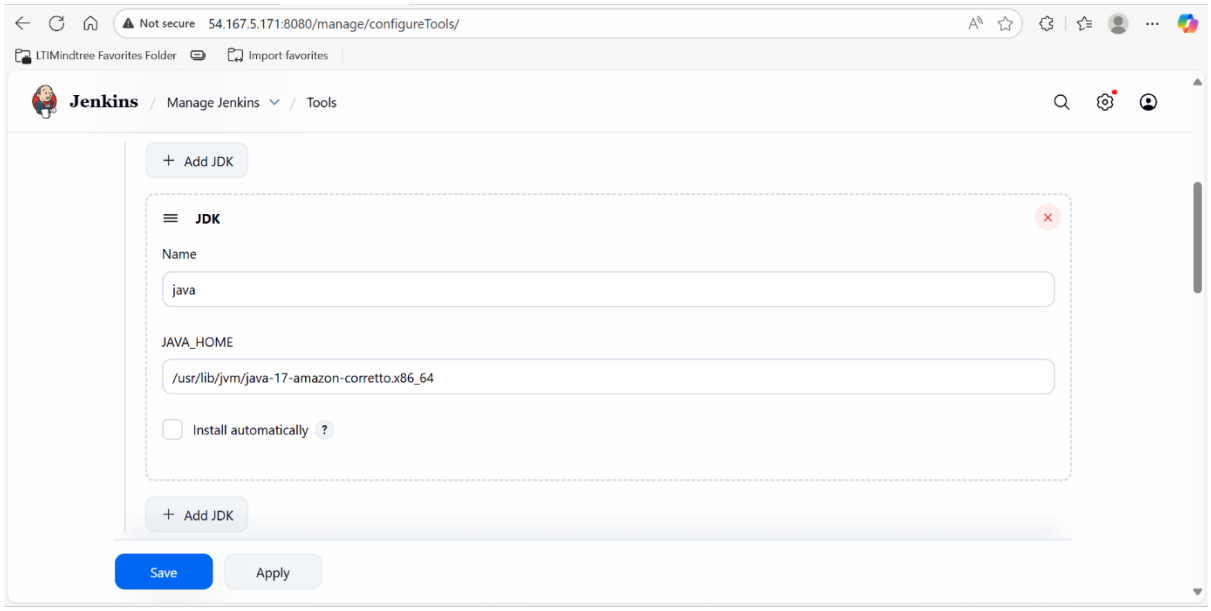


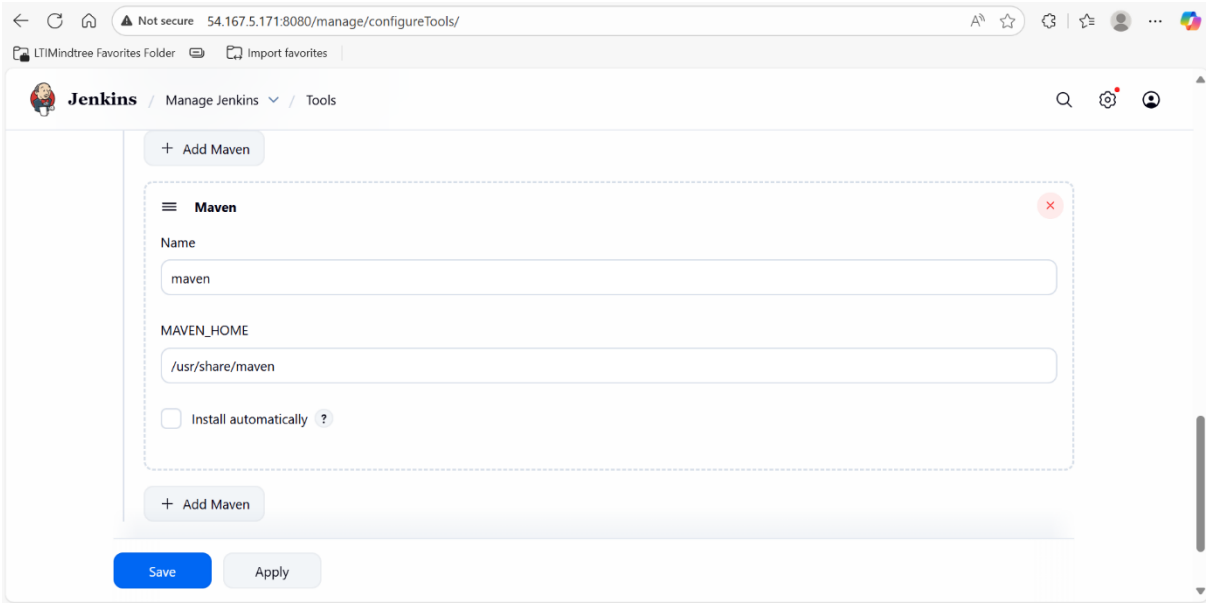
We need to add Jenkins in SSH server of Jenkins.



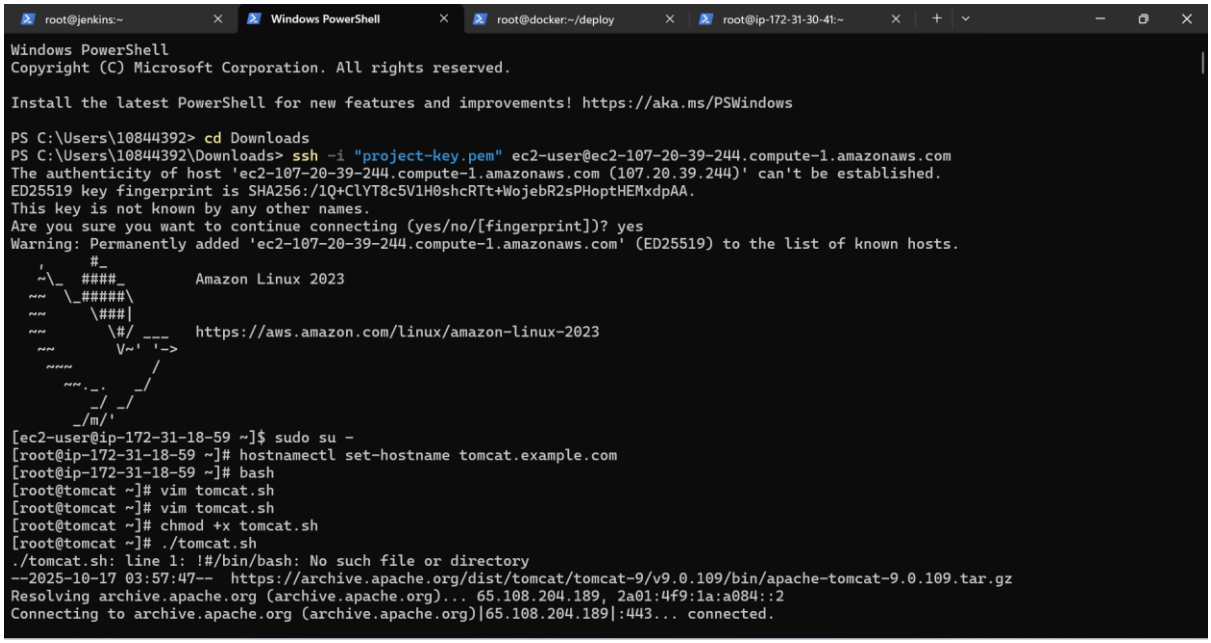
Add Java and Maven paths in tools of Jenkins.


root@jenkins:~
Windows PowerShell
root@docker:~/deploy
root@ip-172-31-30-41:~
Verifying : perl-Error-1:0.17029-5.amzn2023.0.2.noarch 4/8
Verifying : perl-File-Find-1.37-477.amzn2023.0.7.noarch 5/8
Verifying : perl-Git-2.50.1-1.amzn2023.0.1.noarch 6/8
Verifying : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 7/8
Verifying : perl-lib-0.65-477.amzn2023.0.7.x86_64 8/8
Installed:
git-2.50.1-1.amzn2023.0.1.x86_64 git-core-2.50.1-1.amzn2023.0.1.x86_64 git-core-doc-2.50.1-1.amzn2023.0.1.noarch
perl-Error-1:0.17029-5.amzn2023.0.2.noarch perl-File-Find-1.37-477.amzn2023.0.7.noarch perl-Git-2.50.1-1.amzn2023.0.1.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 perl-lib-0.65-477.amzn2023.0.7.x86_64
Complete!
[root@jenkins ~]# mvn -v
Apache Maven 3.8.4 (Red Hat 3.8.4-3.amzn2023.0.5)
Maven home: /usr/share/maven
Java version: 17.0.16, vendor: Amazon.com Inc., runtime: /usr/lib/jvm/java-17-amazon-corretto.x86_64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.1.155-176.282.amzn2023.x86_64", arch: "amd64", family: "unix"
[root@jenkins ~]# cd .ssh/
[root@jenkins .ssh]# vim authorized_keys
"authorized_keys" 6L, 1140B written
[root@jenkins .ssh]# cd
[root@jenkins ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:xPi6cZxernfizz7dMuqNn4QgaXKC5e4yyqZhykrm08k root@jenkins.example.com
The key's randomart image is:
+---[RSA 3072]----+


Not secure 54.167.5.171:8080/manage/configureTools/
LTIMindtree Favorites Folder
Import favorites
Jenkins / Manage Jenkins / Tools
Add JDK
JDK
Name
java
JAVA_HOME
/usr/lib/jvm/java-17-amazon-corretto.x86_64
Install automatically
Add JDK
Save
Apply



Now connect Tomcat instance to terminal.



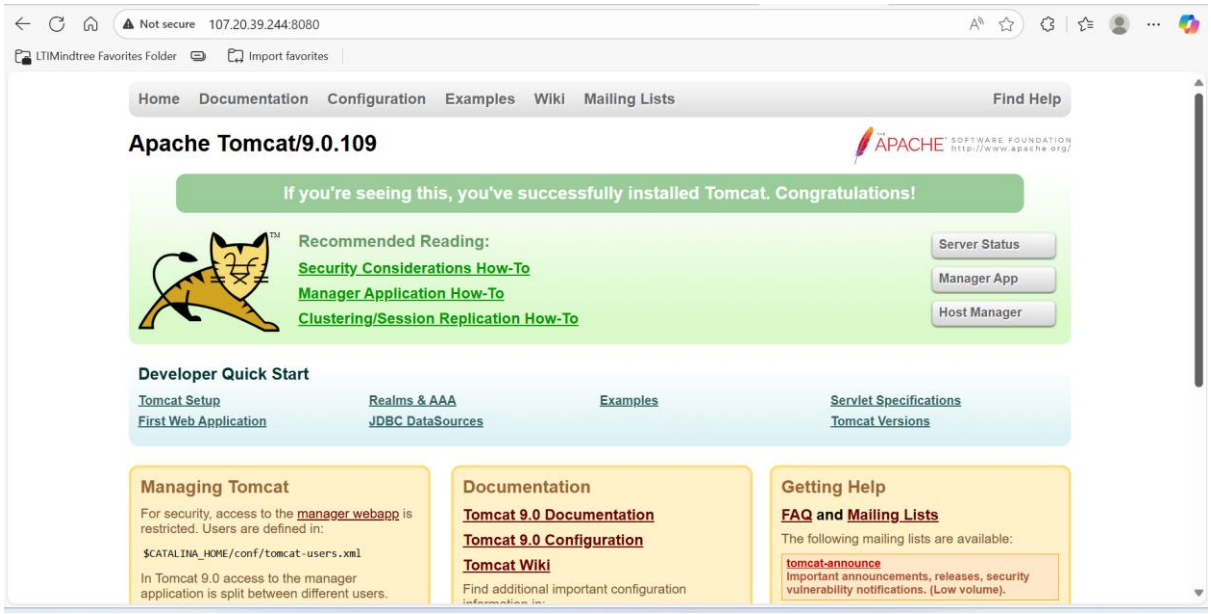
Install Tomcat by giving required commands.

Add manager roles in tomcat terminal.

