

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
 ~ \~'`->
 ~ \_/
 ~ \_/_/
 _/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:N50lGKdQ/LLBkt6nPbY2U5GFNDhEVMN9cvqLjTavjA root@jenkins.example.com
The key's randomart image is:
+---[RSA 3072]---+
| .. oBB+
| .. .oo|
| .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 AAAAB3NzaC1yc2EAAAQABAAAbgQCwErGxQJrBA2mHBCMum/086sNq20a6B9CMiK4QDkpl+tfB0QhVYqY6FQ1cCeETYVTNVrRgjHk7VsHvOjGZX8ame6xVwkbhxbFV8nclBkxM9epdfLP1Pf86Lg7tm90G2cnHm0hASdgRCvw07UbxE07I9ZNRRdG7d3xMu0j9XN8d0W7md0Fnbyt3gM8sSgvon9MMxbAbfFDIS9404M6Z30bnsFqMU3yl/sdnYepuPuMAF638qvzR1J24Fbj2XF23r600Mxkape05PRRFjZqlrmruHVxonTo2uUX7+IfDBDcDB/8xPLakMsoFYTLF/GKKVV7KE3Cg1wvVQRs1DdPkqtVbZAzhXJKGq29Fcvy2AxAlmYGHG29eLOxrLgRScl/v45jlx+AnUw0DJB1UlwLqoS6qY54wXC+cyVCH513bx79YsmhljA8cfgLN8J+UHCzo05KN7qypwI83flyDnp4w61WGNZBxs1BJf2mBLXGOVNhQPTzde50xX6n3+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.

The screenshot shows the GitHub settings interface for adding a new SSH key. On the left, there's a sidebar with various account and access options. The 'SSH and GPG keys' section is currently selected. The main area is titled 'Add new SSH Key' and contains fields for 'Title' (set to 'ankitha') and 'Key type' (set to 'Authentication Key'). Below these is a large text input field containing the SSH key itself, which starts with 'ssh-rsa AAAAB3NzaC1yc2EAAAQABAA...'. At the bottom of this section is a green 'Add SSH key' button.

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:+DiY3vvvV6TuJhbzisF/zLDA0zPMsvHdkr4UvC0qU.
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.

The screenshot shows a GitHub repository page for 'devops-project'. The repository is public and has 18 commits. The commits are listed as follows:

Author	File	Message	Date
ankithapatturi	Update index.jsp	a89e53f - 40 minutes ago	18 Commits
	server	fin-com	last year
	webapp	Update index.jsp	40 minutes ago
	Dockerfile	fin-com	last year
	Jenkinsfile	fin-com	last year
	README.md	Update README.md	last year
	pom.xml	fin-com	last year

On the right side, there is an 'About' section with the message 'No description, website, or topics provided.' and a 'Releases' section indicating 'No releases published'.

All the files are present in my Github Repository.

Create the instances for our project requirements.

Jenkins&Maven:

The screenshot shows the AWS EC2 Instances page. There are six instances listed, all of which are running. The instance details are as follows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
docker-server	i-0d2acaee475a3f3a3	Running	t3.micro	3/3 checks passed	View alarms +
tomcat-server	i-04240ebe8d9920e1e	Running	t3.micro	3/3 checks passed	View alarms +
jenkins	i-02dd8938b7d6f2dd5	Running	c7i-flex.large	3/3 checks passed	View alarms +
eks-node	i-0677139bcd8d1364e	Running	t3.micro	3/3 checks passed	View alarms +

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/esLz4Nwtx8yJRhwE30bCH4.
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 -
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.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

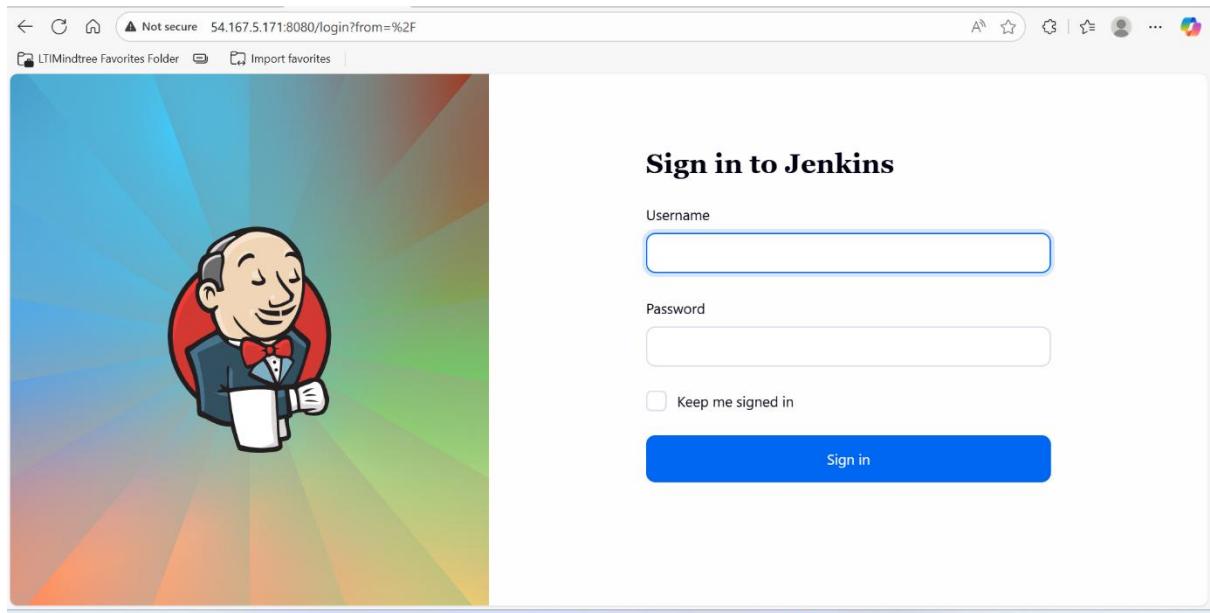
Please copy the password from either location and paste it below.

