EXPERIMENT NO. 5

AIM : To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.

THEORY :

Jenkins Pipeline

Jenkins Pipeline is a combination of plugins that supports integration and implementation of continuous delivery pipelines. It has an extensible automation server to create simple and complex delivery pipelines as code via pipeline DSL. A Pipeline is a group of events interlinked with each other in a sequence.

In a Jenkins pipeline, every job or event has some sort of dependency on at least one or more events.

The benefits of using Jenkins are:

- You can create pipelines automatically for all branches and execute pull requests with just one Jenkins.
- You can review your Jenkins code on the pipeline
- You can audit your Jenkins pipeline
- This is the singular source for your pipeline and can be modified by multiple users.

Maven Jenkins

Maven is used to define project structure, dependencies, build, and test management.

Using pom. xml(Maven) you can configure dependencies needed for building testing and running code.

Maven automatically downloads the necessary files from the repository while building the project.

Purpose of Maven :

- A maven is a build tool designed to manage dependencies and the software lifecycle. It is also designed to work with plugins that allow users to add other tasks to the standard compile, test, package, install, deploy tasks.
- Jenkins is designed for the purpose of implementing Continuous Integration (CI). It checks code out of a repository, builds and packages it, and sends it out to a server for testing – automatically. Jenkins can use Maven as its build tool.

Gradle Jenkins :

Gradle is managed as another tool inside Jenkins (the same way as Ant or Maven), including support for automatic installation and a new build step is provided to execute Gradle tasks.

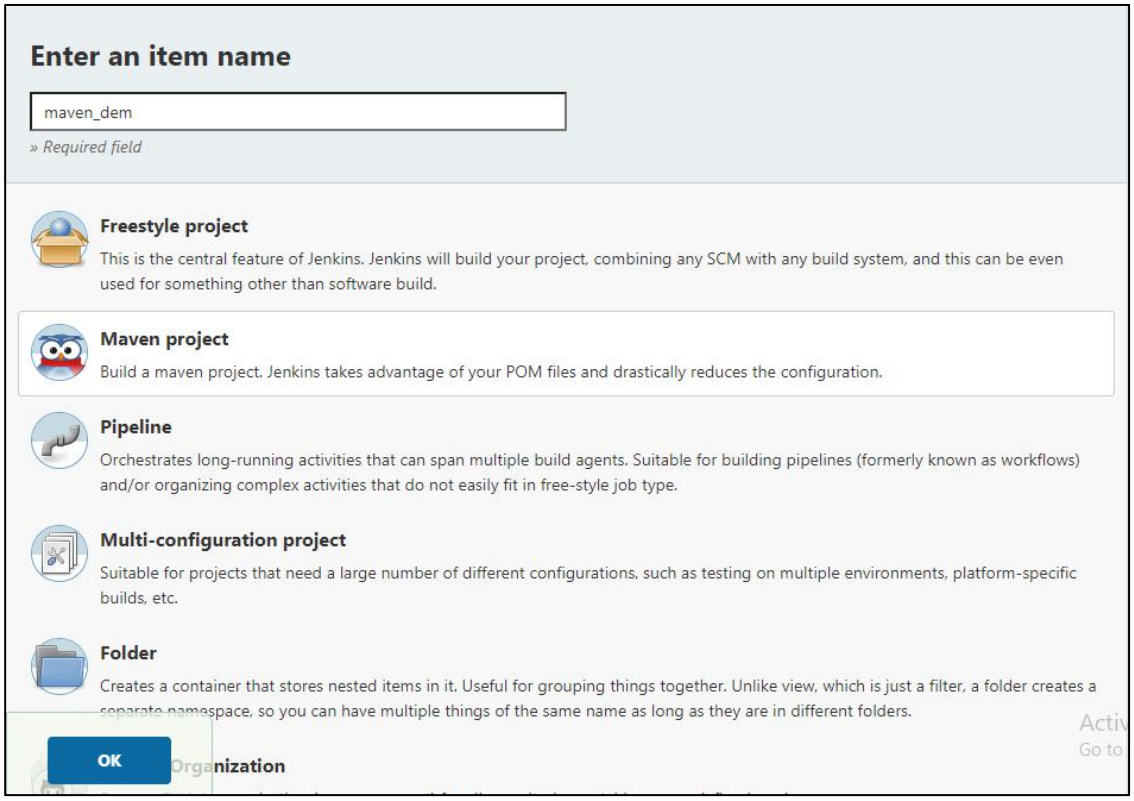
It also allows detecting Build Scans in arbitrary console logs, for Maven and Gradle builds and display them in the Jenkins UI. It is a powerful build tool for the JVM.