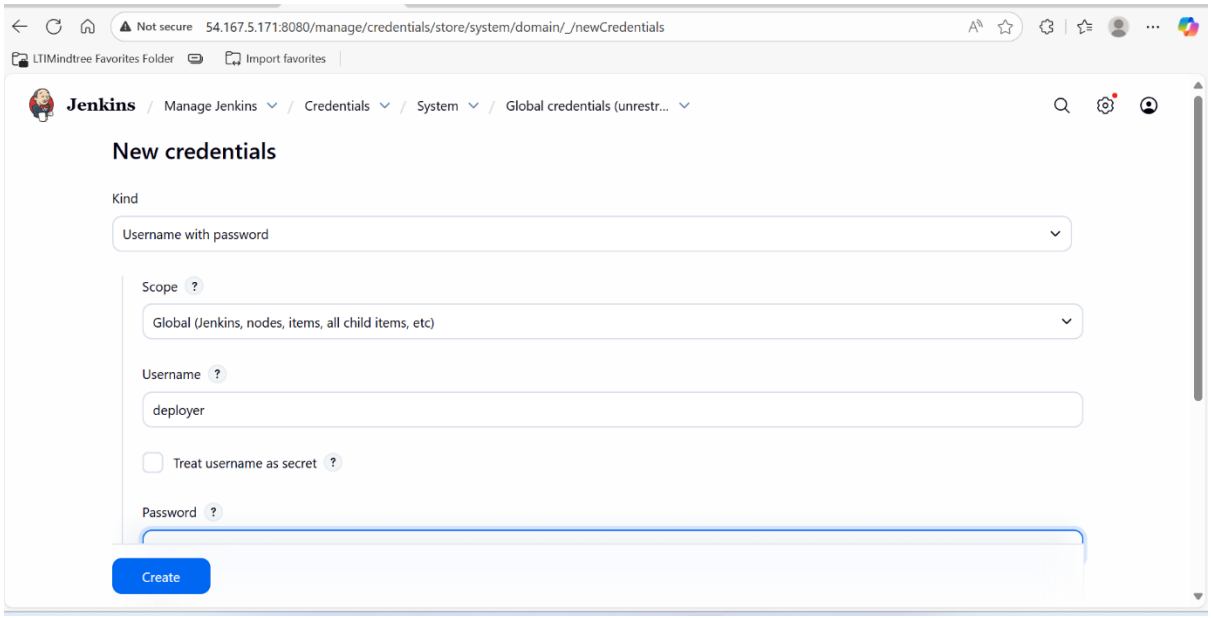

```
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<role rolename="manager-gui"/>
<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="tomcat" password="s3cret" roles="manager-gui"/>
</tomcat-users>
-- INSERT --                                                    62,64         Bot
```

```
root@jenkins:~ | root@tomcat:~ | root@docker:~/deploy | root@ip-172-31-30-41:~

!#/bin/bash
wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.109/bin/apache-tomcat-9.0.109.tar.gz
tar -xzvf apache-tomcat-9.0.109.tar.gz
yum install java* -y
yum install maven -y
cd apache-tomcat-9.0.109
cd bin
chmod +x startup.sh
chmod +x shutdown.sh
cd ..
cd conf
vim tomcat-users.xml
cd
find -name context.xml
vim ./apache-tomcat-9.0.109/webapps/examples/META-INF/context.xml
vim ./apache-tomcat-9.0.109/webapps/host-manager/META-INF/context.xml
vim ./apache-tomcat-9.0.109/webapps/manager/META-INF/context.xml
cd apache-tomcat-9.0.109
cd bin
./startup.sh
~
~
~
~
~
~
~
~
~
~
~
"tomcat.sh" 20L, 568B                                          20,12         All
```

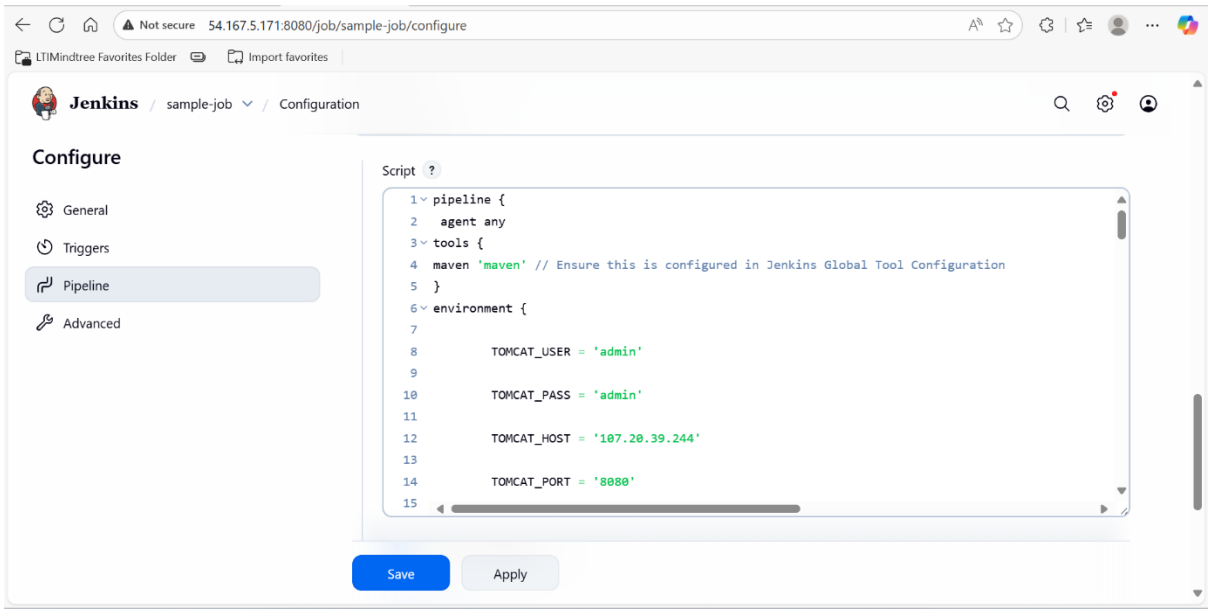
Now paste the public-ip of tomcat server along with port no 8080 in browser.

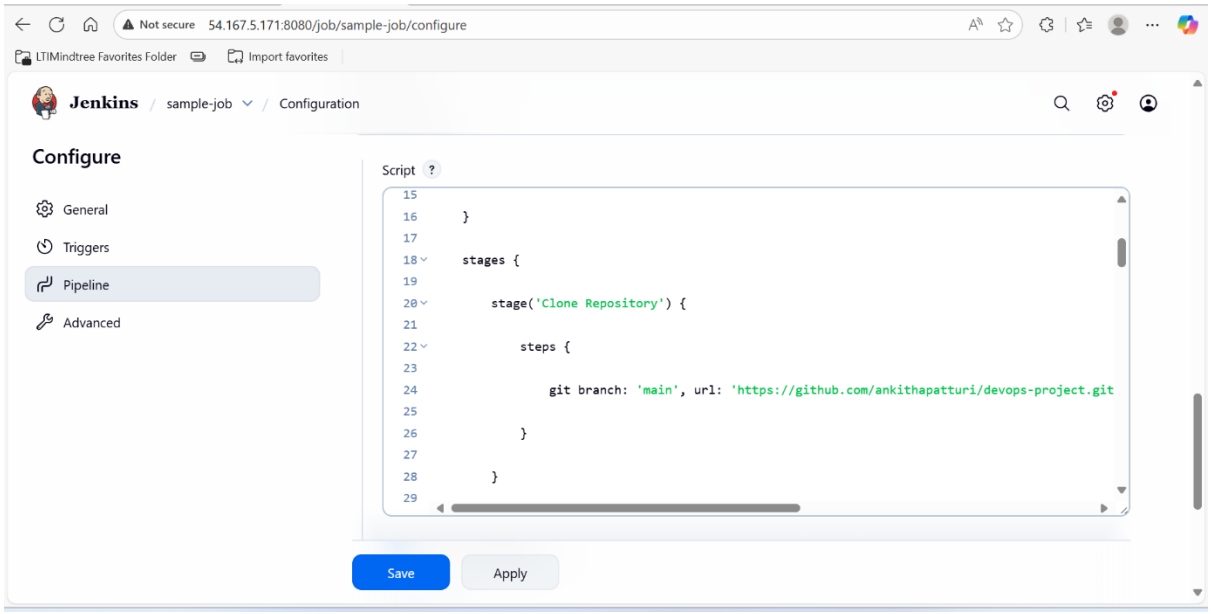


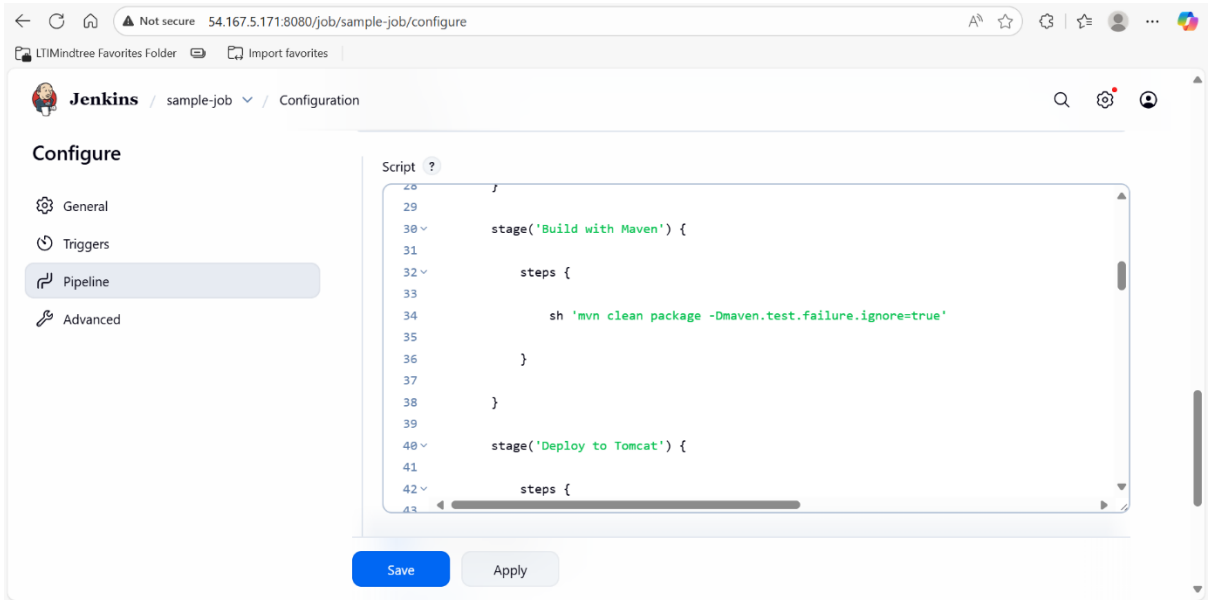
Tomcat is installed successfully and it’s working.

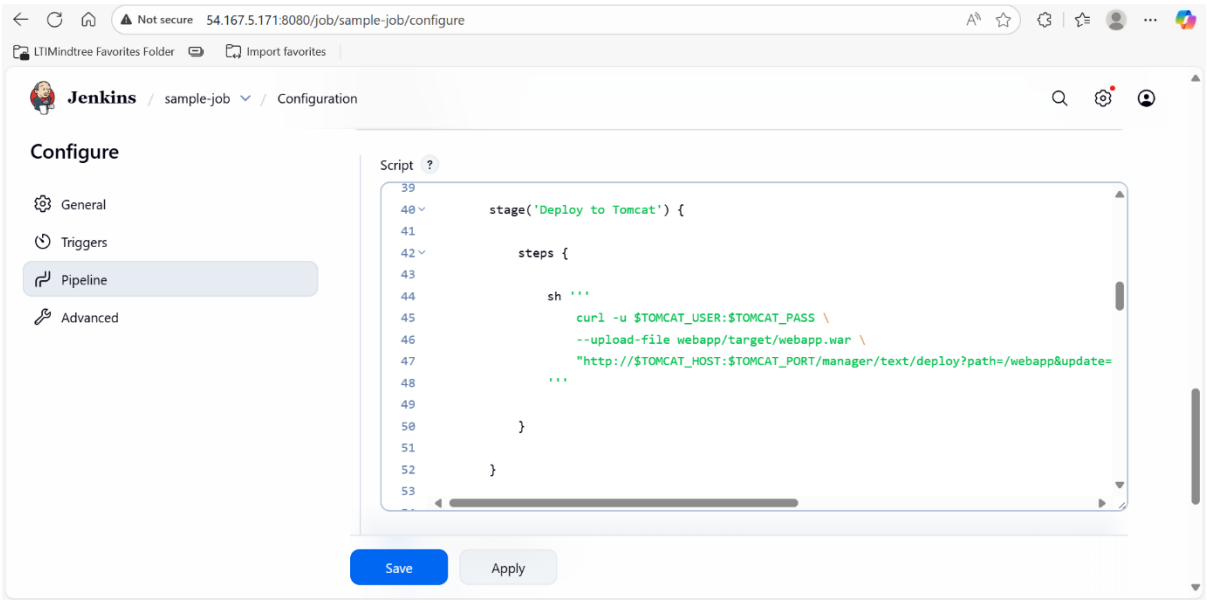


Global credentials has been created for the tomcat server.