Administrator password

Continue

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.

```

root@jenkins:~          X  Windows PowerShell      X  root@docker:~/deploy   X  root@ip-172-31-30-41:~  X  -  D  X
Oct 17 03:44:45 jenkins.example.com systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 17 03:44:45 jenkins.example.com jenkins[26257]: 2025-10-17 03:44:45.932+0000 [id=56]      INFO      h.m.DownloadService$Downl...
Oct 17 03:44:45 jenkins.example.com jenkins[26257]: 2025-10-17 03:44:45.934+0000 [id=56]      INFO      hudson.util.Retrier$star...
[root@jenkins ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
c89120bfbe0f4f8f89d60e52cd200d1f
[root@jenkins ~]# yum install maven -y
Last metadata expiration check: 0:03:38 ago on Fri Oct 17 03:43:55 2025.
Dependencies resolved.
=====
Package           Architecture Version       Repository      Size
=====
Installing:
maven             noarch     1:3.8.4-3.amzn2023.0.5    amazonlinux   18 k
Installing dependencies:
apache-commons-cli noarch     1.5.0-3.amzn2023.0.3    amazonlinux   76 k
apache-commons-codec noarch     1.15-6.amzn2023.0.3    amazonlinux   303 k
apache-commons-io  noarch     1:2.8.0-7.amzn2023.0.5   amazonlinux   283 k
apache-commons-lang3 noarch     3.18.0-1.amzn2023.0.1   amazonlinux   655 k
atinject          noarch     1.0.5-3.amzn2023.0.3   amazonlinux   23 k
cdi-api           noarch     2.0.2-6.amzn2023.0.3   amazonlinux   54 k
google-guice      noarch     4.2.3-8.amzn2023.0.6   amazonlinux   473 k
guava              noarch     31.0.1-3.amzn2023.0.6   amazonlinux   2.4 M
httpcomponents-client noarch     4.5.13-4.amzn2023.0.4   amazonlinux   657 k
httpcomponents-core noarch     4.4.13-6.amzn2023.0.3   amazonlinux   632 k
jakarta-annotations noarch     1.3.5-13.amzn2023.0.3  amazonlinux   46 k
jansi              x86_64    2.4.0-3.amzn2023.0.3   amazonlinux   113 k
java-17-amazon-corretto-devel x86_64    1:17.0.16+8-1.amzn2023.1  amazonlinux   142 k
java-17-amazon-corretto-headless x86_64    1:17.0.16+8-1.amzn2023.1  amazonlinux   91 M
jcl-over-slf4j      noarch     1.7.32-3.amzn2023.0.4   amazonlinux   25 k
jsoup              noarch     1.16.1-4.amzn2023.0.2   amazonlinux   433 k
jsr-305            noarch     3.0.2-5.amzn2023.0.4   amazonlinux   32 k
maven-amazon-corretto17 noarch     1:3.8.4-3.amzn2023.0.5   amazonlinux   9.4 k
maven-lib          noarch     1:3.8.4-3.amzn2023.0.5   amazonlinux   1.5 M

```

```

Complete!
[root@jenkins ~]# yum install git -y
Last metadata expiration check: 0:03:51 ago on Fri Oct 17 03:43:55 2025.
Dependencies resolved.
=====
Package           Architecture      Version       Repository   Size
=====
Installing:
git              x86_64          2.50.1-1.amzn2023.0.1    amazonlinux  53 k
Installing dependencies:
git-core          x86_64          2.50.1-1.amzn2023.0.1    amazonlinux  4.9 M
git-core-doc     noarch          2.50.1-1.amzn2023.0.1    amazonlinux  2.8 M
perl-Error        noarch          1:0.17029-5.amzn2023.0.2  amazonlinux  41 k
perl-File-Find   noarch          1.37-477.amzn2023.0.7   amazonlinux  25 k
perl-Git          noarch          2.50.1-1.amzn2023.0.1    amazonlinux  41 k
perl-TermReadKey x86_64          2.38-9.amzn2023.0.2     amazonlinux  36 k
perl-lib          x86_64          0.65-477.amzn2023.0.7   amazonlinux  15 k
=====
Transaction Summary
=====
Install 8 Packages
Total download size: 7.9 M
Installed size: 41 M
Downloading Packages:
(1/8): git-core-2.50.1-1.amzn2023.0.1.x86_64.rpm 65 MB/s | 4.9 MB  00:00
(2/8): git-core-doc-2.50.1-1.amzn2023.0.1.noarch.rpm 34 MB/s | 2.8 MB  00:00
(3/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm 2.1 MB/s | 41 kB   00:00
(4/8): perl-File-Find-1.37-477.amzn2023.0.7.noarch.rpm 1.2 MB/s | 25 kB   00:00
(5/8): git-2.50.1-1.amzn2023.0.1.x86_64.rpm 495 kB/s | 53 kB   00:00
(6/8): perl-Git-2.50.1-1.amzn2023.0.1.noarch.rpm 1.8 MB/s | 41 kB   00:00
(7/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm 1.6 MB/s | 36 kB   00:00
(8/8): perl-lib-0.65-477.amzn2023.0.7.noarch.rpm

```

Add required plugins in tools of the Jenkins.

The screenshot shows the Jenkins plugin manager interface. The URL is `http://54.167.5.171:8080/manage/pluginManager/available`. The left sidebar has tabs for Updates, Available plugins (which is selected), Installed plugins, and Advanced settings. The main area has a search bar for "Aws credential". Below it, a table lists available plugins:

Install	Name	Released	Health
<input checked="" type="checkbox"/>	GitHub Integration 0.7.2 emailext Build Triggers	9 mo 5 days ago	87
<input checked="" type="checkbox"/>	Maven Integration 3.27 Build Tools	2 mo 6 days ago	100
<input checked="" type="checkbox"/>	Publish Over SSH 390.vb_f56e7405751 Artifact Uploaders Build Tools	3 mo 14 days ago	96
<input checked="" type="checkbox"/>	Deploy to container 1.17		

Not secure 54.167.5.171:8080/manage/pluginManager/available

LTMindtree Favorites Folder Import favorites

Jenkins / Manage Jenkins / Plugins

Plugins

- Updates
- Available plugins**
- Installed plugins
- Advanced settings

Aws credential

Install

Plugin	Description	Last Updated	Version	Downloads
Deploy to container 1.17	Artifact Uploaders	5 mo 7 days ago	1.17	91
Pipeline Graph Analysis 245.v88f03631a_b_21	Library plugins (for use by other plugins)	1 mo 7 days ago	245.v88f03631a_b_21	100
Pipeline: REST API 2.38	User Interface	5 mo 20 days ago	2.38	100
Pipeline: Stage View 2.38	User Interface	5 mo 20 days ago	2.38	100
AWS Credentials 254.v978a_5e206a_d7	aws	1 mo 25 days ago	254.v978a_5e206a_d7	96

Not secure 54.167.5.171:8080/manage/pluginManager/available

LTMindtree Favorites Folder Import favorites

Jenkins / Manage Jenkins / Plugins

Plugins

- Updates
- Available plugins**
- Installed plugins
- Advanced settings

Aws credential

Install

Plugin	Description	Last Updated	Version	Downloads
Deploy to container 1.17	Artifact Uploaders	5 mo 7 days ago	1.17	91
Pipeline Graph Analysis 245.v88f03631a_b_21	Library plugins (for use by other plugins)	1 mo 7 days ago	245.v88f03631a_b_21	100
Pipeline: REST API 2.38	User Interface	5 mo 20 days ago	2.38	100
Pipeline: Stage View 2.38	User Interface	5 mo 20 days ago	2.38	100
AWS Credentials 254.v978a_5e206a_d7	aws	1 mo 25 days ago	254.v978a_5e206a_d7	96

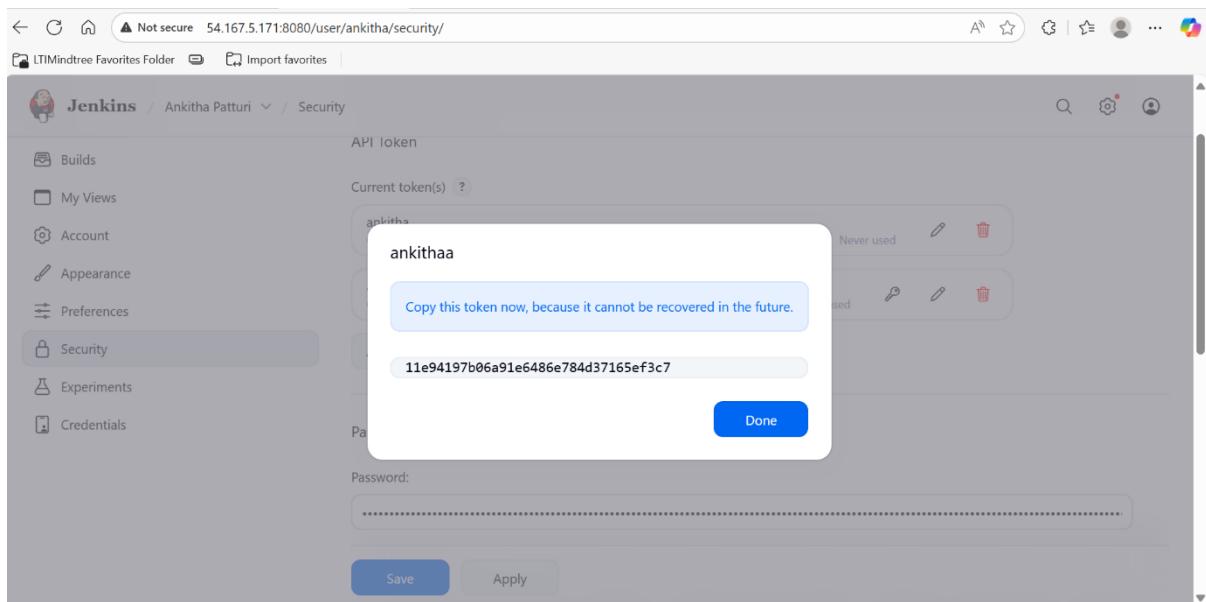
The screenshot shows the Jenkins plugin manager interface. On the left, there's a sidebar with options like 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress' (which is selected). The main area is titled 'Download progress' and shows a list of installed plugins with their status: GitHub Integration, Maven Integration, Publish Over SSH, Deploy to container, Pipeline Graph Analysis, Pipeline: REST API, Pipeline: Stage View, AWS Credentials, and Loading plugin extensions, all marked as 'Success'. At the bottom, there's a note: '→ Go back to the top page (you can start using the installed plugins right away)'

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

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

The screenshot shows the Jenkins system configuration page for 'SSH Server'. It has fields for 'Name' (set to 'jenkins'), 'Hostname' (set to '172.31.16.116'), 'Username' (set to 'root'), and 'Remote Directory' (empty). There's also a checkbox for 'Avoid sending files that have not changed'. At the bottom are 'Save' and 'Apply' buttons.

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.

A screenshot of a web browser displaying the GitHub Settings page for the repository 'devops-project'. The URL is https://github.com/ankithapatturi/devops-project/settings/hooks. The left sidebar shows repository settings sections: General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Models), Webhooks (selected), Copilot, and Environments. The main content area is titled 'Webhooks' and contains a list of configured hooks. One hook is listed with the URL 'http://54.167.5.171:8080/github-webhook/push' and a status message 'Last delivery was successful.' Buttons for 'Edit' and 'Delete' are shown next to the hook entry.

Add Java and Maven paths in tools of Jenkins.

```

root@jenkins:~          x  Windows PowerShell          x  root@docker:~/deploy          x  root@ip-172-31-30-41:~          x  +  v  -  o  x
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:xDI6cZxernfizz7dMuqNn4QgaXKC5e4yyqZhykrm08k root@jenkins.example.com
The key's randomart image is:
+--[RSA 3072]----+

```

The screenshot shows the Jenkins management interface for configuring tools. Under the 'Tools' section, a new Java Development Kit (JDK) is being configured. The configuration form includes:

- Name:** java
- JAVA_HOME:** /usr/lib/jvm/java-17-amazon-corretto.x86_64
- Install automatically:** (unchecked)

At the bottom of the form are 'Save' and 'Apply' buttons.

Not secure 54.167.5.171:8080/manage/configureTools/

LTMindtree Favorites Folder Import favorites

Jenkins / Manage Jenkins / Tools

+ Add Maven

Maven

Name: maven

MAVEN_HOME: /usr/share/maven

Install automatically ?

+ Add Maven

Save Apply

Now connect Tomcat instance to terminal.

```
root@jenkins:~          X  Windows PowerShell          X  root@docker:~/deploy      X  root@ip-172-31-30-41:~      X + 
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\l0844392> cd Downloads
PS C:\Users\l0844392\Downloads> ssh -i "project-key.pem" ec2-user@ec2-107-20-39-244.compute-1.amazonaws.com
The authenticity of host 'ec2-107-20-39-244.compute-1.amazonaws.com (107.20.39.244)' can't be established.
ED25519 key fingerprint is SHA256:1Q+CLYT8c5V1H0shcRTT+WojeB2sPHoptHEMxdpAA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-107-20-39-244.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

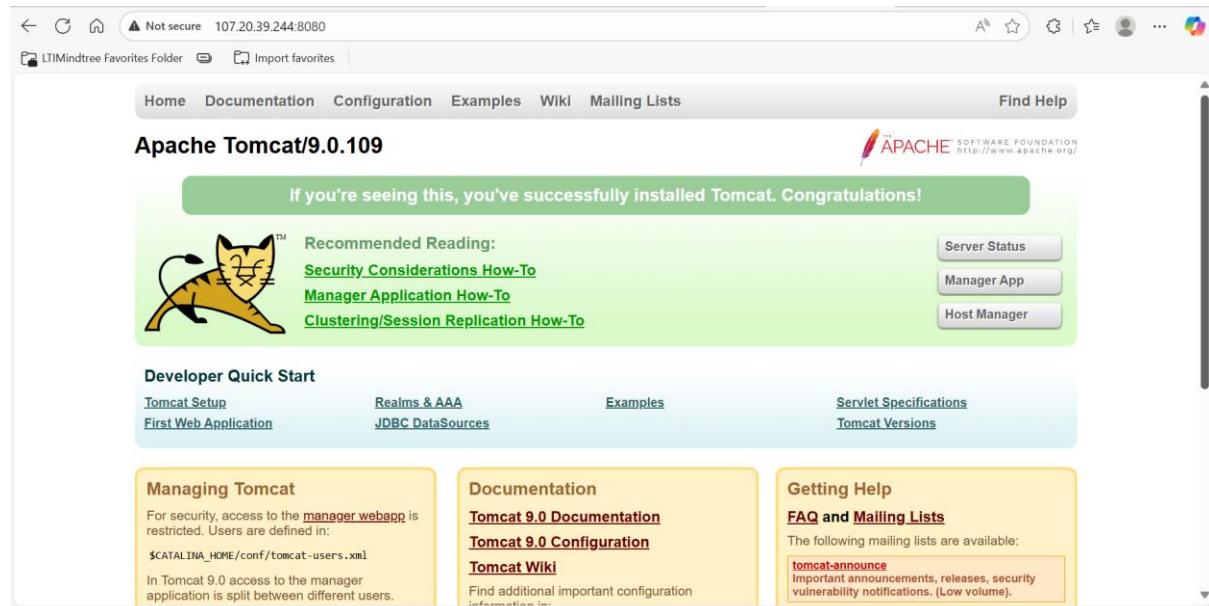
#_
  _\ #####_      Amazon Linux 2023
  _\ #####\_
  _\ \#\#\|_
  _\ \#\#\|_
  _\ / \_\_>  https://aws.amazon.com/linux/amazon-linux-2023
  _\ \_\_>
  _\ \_\_>
  _\ \_\_>
  _\ \_\_>
  _\ \_\_>
[ec2-user@ip-172-31-18-59 ~]$ sudo su -
[root@ip-172-31-18-59 ~]# hostnamectl set-hostname tomcat.example.com
[root@ip-172-31-18-59 ~]# bash
[root@tomcat ~]# vim tomcat.sh
[root@tomcat ~]# vim tomcat.sh
[root@tomcat ~]# chmod +x tomcat.sh
[root@tomcat ~]# ./tomcat.sh
./tomcat.sh: line 1: !#/bin/bash: No such file or directory
--2025-10-17 03:57:47--  https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.109/bin/apache-tomcat-9.0.109.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
```

Install Tomcat by giving required commands.