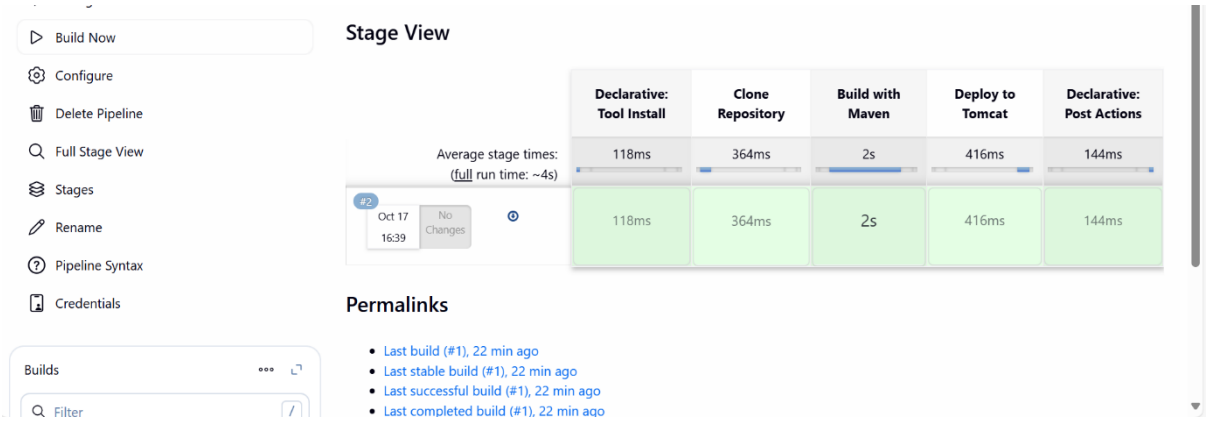

Not secure 54.167.5.171:8080/job/sample-job/configure
LTIMindtree Favorites Folder
Import favorites
Jenkins / sample-job / Configuration
Configure
General
Triggers
Pipeline
Advanced
Script ?
15
16 }
17
18 stages {
19
20 stage('Clone Repository') {
21
22 steps {
23
24 git branch: 'main', url: 'https://github.com/ankithapatturi/devops-project.git
25
26 }
27
28 }
29
Save
Apply


Not secure 54.167.5.171:8080/job/sample-job/configure
LTIMindtree Favorites Folder
Import favorites
Jenkins / sample-job / Configuration
Configure
General
Triggers
Pipeline
Advanced
Script ?
29
30 stage('Build with Maven') {
31
32 steps {
33
34 sh 'mvn clean package -Dmaven.test.failure.ignore=true'
35
36 }
37
38 }
39
40 stage('Deploy to Tomcat') {
41
42 steps {
Save
Apply

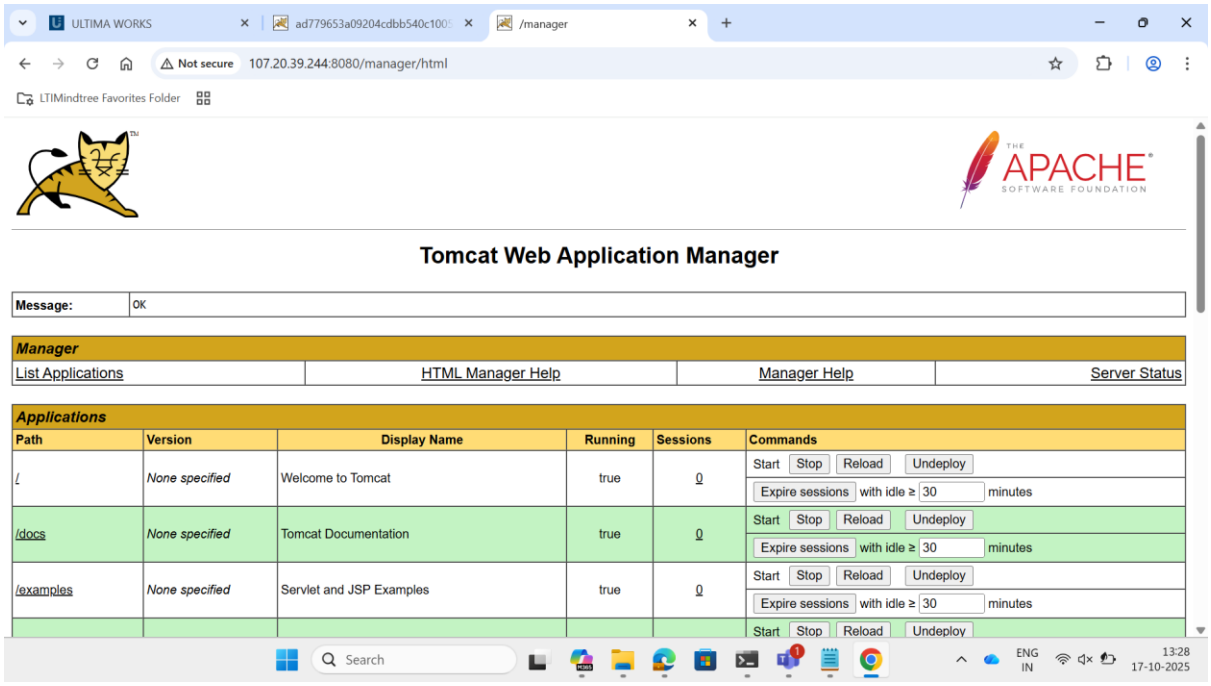


Write a groovy script in pipeline project of Jenkins to see wheather Tomcat server is running.

Build it and test it.



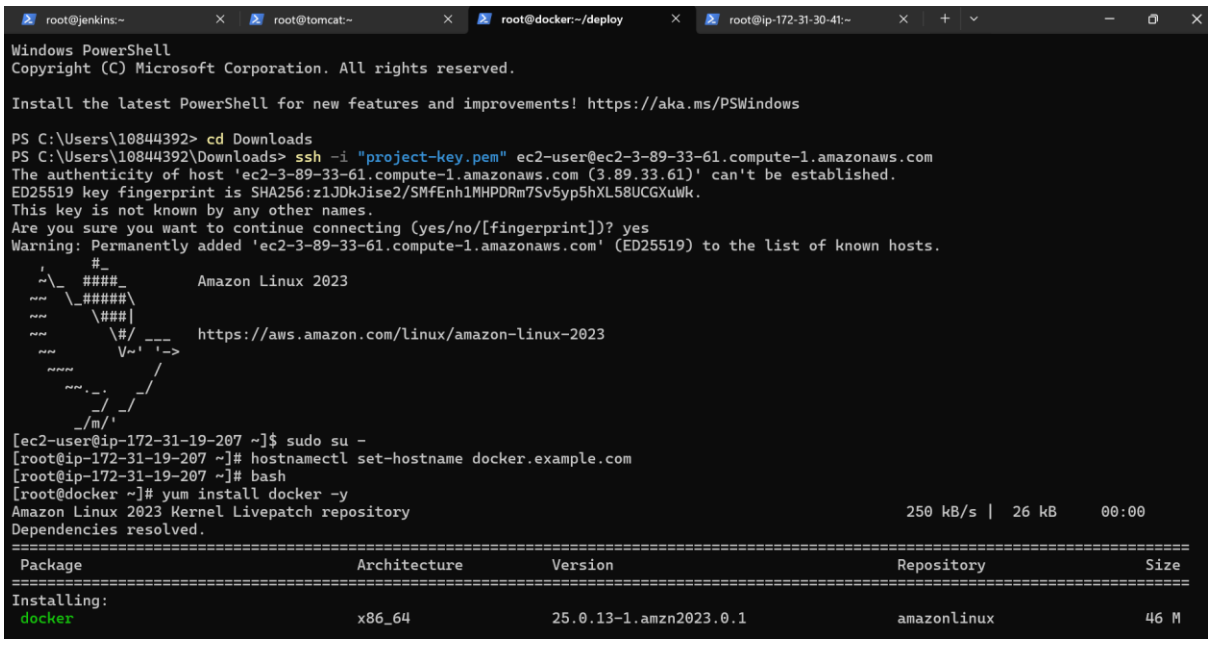
Refresh the browser now to see the changes reflected here.



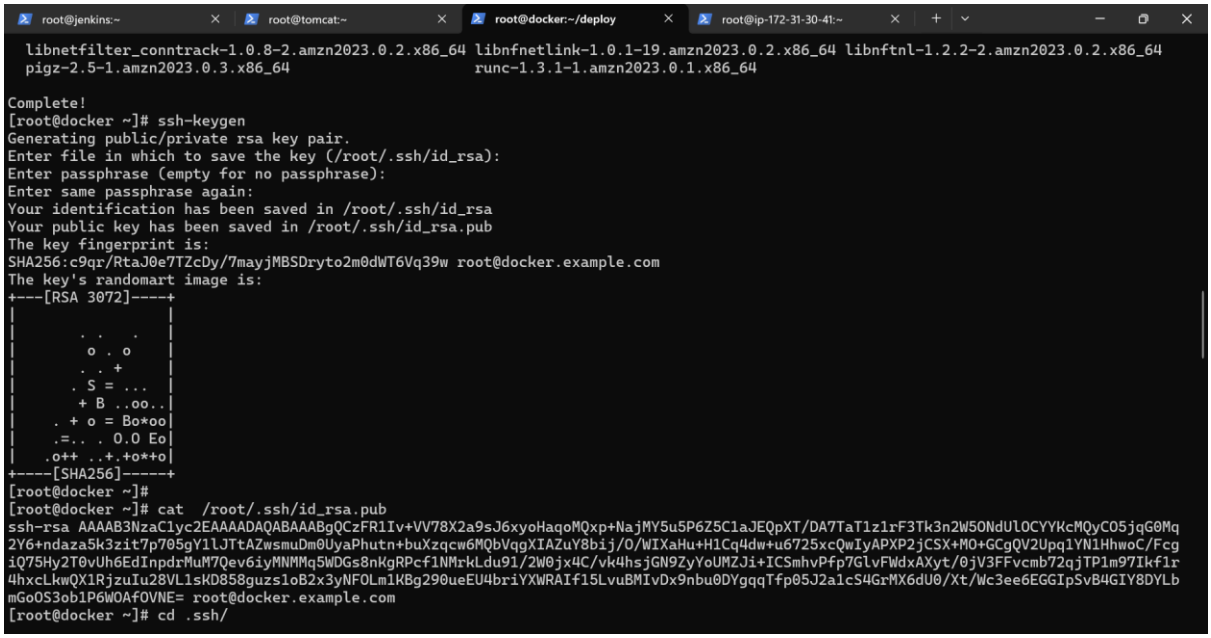
We can able to access the Tomcat page now.

Docker:

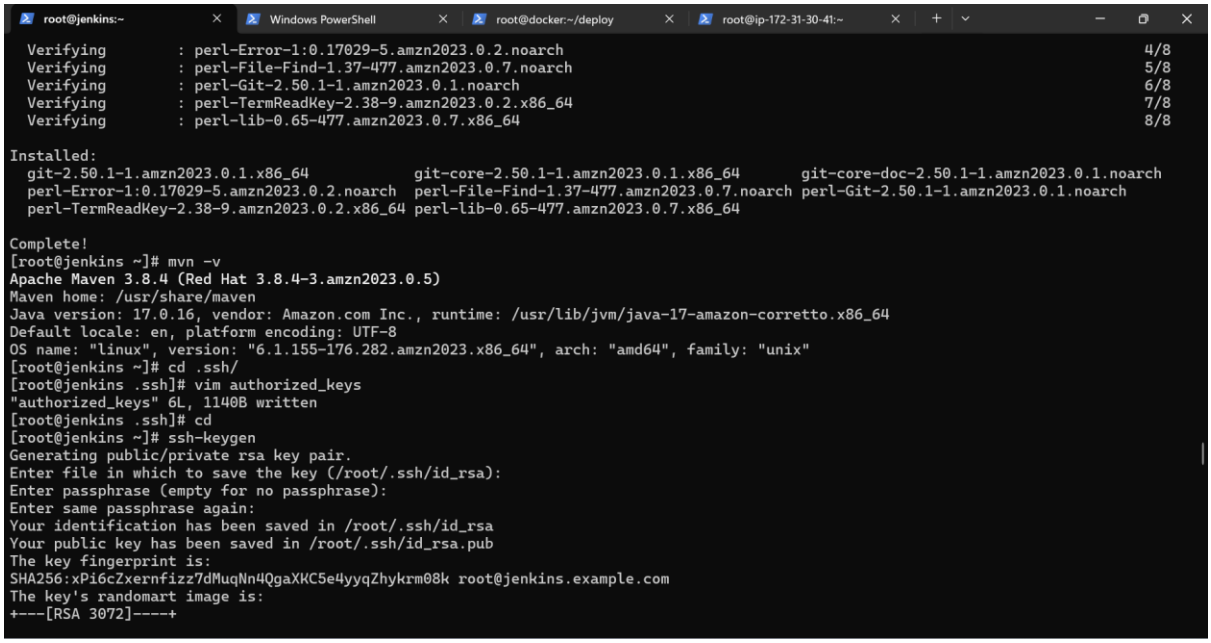
Connect docker instance to the terminal.



Install docker and generate SSH key in docker and Jenkins to establish connection between them.



Exchange SSH keys between the docker and Jenkins.