Add manager roles in tomcat terminal.

```
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" 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 --
```

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



Tomcat is installed successfully and it's working.

Not secure 54.167.5.171:8080/manage/credentials/store/system/domain/_/newCredentials

LTMindtree Favorites Folder Import favorites

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestr...)

New credentials

Kind: Username with password

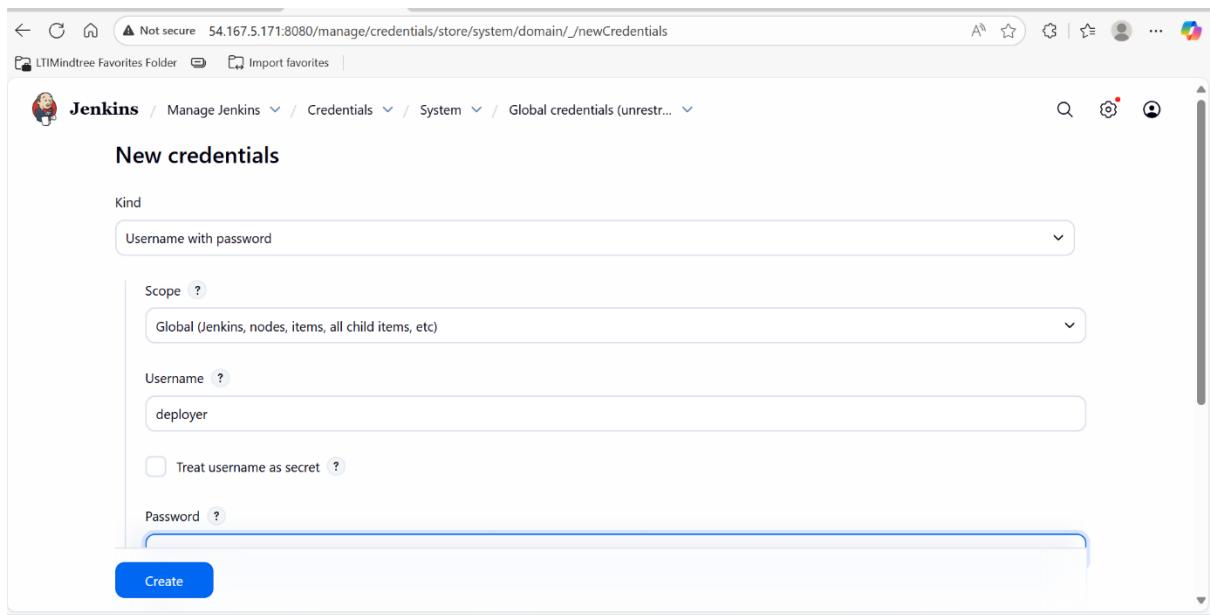
Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: deployer

Treat username as secret

Password

Create



Global credentials has been created for the tomcat server.

Not secure 54.167.5.171:8080/job/sample-job/configure

LTMindtree Favorites Folder Import favorites

Jenkins / sample-job / Configuration

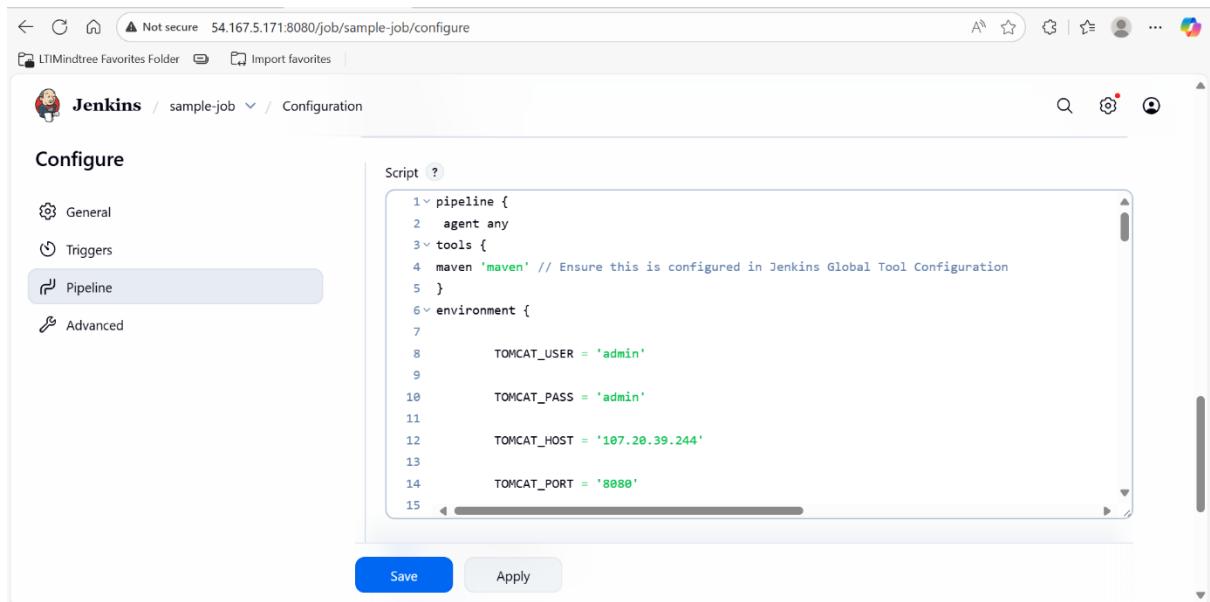
Configure

General Triggers Pipeline **Pipeline** Advanced

Script

```
1 pipeline {  
2   agent any  
3   tools {  
4     maven 'maven' // Ensure this is configured in Jenkins Global Tool Configuration  
5   }  
6   environment {  
7     TOMCAT_USER = 'admin'  
8     TOMCAT_PASS = 'admin'  
9     TOMCAT_HOST = '107.20.39.244'  
10    TOMCAT_PORT = '8080'  
11  }
```

Save Apply



Configure

General Triggers Pipeline Advanced

```
15
16      }
17
18  stages {
19    stage('Clone Repository') {
20      steps {
21        git branch: 'main', url: 'https://github.com/ankithapatturi/devops-project.git'
22      }
23    }
24  }
25
26  }
27
28  }
29
```

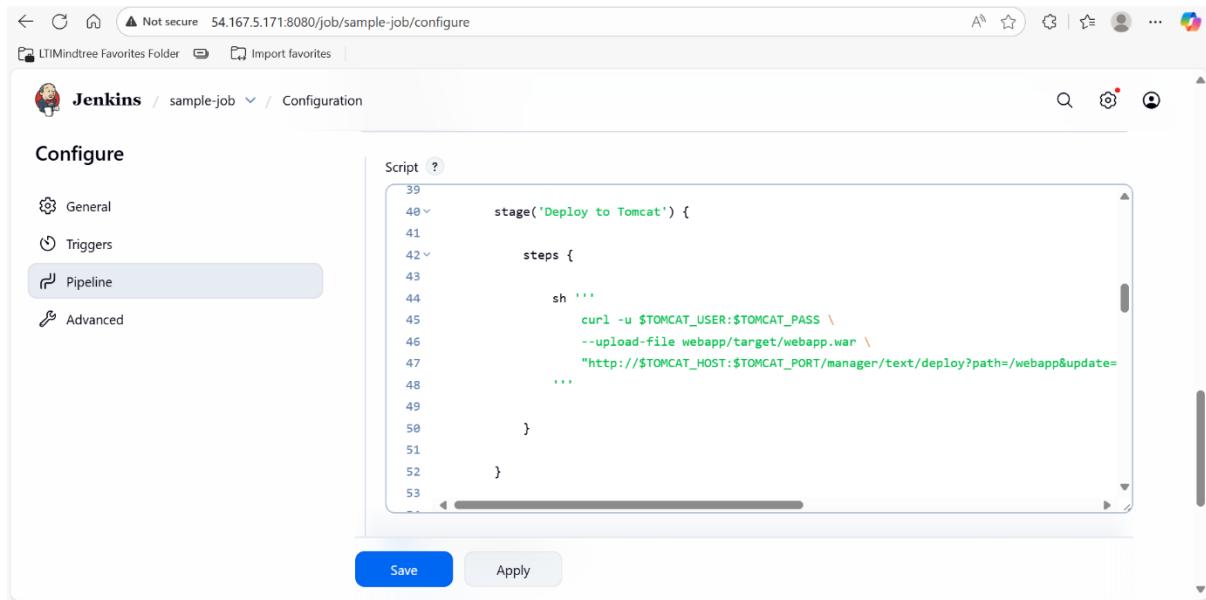
Save Apply

Configure

General Triggers Pipeline Advanced

```
26
27
28  stage('Clone Repository') {
29    steps {
30      git branch: 'main', url: 'https://github.com/ankithapatturi/devops-project.git'
31    }
32  }
33
34  stage('Build with Maven') {
35    steps {
36      sh 'mvn clean package -Dmaven.test.failure.ignore=true'
37    }
38  }
39
40  stage('Deploy to Tomcat') {
41    steps {
42    }
43}
```

Save Apply



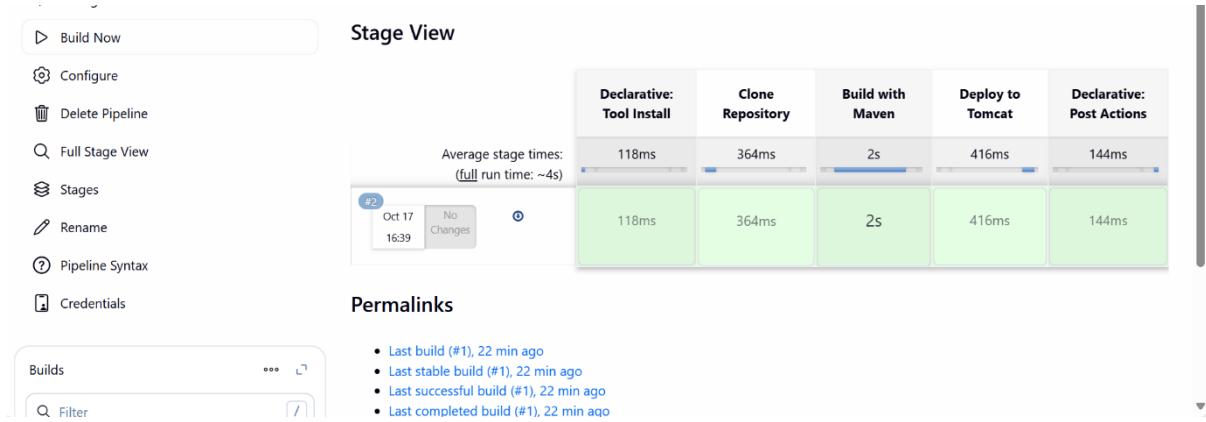
The screenshot shows the Jenkins Pipeline configuration page for a job named "sample-job". The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area contains a "Script" section with the following Groovy code:

```
39
40    stage('Deploy to Tomcat') {
41
42        steps {
43
44            sh '''
45                curl -u $TOMCAT_USER:$TOMCAT_PASS \
46                --upload-file webapp/target/webapp.war \
47                "http://$TOMCAT_HOST:$TOMCAT_PORT/manager/text/deploy?path=/webapp&update="
48            '''
49
50        }
51
52    }
53
```

At the bottom are "Save" and "Apply" buttons.

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

Build it and test it.



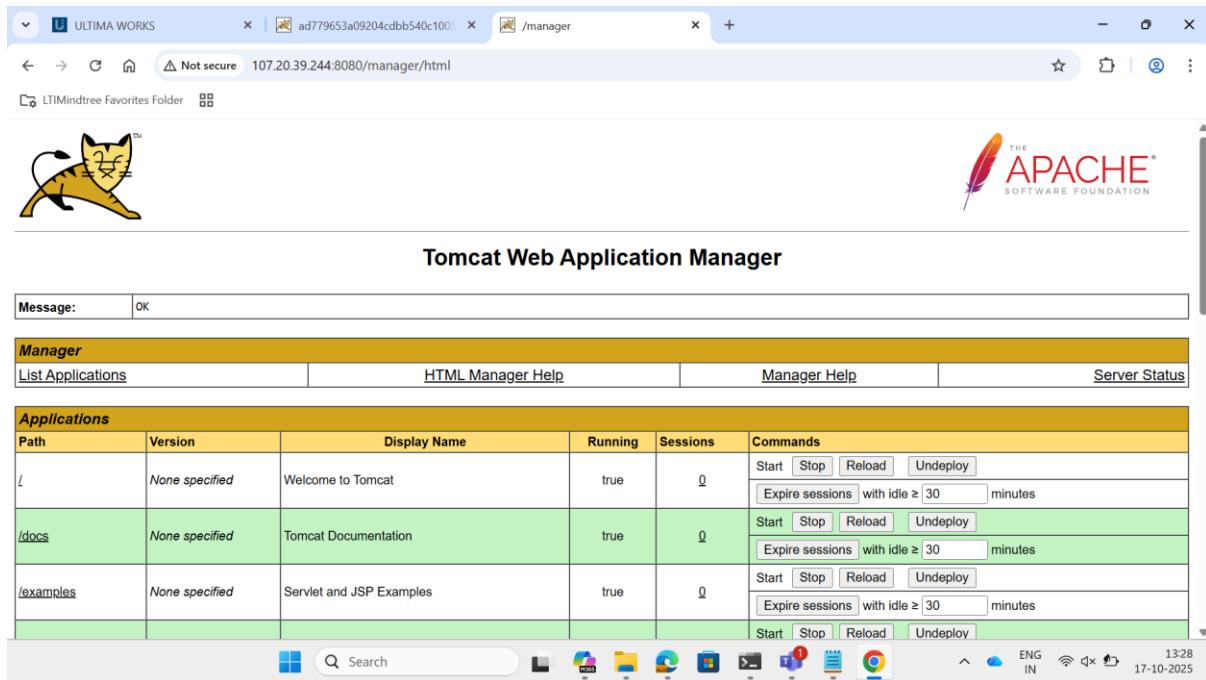
The screenshot shows the Jenkins Stage View for a pipeline. On the left is a sidebar with options: Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, Pipeline Syntax, and Credentials. The main area is titled "Stage View" and displays a table of average stage times:

	Declarative: Tool Install	Clone Repository	Build with Maven	Deploy to Tomcat	Declarative: Post Actions
Average stage times: (full run time: ~4s)	118ms	364ms	2s	416ms	144ms
Oct 17 16:39	118ms	364ms	2s	416ms	144ms

Below the table is a "Permalinks" section with a list of recent builds:

- Last build (#1), 22 min ago
- Last stable build (#1), 22 min ago
- Last successful build (#1), 22 min ago
- Last completed build (#1), 22 min ago

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.

```

root@jenkins:~ x root@tomcat:~ x root@docker:~/deploy x root@ip-172-31-30-41:~ x
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> cd Downloads
PS C:\Users\10844392\Downloads> ssh -i "project-key.pem" ec2-user@ec2-3-89-33-61.compute-1.amazonaws.com
The authenticity of host 'ec2-3-89-33-61.compute-1.amazonaws.com (3.89.33.61)' can't be established.
ED25519 key fingerprint is SHA256:z1JDkJise2/SMfEnh1MHPDRm7Sv5yp5hXL58UCGXuWk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-89-33-61.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
~\_ #####
~~ \####\_
~~ \###_
~~ \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '-->
~~ / \
~~ .--. / \
~~ /_/
~~ /m'/

[ec2-user@ip-172-31-19-207 ~]$ sudo su -
[root@ip-172-31-19-207 ~]# hostnamectl set-hostname docker.example.com
[root@ip-172-31-19-207 ~]# bash
[root@docker ~]# yum install docker -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
=====
Package           Architecture      Version          Repository      Size
=====
Installing:
  docker          x86_64          25.0.13-1.amzn2023.0.1    amazonlinux   46 M
=====
```

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

```

root@jenkins:~ root@tomcat:~ root@docker:~/deploy root@ip-172-31-30-41:~
libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 libnfnl-1.0.1-19.amzn2023.0.2.x86_64 libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64 runc-1.3.1-1.amzn2023.0.1.x86_64

Complete!
[root@docker ~]# 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:c9qrRtaJ0e7TzCdy/7mayjMBSDryto2m0dT6Vq39w root@docker.example.com
The key's randomart image is:
+--[RSA 3072]----+
| . . .
| o . o
| . + .
| . S = ...
| + B ..oo..
| . + o = Bo*oo
| .=.. . 0.0 Eo
| .o++ ..+.+o*+o
+---[SHA256]-----+
[root@docker ~]#
[root@docker ~]# cat /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAAQGCrZfRIIv+VV78X2a9sJ6xyoHaqoMQxp+NajMY5u5P6Z5C1aJEQpXT/DA7TaT1z1rF3Tk3n2W50NdUlOCYYKcM0yC05jqG0Mq
2Y6+ndaza5k3zit7p705gV1lJTtAzwsmuDm0uyaPhutn+buXzqcw6MqbVqgXIAzUy8bijj/O/WIXaHu+H1Cq4dw+u6725xcQwIyAPXP2jCSX+M0+GCgQV2Upq1YN1Hhwoc/Fcg
i075ly2T0vUh6Ed1npdrMuM7Qev6iyMMMc5WDGs8nkgrpcf1NMrkldu91/2W0jx4C/vk4hsjG9ZyYoUMZJi+ICSmhvPfp7GlvFwdxAYxt/0jV3FFvcmb72qjTP1m97Ikf1r
4hxcLkw0X1RjzuIu28VL1skD858guzsloB2x3yNF0Lm1KBg290ueEU4briYXWRAIF15LvuBMIVdx9nbu0DYgqqTfp05J2a1cS4GrMX6dU0/Xt/Wc3ee6EGGIpSvB4GIY8DYLb
mGo053ob1P6WOAfVNNE= root@docker.example.com
[root@docker ~]# cd .ssh/

```

Exchange SSH keys between the docker and 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]----+

```

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

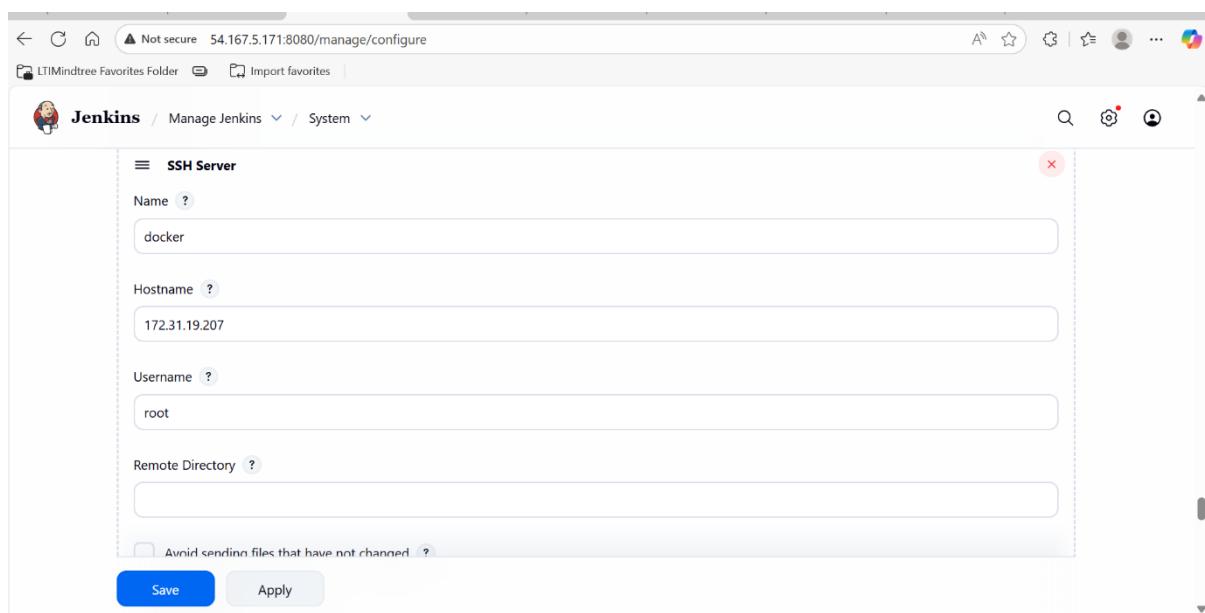
#PubkeyAuthentication yes

#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.