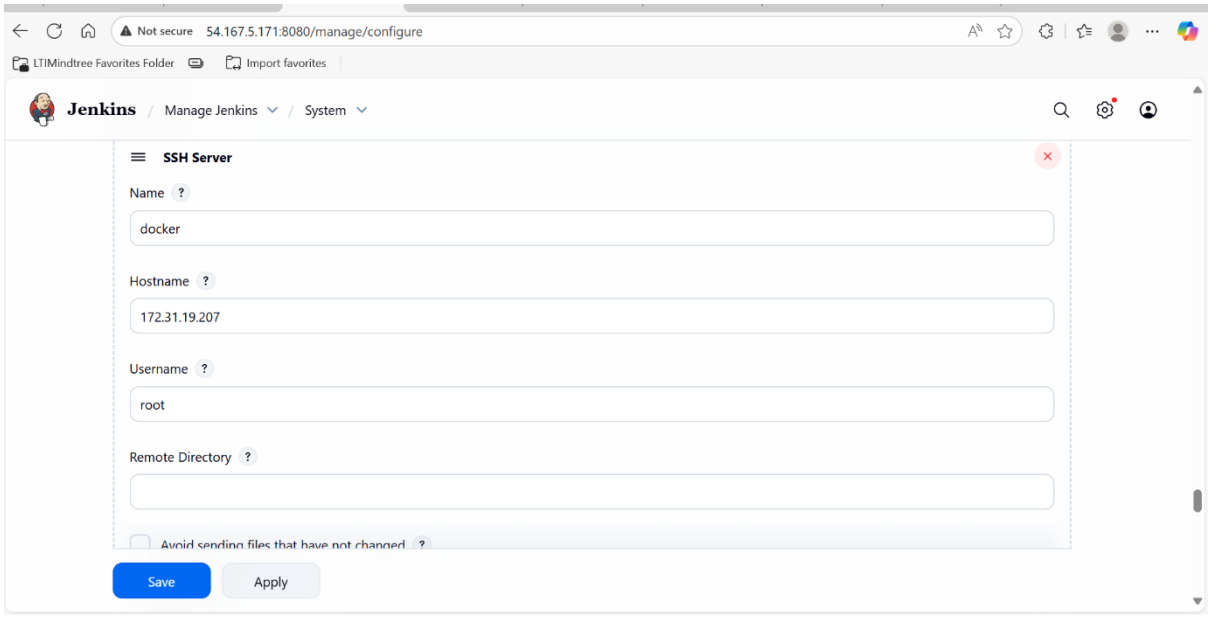
Go to sshd and give some permissions restart and enable sshd in both the terminals.

Enable permissions such as Permit root login and password Authentication in both Jenkins and docker terminal of sshd.

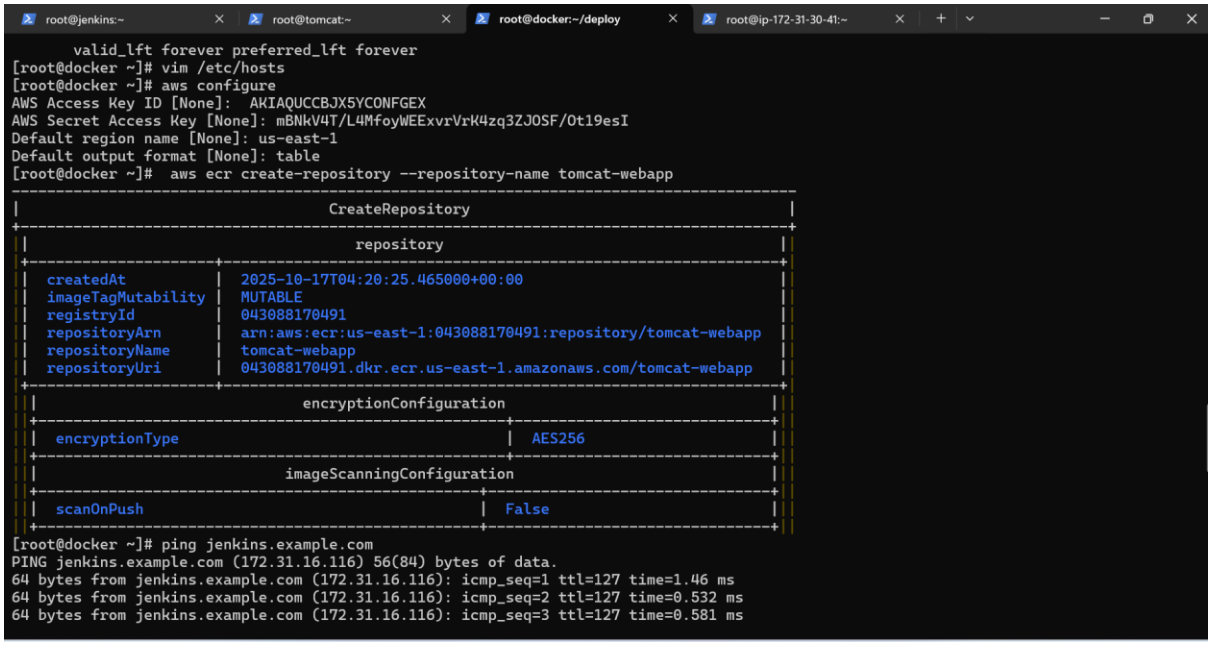
```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```

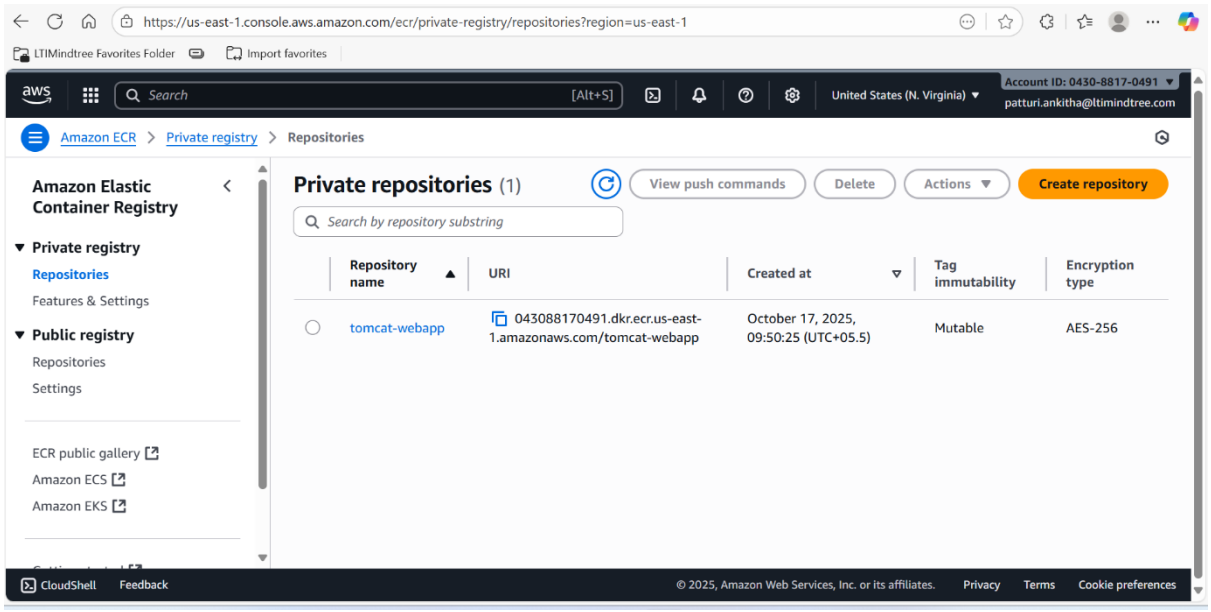
```
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# Explicitly disable PasswordAuthentication. By presetting it, we
# avoid the cloud-init set_passwords module modifying sshd_config and
# restarting sshd in the default instance launch configuration.
PasswordAuthentication yes
PermitEmptyPasswords no
```

We need to add docker data in SSH server of Jenkins.



We need to push a repository into ECR from the terminal.

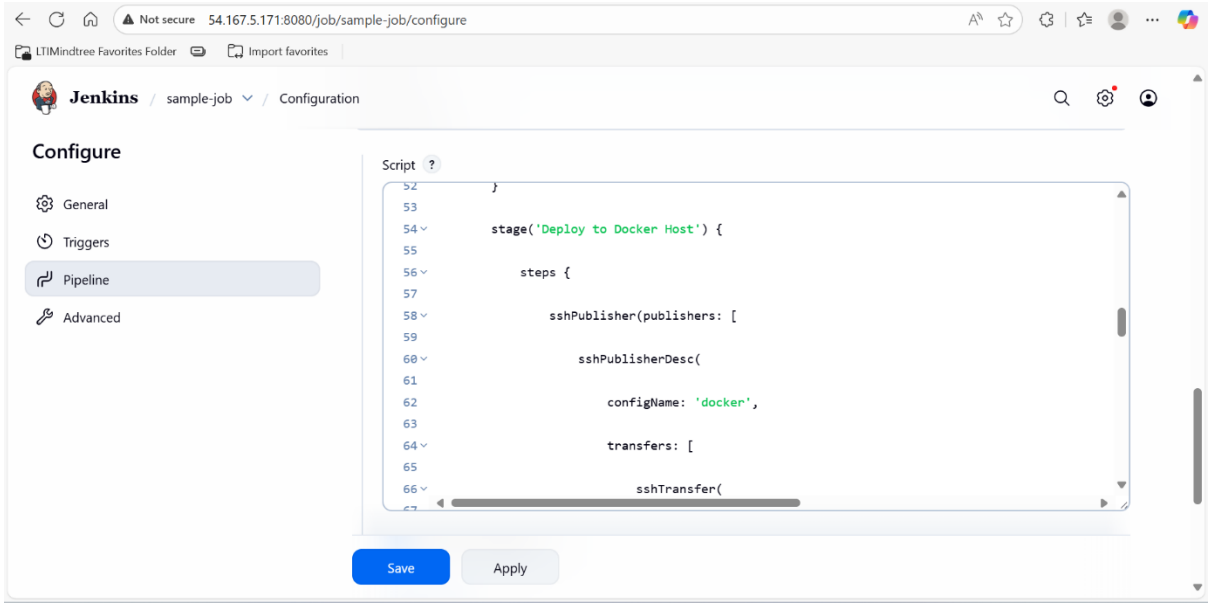


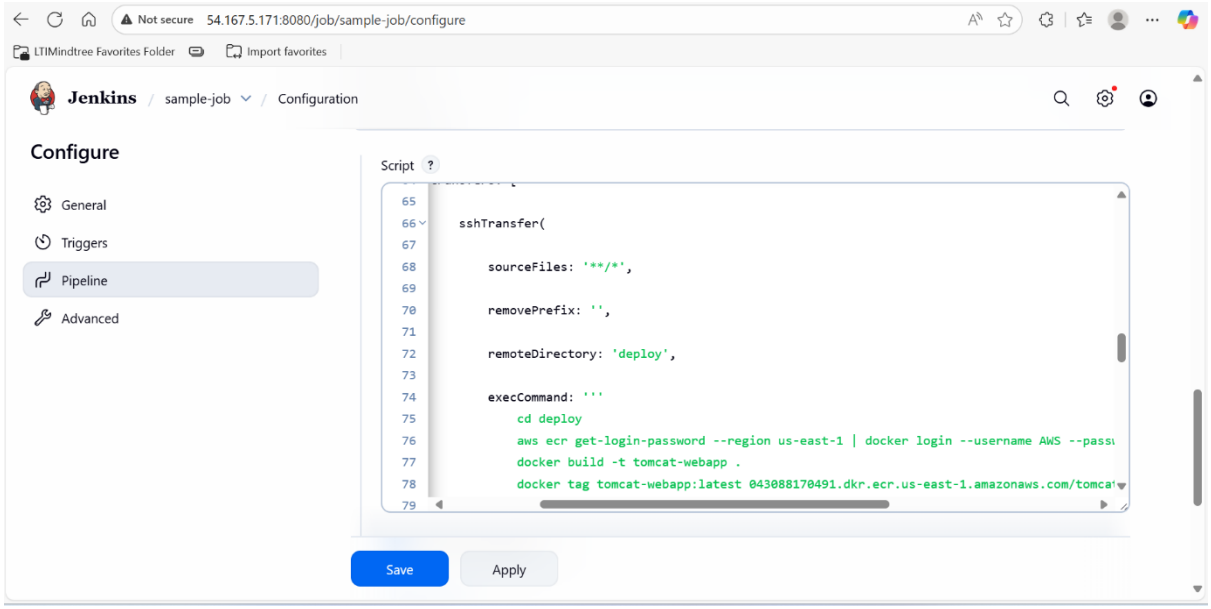


ECR is created. Now we need to push images into ECR

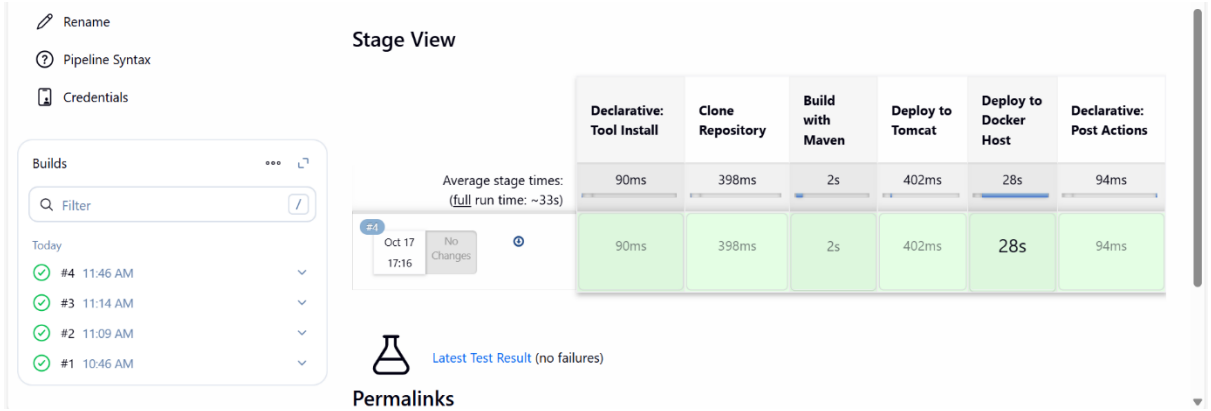
By using declarative approach we can push the images into ECR repository.

Continue the groovy script by adding docker in it.

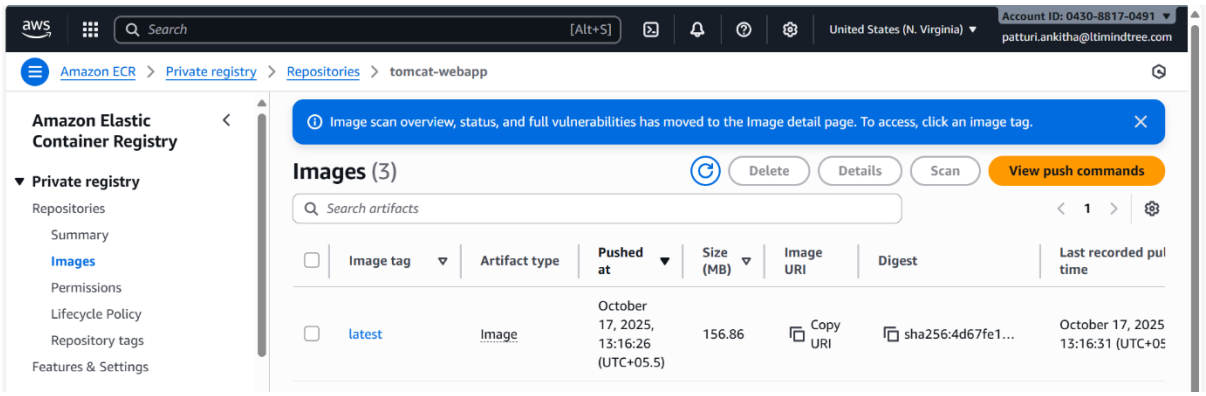




Build the script and see wheather images are pushed into ECR repository.



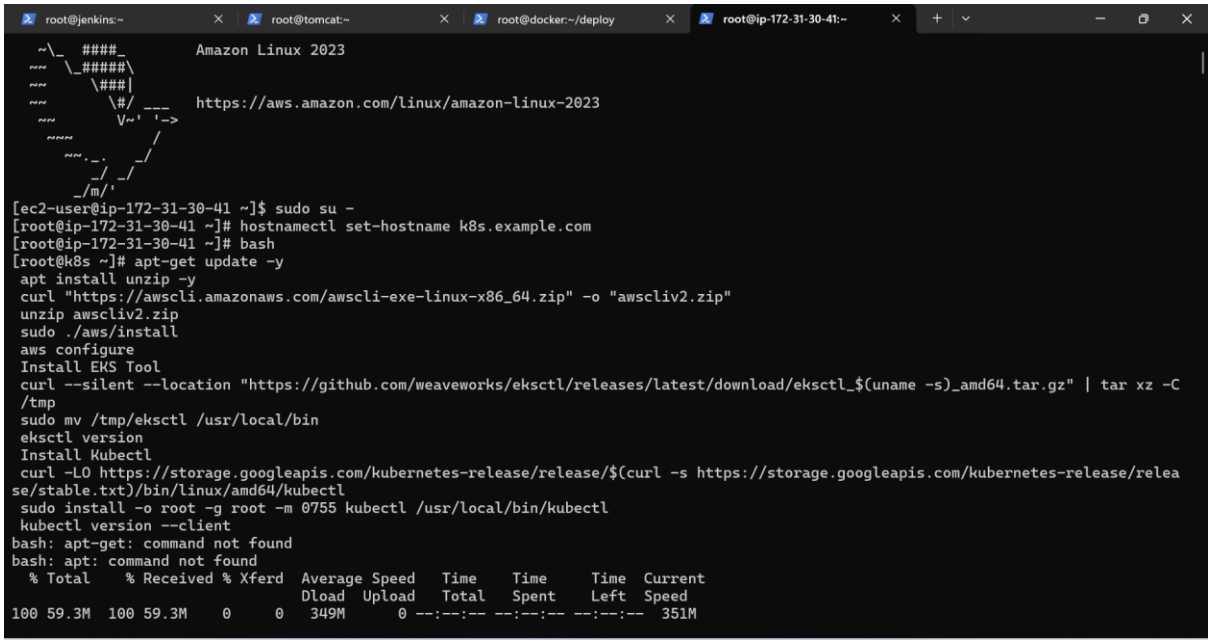
When the build is success we can see the image pushed in ECR repository.



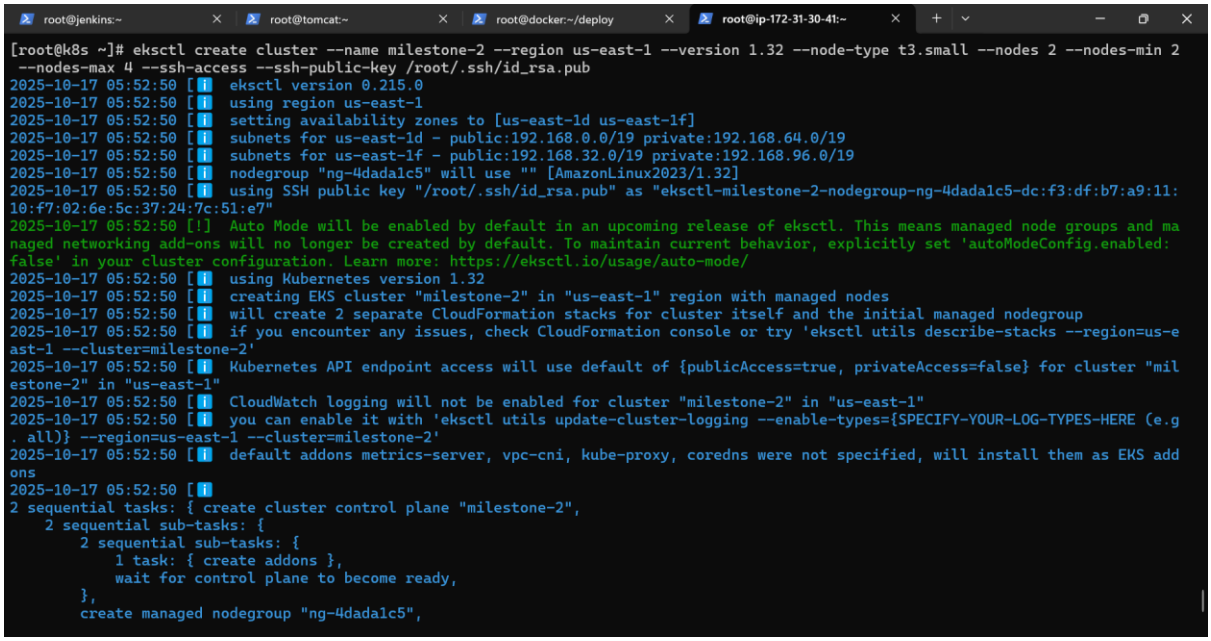
The image is created in the repository of ECR.

Kubernetes EKS:

Connect EKS instance to terminal of k8s

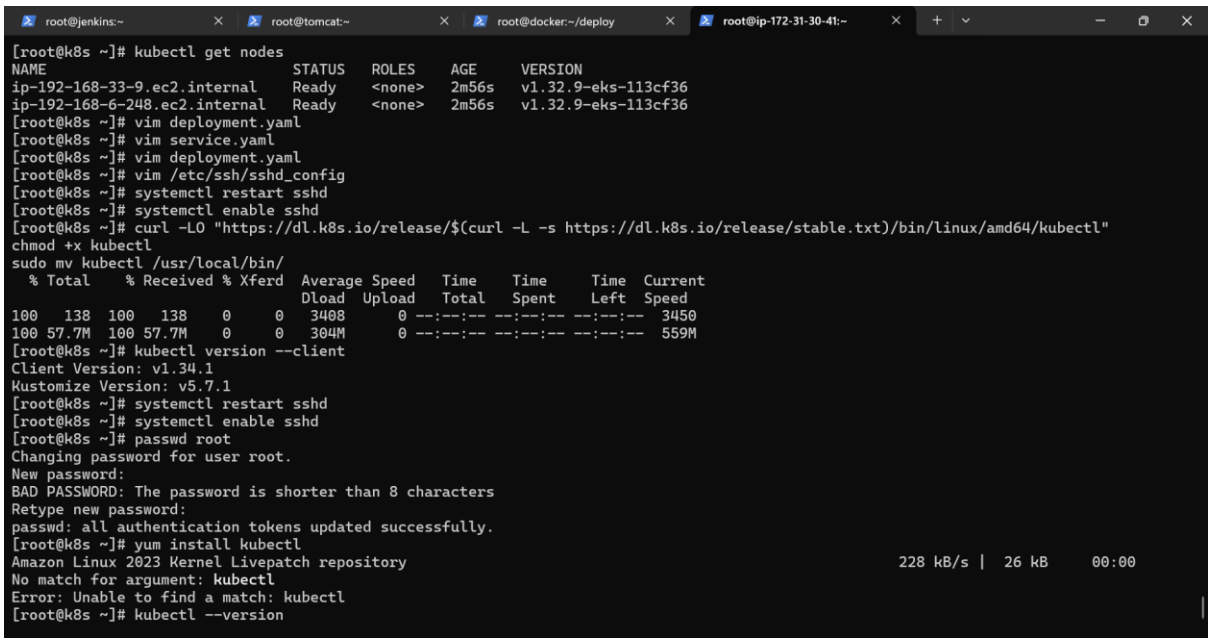


Install k8s in terminal and also install cluster.

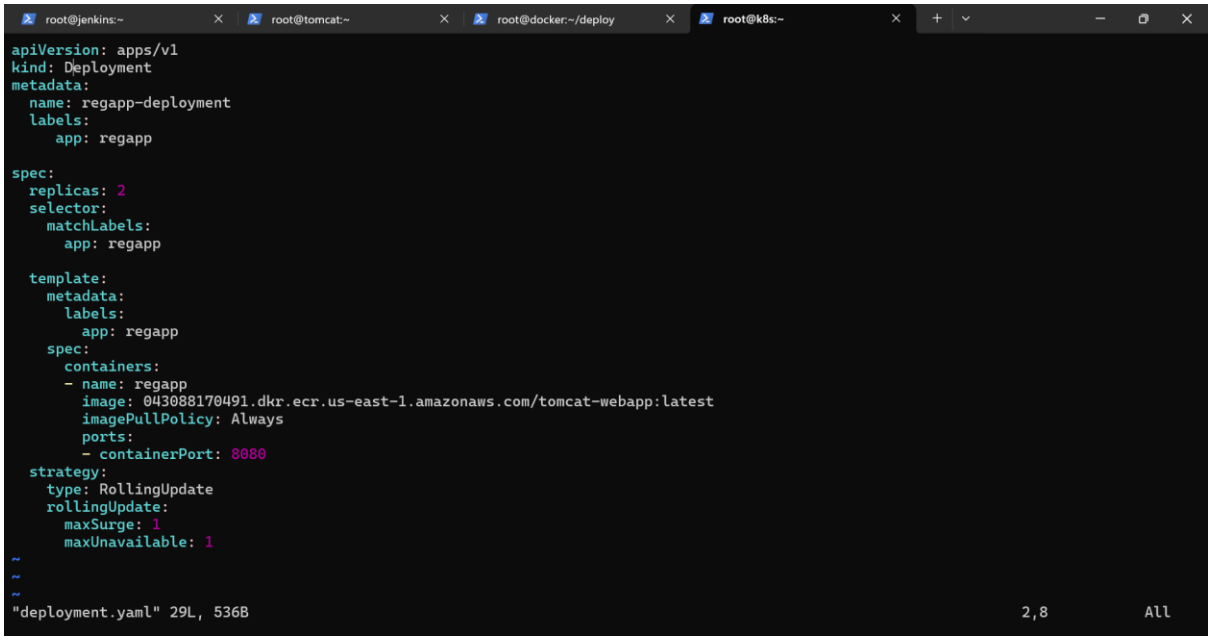


Create deployment file and specify image URL properly.

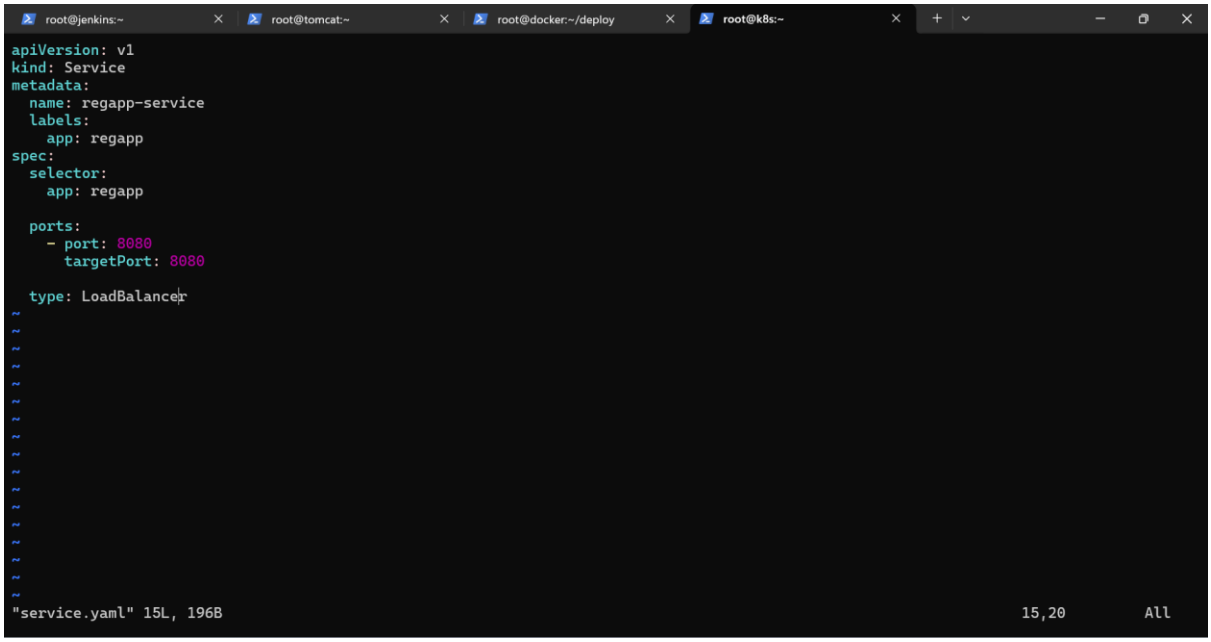
Here we need to create a role in IAM to setup control in EKS.