```

root@jenkins:~| root@tomcat:~| root@docker:~/deploy| root@ip-172-31-30-41:~|
valid_lft forever preferred_lft forever
[root@docker ~]# vim /etc/hosts
[root@docker ~]# aws configure
AWS Access Key ID [None]: AKIAQUCCBJX5YCONFGEK
AWS Secret Access Key [None]: mBNkV4T/L4MfoyWEEvrVrK4zq3ZJ0SF/0t19esI
Default region name [None]: us-east-1
Default output format [None]: table
[root@docker ~]# aws ecr create-repository --repository-name tomcat-webapp
+-----+-----+
|          CreateRepository          |
+-----+-----+
|          repository               |
+-----+-----+
| createdAt | 2025-10-17T04:20:25.465000+00:00
| imageTagMutability | MUTABLE
| registryId | 043088170491
| repositoryArn | arn:aws:ecr:us-east-1:043088170491:repository/tomcat-webapp
| repositoryName | tomcat-webapp
| repositoryUri | 043088170491.dkr.ecr.us-east-1.amazonaws.com/tomcat-webapp
+-----+-----+
|          encryptionConfiguration   |
+-----+-----+
| encryptionType | AES256
+-----+-----+
|          imageScanningConfiguration |
+-----+-----+
| scanOnPush | False
+-----+-----+
[root@docker ~]# ping jenkins.example.com
PING jenkins.example.com (172.31.16.116) 56(84) bytes of data.
64 bytes from jenkins.example.com (172.31.16.116): icmp_seq=1 ttl=127 time=1.46 ms
64 bytes from jenkins.example.com (172.31.16.116): icmp_seq=2 ttl=127 time=0.532 ms
64 bytes from jenkins.example.com (172.31.16.116): icmp_seq=3 ttl=127 time=0.581 ms

```

Repository name	URI	Created at	Tag immutability	Encryption type
tomcat-webapp	043088170491.dkr.ecr.us-east-1.amazonaws.com/tomcat-webapp	October 17, 2025, 09:50:25 (UTC+05.5)	Mutable	AES-256

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.

Jenkins / sample-job / Configuration

Configure

- General
- Triggers
- Pipeline**
- Advanced

```

52      }
53
54      stage('Deploy to Docker Host') {
55
56          steps {
57
58              sshPublisher(publishers: [
59
60                  sshPublisherDesc(
61
62                      configName: 'docker',
63
64                      transfers: [
65
66                          sshTransfer(
67
68
69
70
71
72
73
74
75
76
77
78
79

```

Save Apply

Jenkins / sample-job / Configuration

Configure

- General
- Triggers
- Pipeline**
- Advanced

```

65      sshTransfer(
66
67
68      sourceFiles: '**/*',
69
70      removePrefix: '',
71
72      remoteDirectory: 'deploy',
73
74      execCommand: '''
75          cd deploy
76          aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-
77          docker build -t tomcat-webapp .
78          docker tag tomcat-webapp:latest 043088170491.dkr.ecr.us-east-1.amazonaws.com/tomca
79

```

Save Apply

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

Rename Pipeline Syntax Credentials

Stage View

Average stage times:
(full run time: ~33s)

Declarative: Tool Install	Clone Repository	Build with Maven	Deploy to Tomcat	Deploy to Docker Host	Declarative: Post Actions
90ms	398ms	2s	402ms	28s	94ms
Oct 17 17:16	No Changes				
90ms	398ms	2s	402ms	28s	94ms

Latest Test Result (no failures)

Permalinks

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

```
root@jenkins:~          x | root@tomcat:~          x | root@docker:~/deploy x | root@ip-172-31-30-41:~ x | + | - | 
~\_\_ #####_      Amazon Linux 2023
~~\_\_ #####\
~~ \###|
~~ \#/   __ https://aws.amazon.com/linux/amazon-linux-2023
~~ V~'  '-->
~~ /
~~ .-' /_/
~~ /_/
~/m/` 

[ec2-user@ip-172-31-30-41 ~]$ sudo su -
[root@ip-172-31-30-41 ~]# hostnamectl set-hostname k8s.example.com
[root@ip-172-31-30-41 ~]# bash
[root@k8s ~]# apt-get update -y
apt install unzip -y
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws configure
Install EKS Tool
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
Install Kubectl
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/releas
se/stable.txt)/bin/linux/amd64/kubectl
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
bash: apt-get: command not found
bash: apt: command not found
% Total    % Received % Xferd  Average Speed   Time   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 59.3M  100 59.3M    0     0  349M      0  --:--:--  --:--:-- 351M
```

Install k8s in terminal and also install cluster.

```
[root@k8s ~]# eksctl create cluster --name milestone-2 --region us-east-1 --version 1.32 --node-type t3.small --nodes 2 --nodes-min 2 --nodes-max 4 --ssh-access --ssh-public-key /root/.ssh/id_rsa.pub
2025-10-17 05:52:50 [ℹ️] eksctl version 0.215.0
2025-10-17 05:52:50 [ℹ️] using region us-east-1
2025-10-17 05:52:50 [ℹ️] setting availability zones to [us-east-1d us-east-1f]
2025-10-17 05:52:50 [ℹ️] subnets for us-east-1d - public:192.168.0.0/19 private:192.168.64.0/19
2025-10-17 05:52:50 [ℹ️] subnets for us-east-1f - public:192.168.32.0/19 private:192.168.96.0/19
2025-10-17 05:52:50 [ℹ️] nodegroup "ng-4dadalc5" will use "" [AmazonLinux2023/1.32]
2025-10-17 05:52:50 [ℹ️] using SSH public key "/root/.ssh/id_rsa.pub" as "eksctl-milestone-2-nodegroup-ng-4dadalc5-dc:f3:df:b7:a9:11:10:f7:6e:5c:37:24:7c:51:e7"
2025-10-17 05:52:50 [!] Auto Mode will be enabled by default in an upcoming release of eksctl. This means managed node groups and managed networking add-ons will no longer be created by default. To maintain current behavior, explicitly set 'autoModeConfig.enabled: false' in your cluster configuration. Learn more: https://eksctl.io/usage/auto-mode/
2025-10-17 05:52:50 [ℹ️] using Kubernetes version 1.32
2025-10-17 05:52:50 [ℹ️] creating EKS cluster "milestone-2" in "us-east-1" region with managed nodes
2025-10-17 05:52:50 [ℹ️] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-10-17 05:52:50 [ℹ️] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=milestone-2'
2025-10-17 05:52:50 [ℹ️] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "milestone-2" in "us-east-1"
2025-10-17 05:52:50 [ℹ️] CloudWatch logging will not be enabled for cluster "milestone-2" in "us-east-1"
2025-10-17 05:52:50 [ℹ️] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=milestone-2'
2025-10-17 05:52:50 [ℹ️] default addons metrics-server, vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2025-10-17 05:52:50 [ℹ️]
2 sequential tasks: { create cluster control plane "milestone-2",
  2 sequential sub-tasks: {
    2 sequential sub-tasks: {
      1 task: { create addons },
      wait for control plane to become ready,
    },
    create managed nodegroup "ng-4dadalc5",
  }
}
```

Create deployment file and specify image URL properly.

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

```
[root@k8s ~]# kubectl get nodes
NAME           STATUS   ROLES     AGE      VERSION
ip-192-168-33-9.ec2.internal   Ready    <none>   2m56s   v1.32.9-eks-113cf36
ip-192-168-6-248.ec2.internal   Ready    <none>   2m56s   v1.32.9-eks-113cf36
[root@k8s ~]# vim deployment.yaml
[root@k8s ~]# vim service.yaml
[root@k8s ~]# vim deployment.yaml
[root@k8s ~]# vim /etc/ssh/sshd_config
[root@k8s ~]# systemctl restart sshd
[root@k8s ~]# systemctl enable sshd
[root@k8s ~]# curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
  % Total    % Received % Xferd  Average Speed   Time   Time   Current
                                 Dload  Upload   Total Spent   Left  Speed
100  138  100  138    0     0  3408      0  --::-- --::-- 3450
100 57.7M  100 57.7M    0     0  304M      0  --::-- --::-- 559M
[root@k8s ~]# kubectl version --client
Client Version: v1.34.1
Kustomize Version: v5.7.1
[root@k8s ~]# systemctl restart sshd
[root@k8s ~]# systemctl enable sshd
[root@k8s ~]# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@k8s ~]# yum install kubectl
Amazon Linux 2023 Kernel Livepatch repository
No match for argument: kubectl
Error: Unable to find a match: kubectl
[root@k8s ~]# kubectl --version
228 kB/s | 26 kB  00:00
```

The deployment file should be in this format.