The deployment file should be in this format.



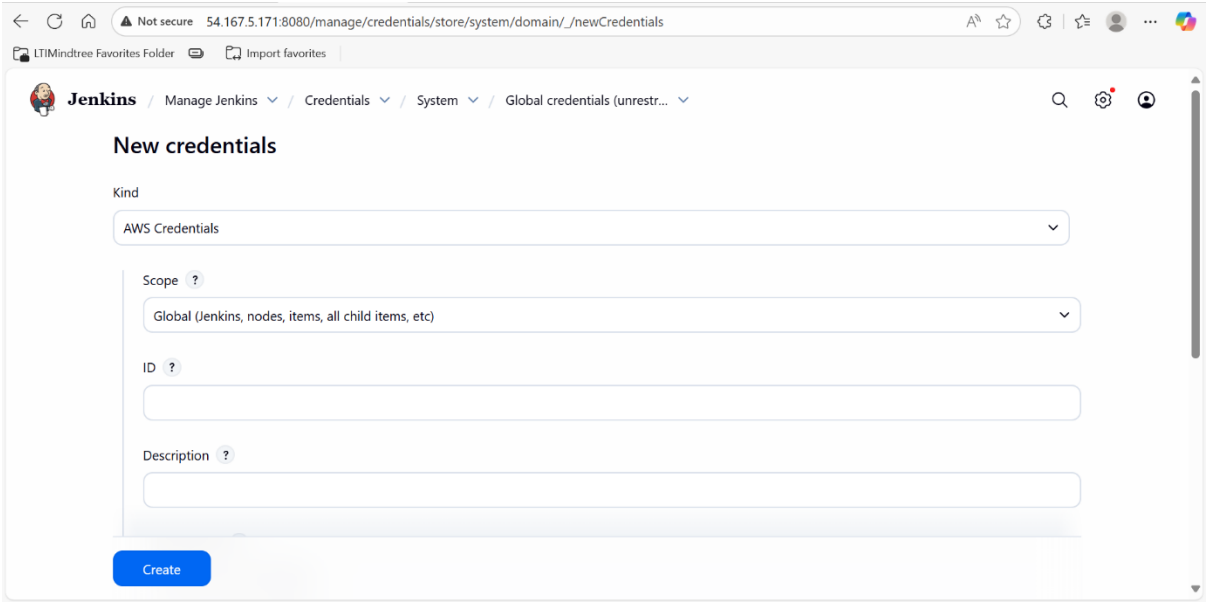
The Service file should be in this format.



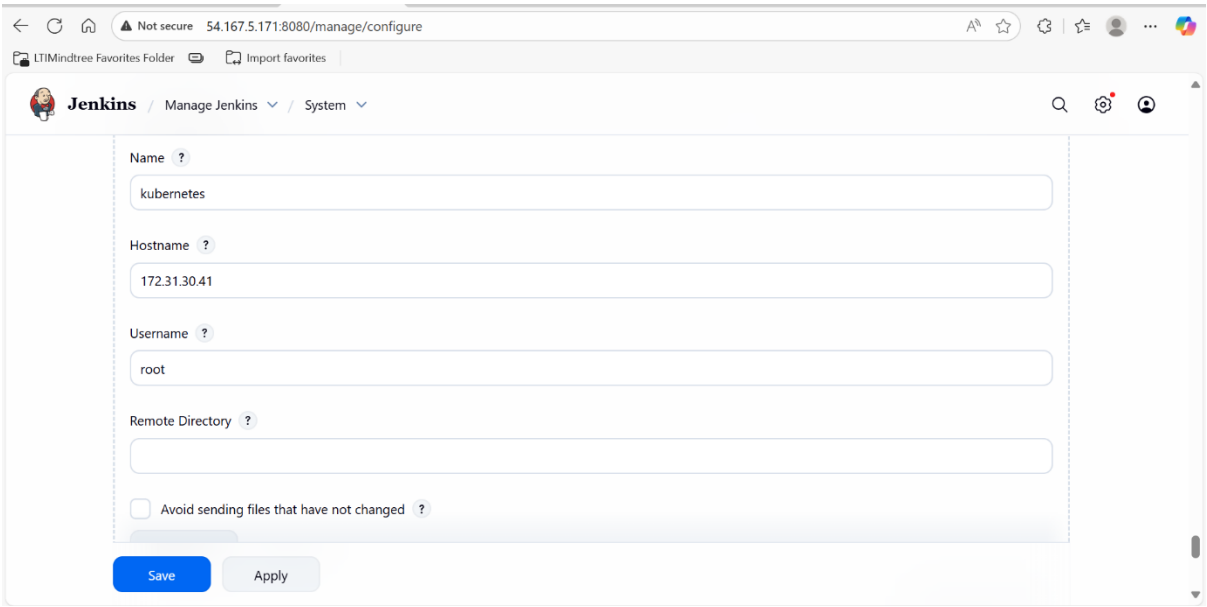
Here we specify for creating load balancers.

Also make sure that sshd should be configured and made necessary changes in it.

Add global credentials in Jenkins

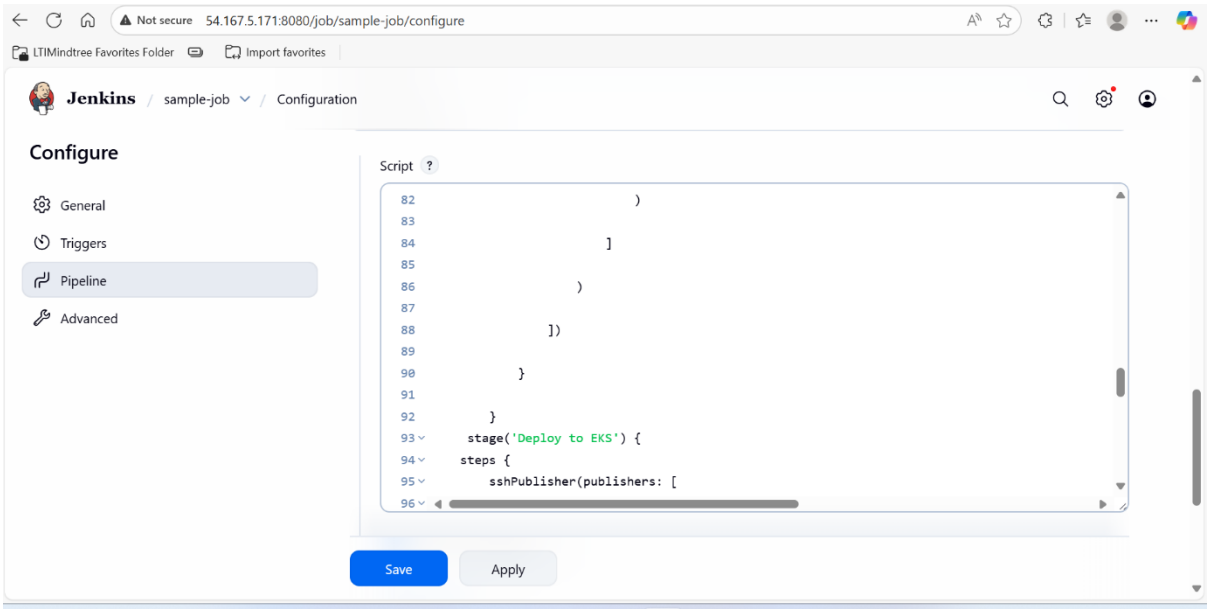


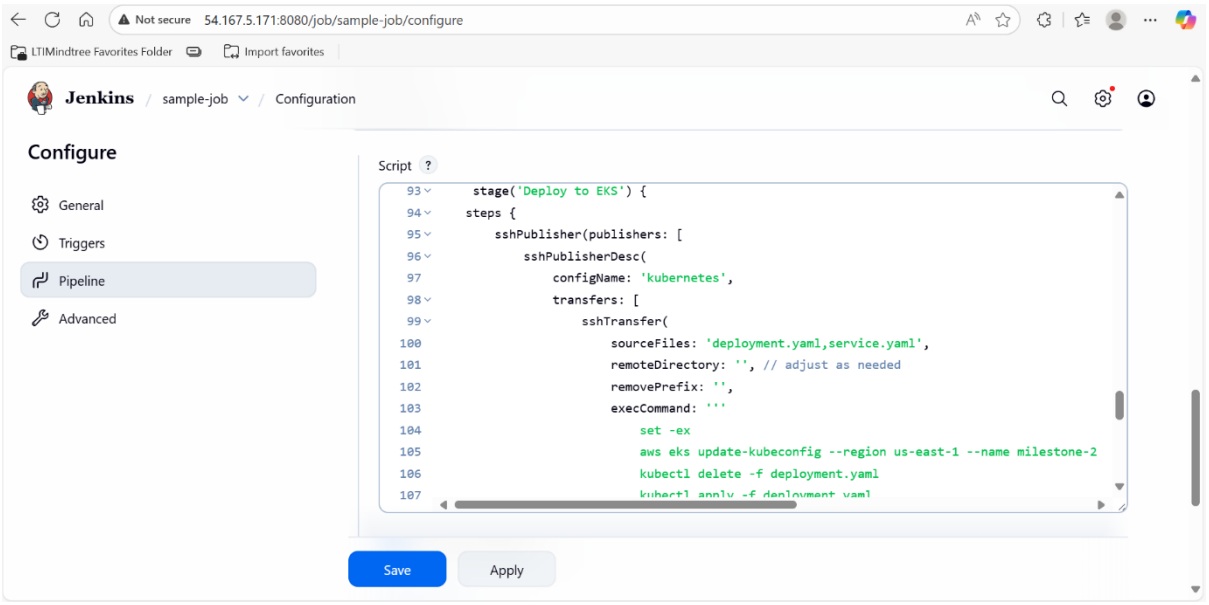
Create k8s SSH server in Jenkins

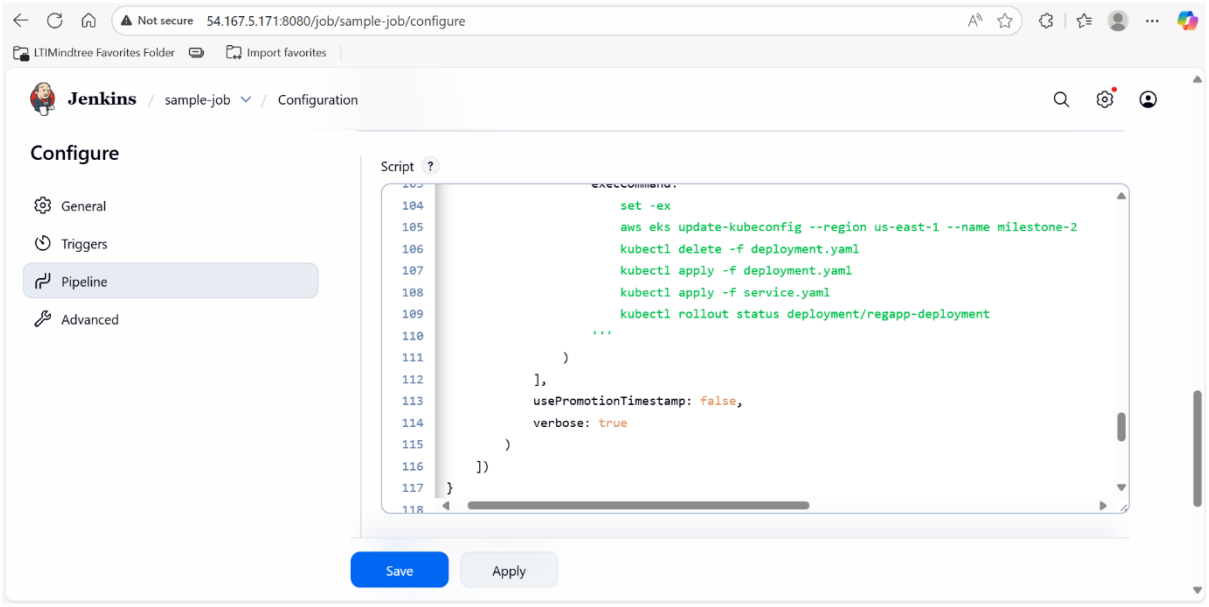


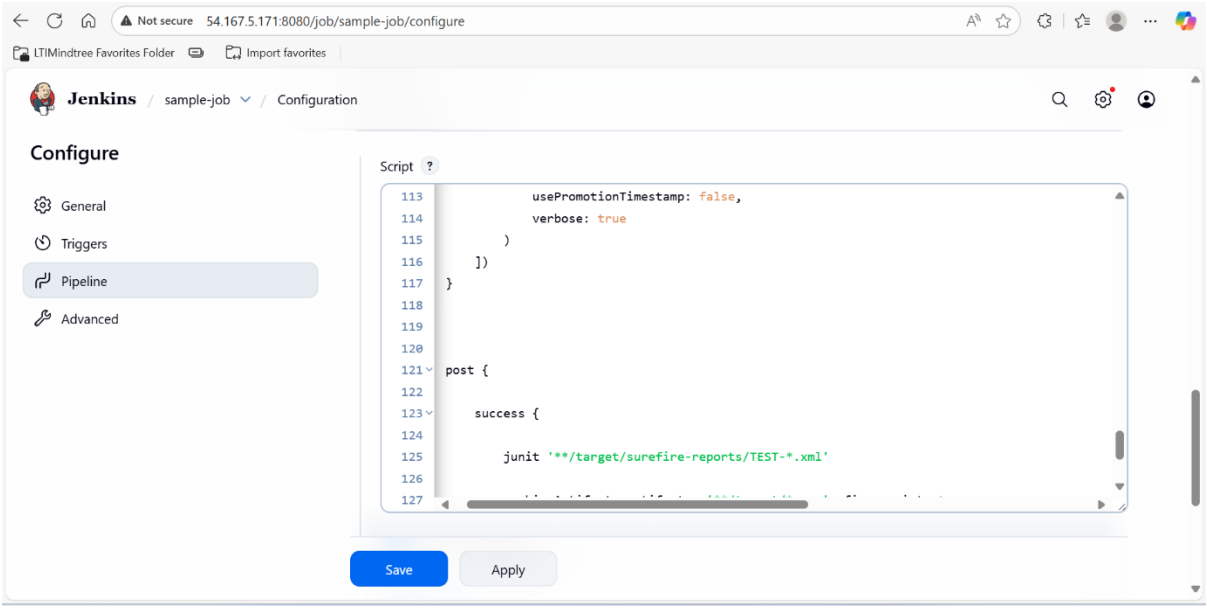
Now create a pipeline project in Jenkins by adding new item.

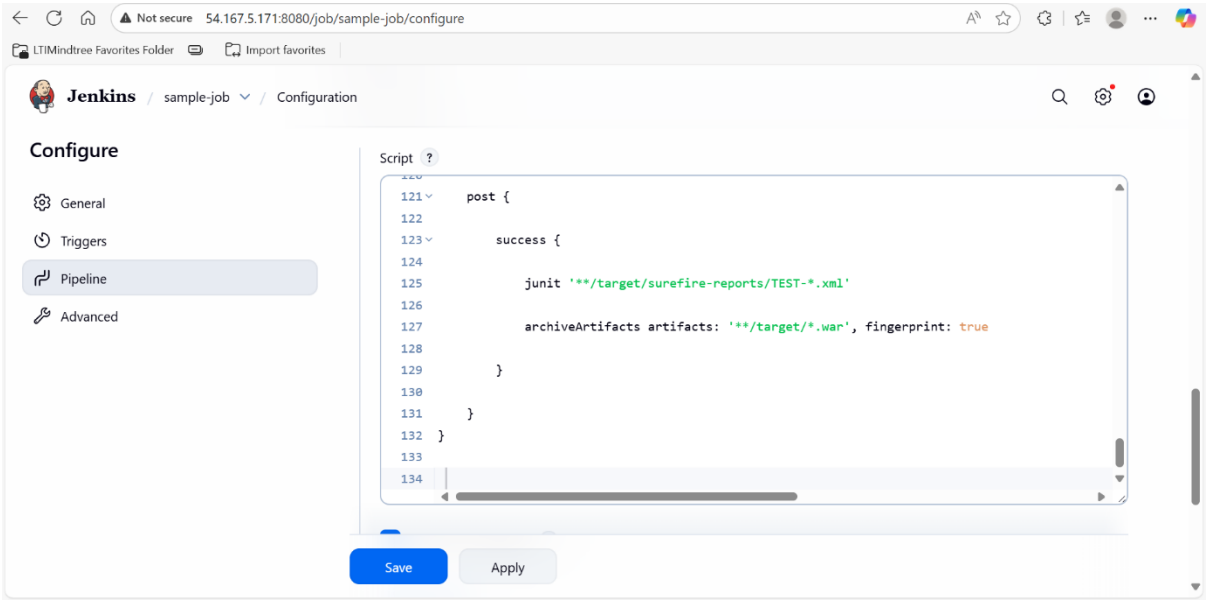
Apply groovy script as this is the declarative approach.


Not secure 54.167.5.171:8080/job/sample-job/configure
LTIMindtree Favorites Folder
Import favorites
Jenkins / sample-job / Configuration
Configure
General
Triggers
Pipeline
Advanced
Script
82)
83
84]
85
86)
87
88])
89
90 }
91
92 }
93 stage('Deploy to EKS') {
94 steps {
95 sshPublisher(publishers: [
96
Save
Apply

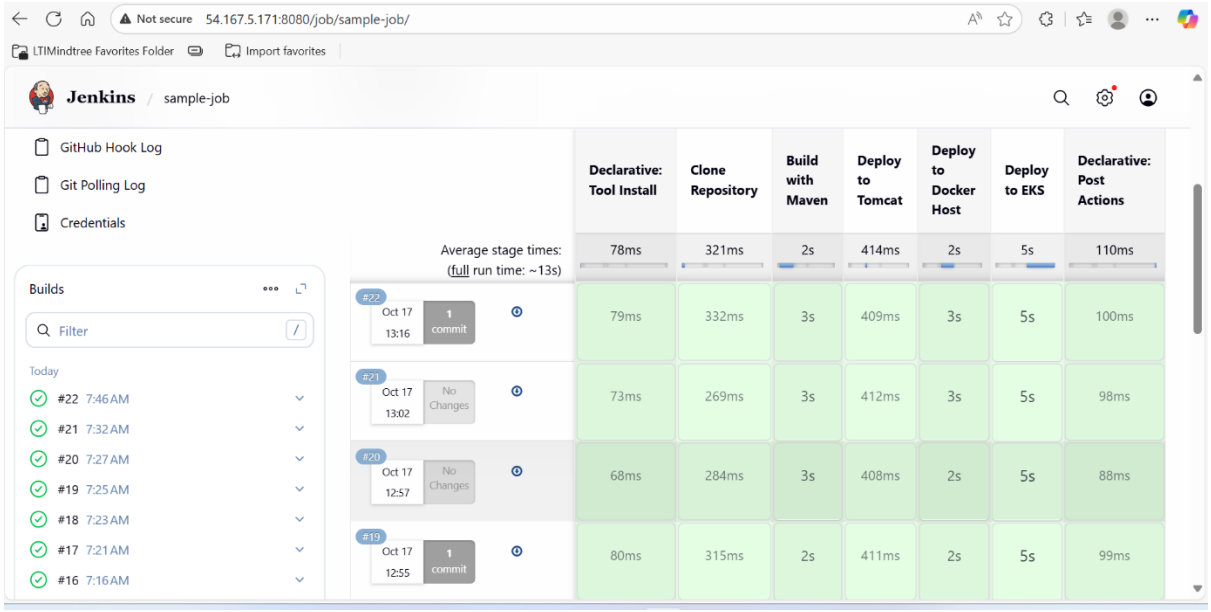

Not secure 54.167.5.171:8080/job/sample-job/configure
LTIMindtree Favorites Folder
Import favorites
Jenkins / sample-job / Configuration
Configure
General
Triggers
Pipeline
Advanced
Script
93 stage('Deploy to EKS') {
94 steps {
95 sshPublisher(publishers: [
96 sshPublisherDesc(
97 configName: 'kubernetes',
98 transfers: [
99 sshTransfer(
100 sourceFiles: 'deployment.yaml,service.yaml',
101 remoteDirectory: '', // adjust as needed
102 removePrefix: '',