```
root@jenkins:~          root@tomcat:~          root@docker:~/deploy      root@k8s:~          + - X
apiVersion: apps/v1
kind: Deployment
metadata:
  name: regapp-deployment
  labels:
    app: regapp

spec:
  replicas: 2
  selector:
    matchLabels:
      app: regapp

  template:
    metadata:
      labels:
        app: regapp
    spec:
      containers:
        - name: regapp
          image: 043088170491.dkr.ecr.us-east-1.amazonaws.com/tomcat-webapp:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
~
~
~"deployment.yaml" 29L, 536B
2,8          All
```

The Service file should be in this format.

```
root@jenkins:~          root@tomcat:~          root@docker:~/deploy      root@k8s:~          + - X
apiVersion: v1
kind: Service
metadata:
  name: regapp-service
  labels:
    app: regapp
spec:
  selector:
    app: regapp

  ports:
    - port: 8080
      targetPort: 8080

  type: LoadBalancer
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~"service.yaml" 15L, 196B
15,20          All
```

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

The screenshot shows the Jenkins 'New credentials' creation interface. The 'Kind' dropdown is set to 'AWS Credentials'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. There are fields for 'ID' and 'Description', both of which are currently empty. A blue 'Create' button is at the bottom.

Create k8s SSH server in Jenkins

The screenshot shows the Jenkins 'System' configuration interface. A new SSH server is being created with the following details:

- Name: kubernetes
- Hostname: 172.31.30.41
- Username: root
- Remote Directory: (empty)
- A checkbox labeled 'Avoid sending files that have not changed' is unchecked.

At the bottom are 'Save' and 'Apply' buttons.

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

Apply groovy script as this is the declarative approach.

Configure

General Triggers Pipeline Advanced

```
82
83
84
85
86
87
88
89
90
91
92
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
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     }
118 }
```

Save Apply

Configure

General Triggers Pipeline Advanced

```
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
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     }
118 }
```

Save Apply

Configure

General Triggers Pipeline Advanced

```
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118 }
```

Save Apply

Not secure 54.167.5.171:8080/job/sample-job/configure

LTMindtree Favorites Folder Import favorites

Jenkins / sample-job / Configuration

Configure

General Triggers Pipeline **Advanced**

```
113     usePromotionTimestamp: false,
114     verbose: true
115   )
116 }
117 }
118
119
120
121 post {
122
123 success {
124
125   junit '**/target/surefire-reports/TEST-*.xml'
126
127
128
129
130
131
132
133
134 }
```

Save Apply

Not secure 54.167.5.171:8080/job/sample-job/configure

LTMindtree Favorites Folder Import favorites

Jenkins / sample-job / Configuration

Configure

General Triggers Pipeline **Advanced**

```
121 post {
122
123 success {
124
125   junit '**/target/surefire-reports/TEST-*.xml'
126
127   archiveArtifacts artifacts: '**/target/*.war', fingerprint: true
128
129
130
131
132
133
134 }
```

Save Apply

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

Not secure 54.167.5.171:8080/job/sample-job/ LTMindtree Favorites Folder Import favorites

Jenkins / sample-job

	Declarative: Tool Install	Clone Repository	Build with Maven	Deploy to Tomcat	Deploy to Docker Host	Deploy to EKS	Declarative: Post Actions
Average stage times: (full run time: ~13s)	78ms	321ms	2s	414ms	2s	5s	110ms
#22 Oct 17 13:16 1 commit	79ms	332ms	3s	409ms	3s	5s	100ms
#21 Oct 17 13:02 No Changes	73ms	269ms	3s	412ms	3s	5s	98ms
#20 Oct 17 12:57 No Changes	68ms	284ms	3s	408ms	2s	5s	88ms
#19 Oct 17 12:55 1 commit	80ms	315ms	2s	411ms	2s	5s	99ms

Builds

Filter

Today

- #22 7:46AM
- #21 7:32AM
- #20 7:27AM
- #19 7:25AM
- #18 7:23AM
- #17 7:21AM
- #16 7:16AM

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

Not secure 54.167.5.171:8080/job/sample-job/ LTMindtree Favorites Folder Import favorites

Jenkins / sample-job

Status sample-job Add description

</> Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

GitHub Hook Log

Git Polling Log

Credentials

Last Successful Artifacts

webapp.war 2.32 KiB view

Test Result Trend

Passed Skipped Failed

Stage View

Declarative: Tool Install	Clone Repository	Build with Maven	Deploy to Tomcat	Deploy to Docker Host	Deploy to EKS	Declarative: Post Actions
Average stage times: 78ms	321ms	2s	414ms	2s	5s	110ms

This indicates the status of build.

The screenshot shows the Jenkins Pipeline Overview for job sample-job #22. The pipeline consists of several steps: Tool Install (61ms), Clone Repository (0.31s), Build with Maven (2.9s), Deploy to Tomcat (0.39s), Deploy to Docker Host (3.7s), Deploy to EKS (5.6s), and Post Actions (83ms). The Post Actions step includes tasks such as Archiving JUnit-formatted test results and Archiving artifacts. The build was started 2 hours and 2 minutes ago and took 83ms.

The post actions of the build are mentioned above.

The screenshot shows the Jenkins Home page. It displays the Build History for the sample-job project, which has no builds in the queue. The Build Executor Status shows 0/2 executors available. The page also includes links for REST API and Jenkins version 2.528.1.

This indicates the sample-job project is successfully done.

Jenkins / sample-job / #22 / Pipeline Overview

Graph

```
graph LR; Start((Start)) --> ToolInstall[Tool Install]; ToolInstall --> CloneRepository[Clone Repository]; CloneRepository --> BuildWithMaven[Build with Maven]; BuildWithMaven --> DeployToTomcat[Deploy to Tomcat]; DeployToTomcat --> DeployToDockerHost[Deploy to Docker Host]; DeployToDockerHost --> DeployToEKS[Deploy to EKS]; DeployToEKS --> PostActions[Post Actions]; PostActions --> End((End))
```

Post Actions

- Tool Install 61ms
- Clone Repository 0.31s

Archive JUnit-formatted test results **/target/surefire-reports/TEST-*.xml >

Archive the artifacts >

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

Amazon Elastic Container Registry

Private registry

Images (3)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest	Image	October 17, 2025, 13:16:26 (UTC+05.5)	156.86	<input type="checkbox"/> Copy URI	<input type="checkbox"/> sha256:4d67fe1...	October 17, 2025 13:16:31 (UTC+05)
-	Image	October 17, 2025, 13:02:22 (UTC+05.5)	156.86	<input type="checkbox"/> Copy URI	<input type="checkbox"/> sha256:1de9aaec...	October 17, 2025 13:02:27 (UTC+05)

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

This is my Devops Project..I'm Ankitha.Patturi (10844392)

Please fill in this form to create an account.

Enter Name	Enter Full Name
Enter mobile	Enter mobile number
Enter Email	Enter Email
Password	Enter Password
Repeat Password	Repeat Password

By creating an account you agree to our [Terms & Privacy](#).

[Register](#)

Already have an account? [Sign in](#).

Thank You

bye

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=true"
        """
    }
}

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.