103 execCommand: '''
104 set -ex
105 aws eks update-kubeconfig --region us-east-1 --name milestone-2
106 kubectl delete -f deployment.yaml
107 kubectl apply -f deployment.yaml
Save
Apply


Not secure 54.167.5.171:8080/job/sample-job/configure
LTIMindtree Favorites Folder
Import favorites
Jenkins / sample-job / Configuration
Configure
General
Triggers
Pipeline
Advanced
Script
104 set -ex
105 aws eks update-kubeconfig --region us-east-1 --name milestone-2
106 kubectl delete -f deployment.yaml
107 kubectl apply -f deployment.yaml
108 kubectl apply -f service.yaml
109 kubectl rollout status deployment/regapp-deployment
110 '''
111)
112],
113 usePromotionTimestamp: false,
114 verbose: true
115)
116])
117 }
Save
Apply

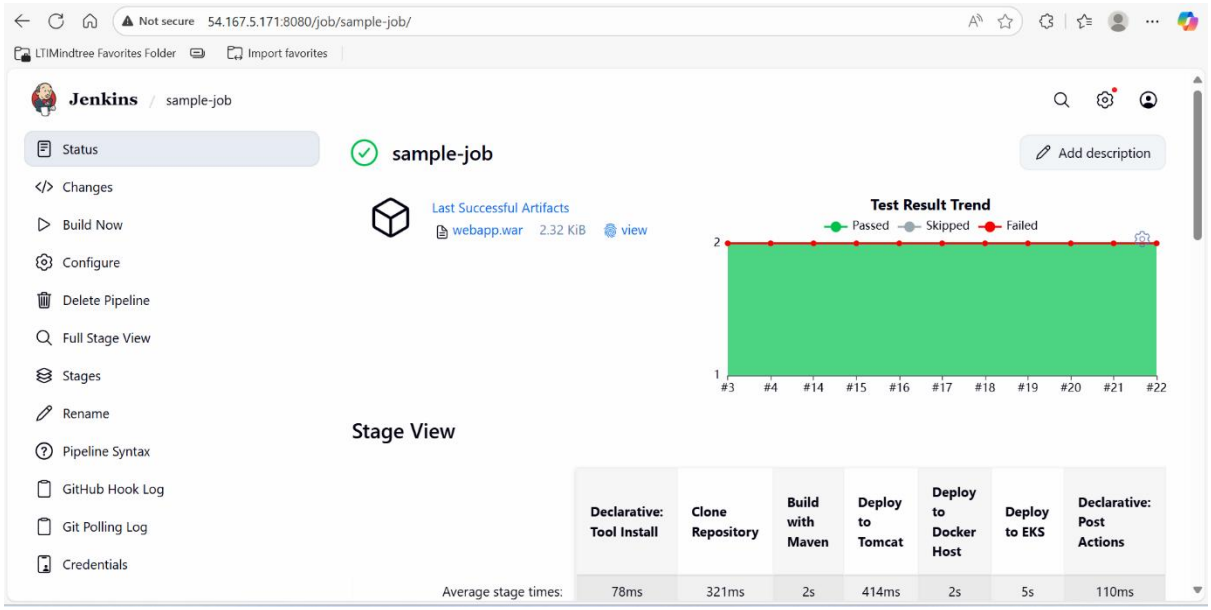




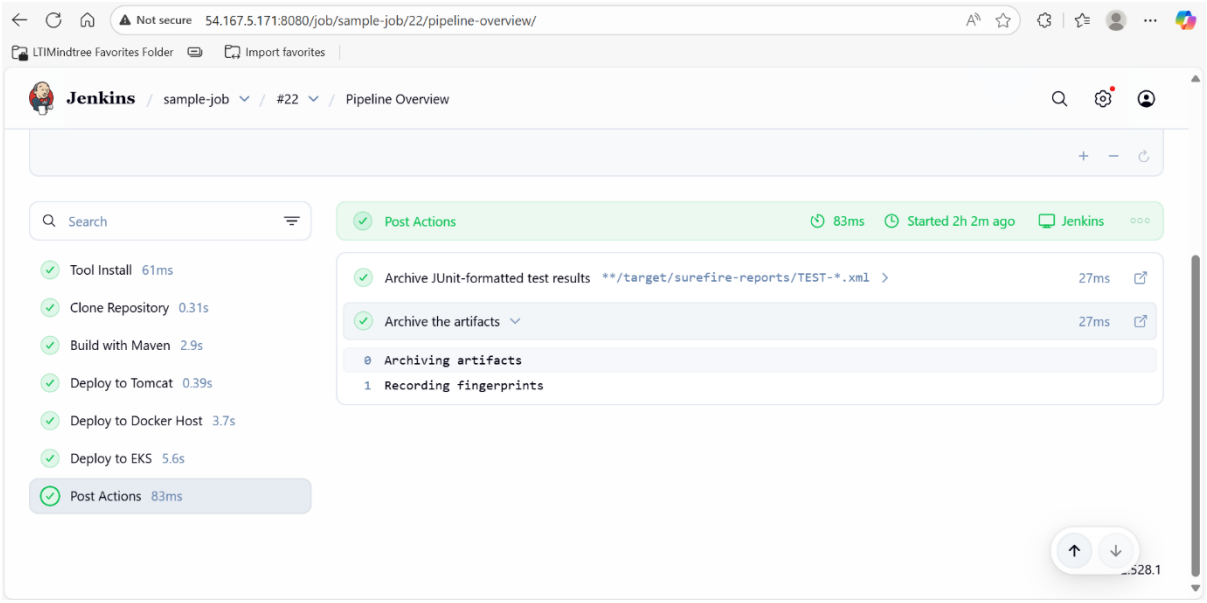
This is the groovy script of our final declarative part. Now click on build now to see how it works.



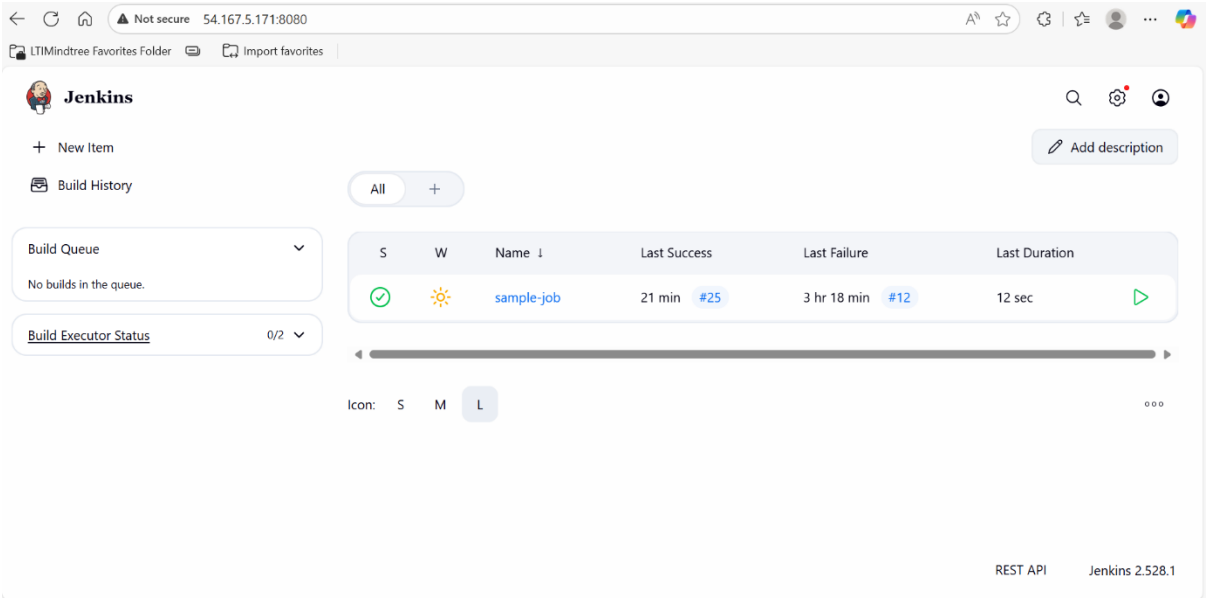
The build was successful so we have build an project. Let's see some of the outputs now.



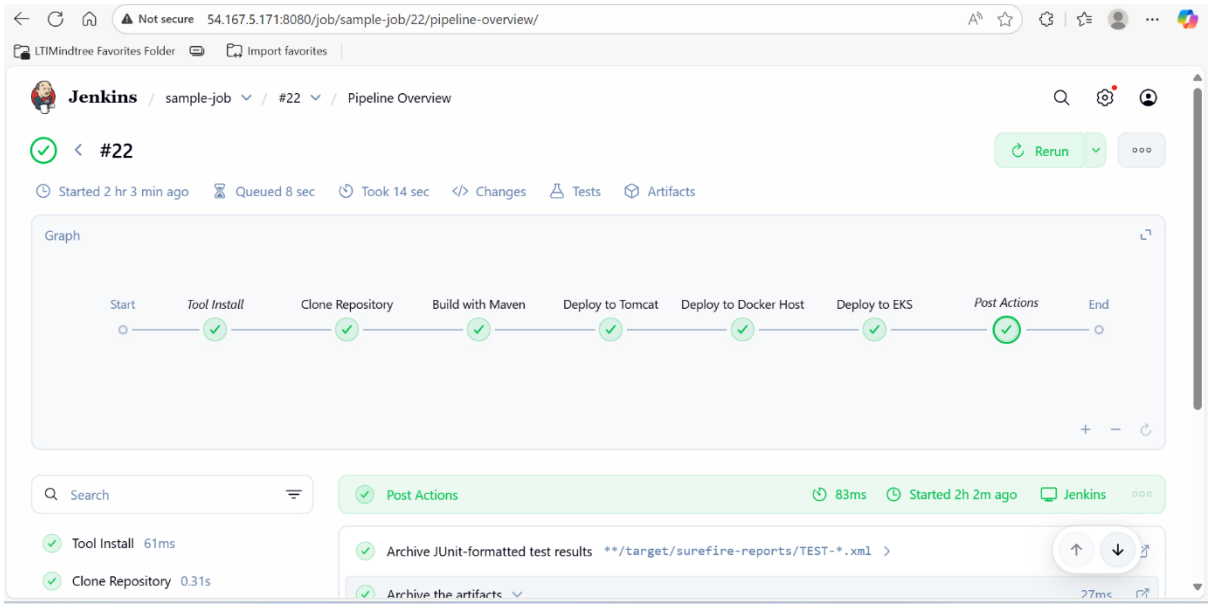
This indicates the status of build.



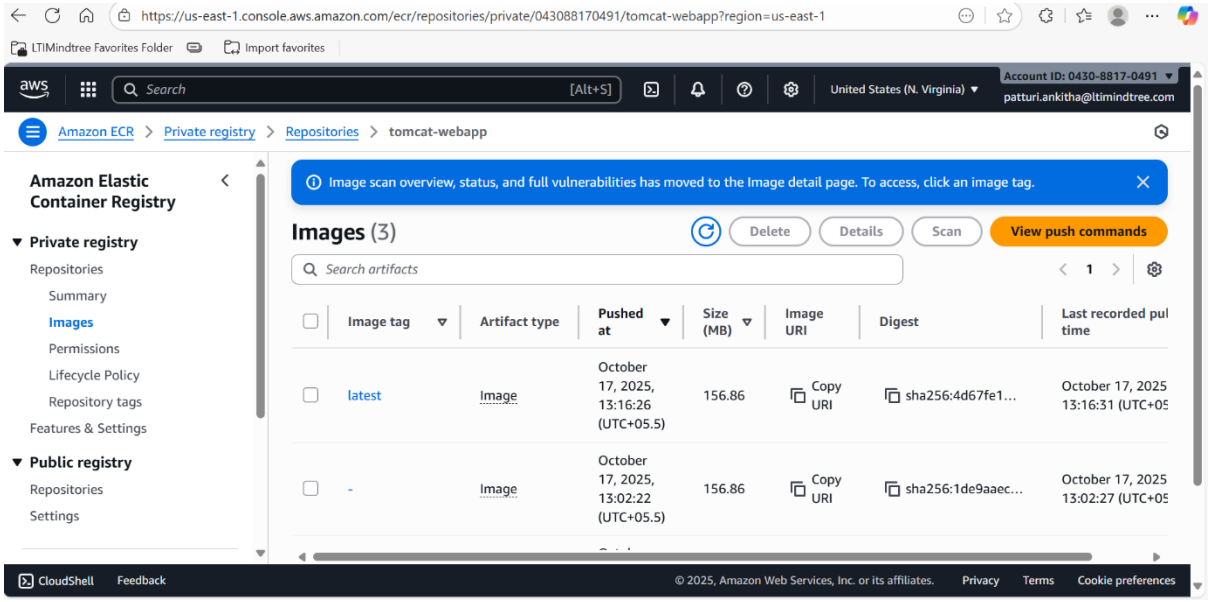
The post actions of the build are mentioned above.



This indicates the sample-job project is successfully done.



This is the final overview of our project which we have done by using different tools in Devops.



The latest images are pushed into the ECR by using Eks cluster.

When we paste the svc url in the browser we will get this output.

Source code:

```
pipeline {
 agent any
tools {
maven 'maven' // Ensure this is configured in Jenkins Global Tool Configuration
}
environment {

        TOMCAT_USER = 'admin'
        TOMCAT_PASS = 'admin'
        TOMCAT_HOST = '107.20.39.244'
        TOMCAT_PORT = '8080'
    }
    stages {
        stage('Clone Repository') {
            steps {
                git branch: 'main', url: 'https://github.com/ankithapatturi/devops-project.git'
            }
        }
```

```
        stage('Build with Maven') {
            steps {
                sh 'mvn clean package -Dmaven.test.failure.ignore=true'


            }


        }
        stage('Deploy to Tomcat') {
            steps {
                sh '''
                    curl -u $TOMCAT_USER:$TOMCAT_PASS \
                    --upload-file webapp/target/webapp.war \

"http://$TOMCAT_HOST:$TOMCAT_PORT/manager/text/deploy?path=/webapp&update=t
rue"
                '''
            }
        }
        stage('Deploy to Docker Host') {
            steps {
                sshPublisher(publishers: [
                    sshPublisherDesc(
                        configName: 'docker',
                        transfers: [
                            sshTransfer(


                                sourceFiles: '**/*',
                                removePrefix: '',
                                remoteDirectory: 'deploy',
                                execCommand: '''
                                    cd deploy
```

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 043088170491.dkr.ecr.us-east-1.amazonaws.com

docker build -t tomcat-webapp .

docker tag tomcat-webapp:latest 043088170491.dkr.ecr.us-east-1.amazonaws.com/tomcat-webapp:latest

docker push 043088170491.dkr.ecr.us-east-1.amazonaws.com/tomcat-webapp:latest

```
                    '''
                )
            ]
        )
    ])
  }
}
 stage('Deploy to EKS') {
steps {
  sshPublisher(publishers: [
    sshPublisherDesc(
      configName: 'kubernetes',
      transfers: [
        sshTransfer(
          sourceFiles: 'deployment.yaml,service.yaml',
          remoteDirectory: '', // adjust as needed
          removePrefix: '',
          execCommand: '''
            set -ex
            aws eks update-kubeconfig --region us-east-1 --name milestone-2
            kubectl apply -f deployment.yaml
            kubectl apply -f service.yaml
            kubectl rollout status deployment/regapp-deployment
          '''
```

```
                )
            ],
            usePromotionTimestamp: false,
            verbose: true
        )
    ])
  }
}
}
  post {
    success {
      junit '**/target/surefire-reports/TEST-*.xml'
      archiveArtifacts artifacts: '**/target/*.war', fingerprint: true
    }
  }
}
```

Advantages of using Declarative Pipeline method:

1. Code-Based Configuration

- Declarative pipelines are written as code (Jenkinsfile), making them version-controlled and easy to track.
- UI method relies on manual configuration, which is harder to audit or replicate.

2. Reusability & Portability

- You can reuse the same Jenkinsfile across multiple projects or environments.
- UI configurations are tied to a specific Jenkins instance and are not easily portable.

3. Better Collaboration

- Teams can collaborate on pipeline code just like application code.
- Changes can be reviewed via pull requests in GitHub.

4. Easier Maintenance

- Declarative syntax is structured and easier to read and maintain.

- UI jobs can become complex and hard to manage as they grow.

5. Supports Advanced Features

- Declarative pipelines support features like:
    - Parallel execution
    - Post-build actions
    - Environment variables
    - Error handling (post, always, success, failure blocks)

6. Secure & Auditable

- All pipeline logic is stored in source control, making it auditable.
- UI-based jobs can be changed without trace unless manually documented.

7. Automation-Friendly

- Declarative pipelines are ideal for Infrastructure as Code (IaC) and DevOps automation.
- UI method is more manual and less suited for dynamic environments.