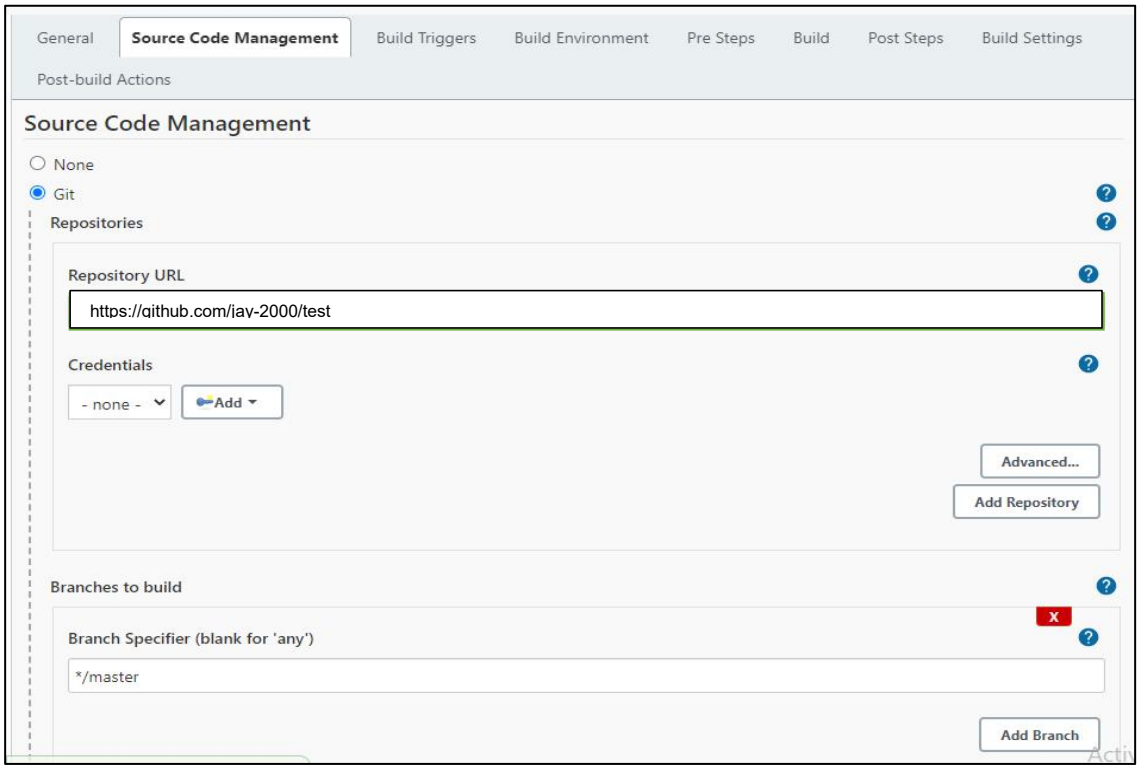
It primarily focuses on build automation and supports multi-language development.

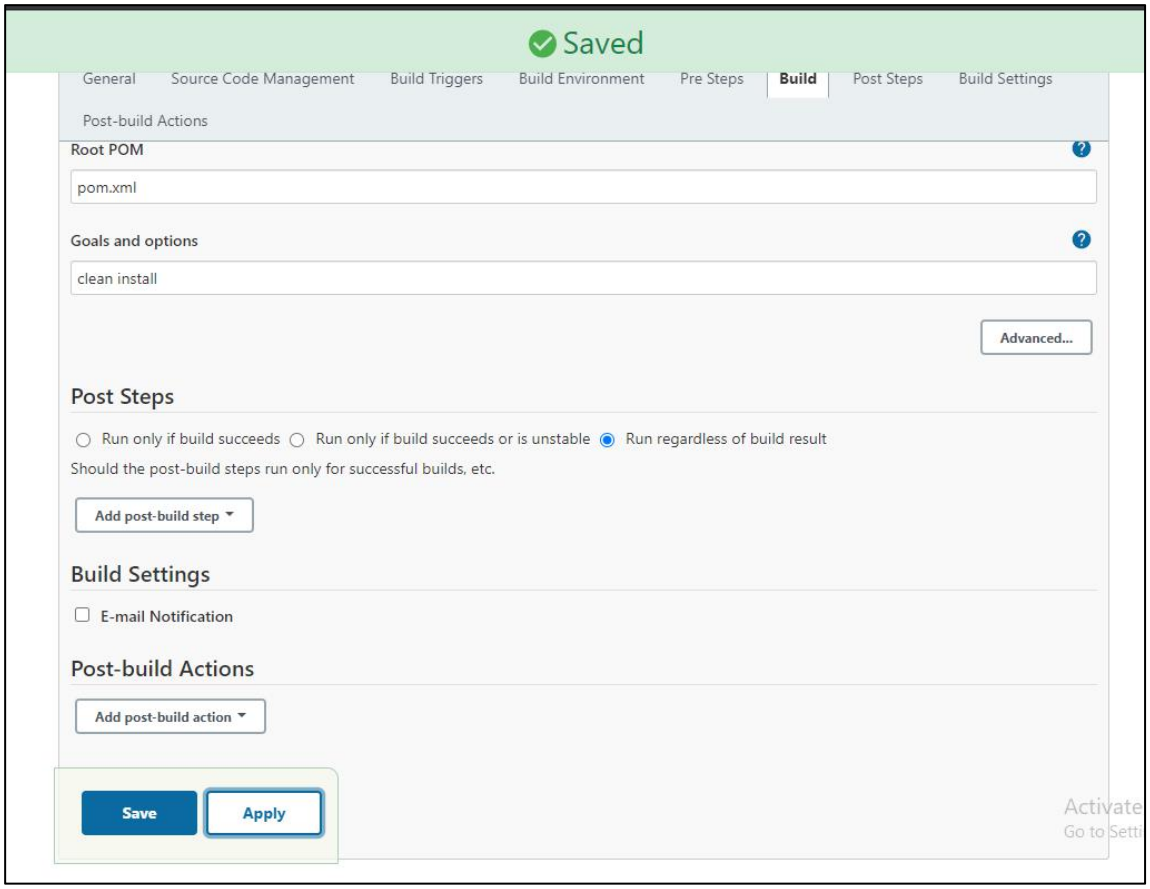
If we are building, testing, publishing, and deploying software on any platform, Gradle provides a flexible model to support the entire development lifecycle from compiling and deploying the project.

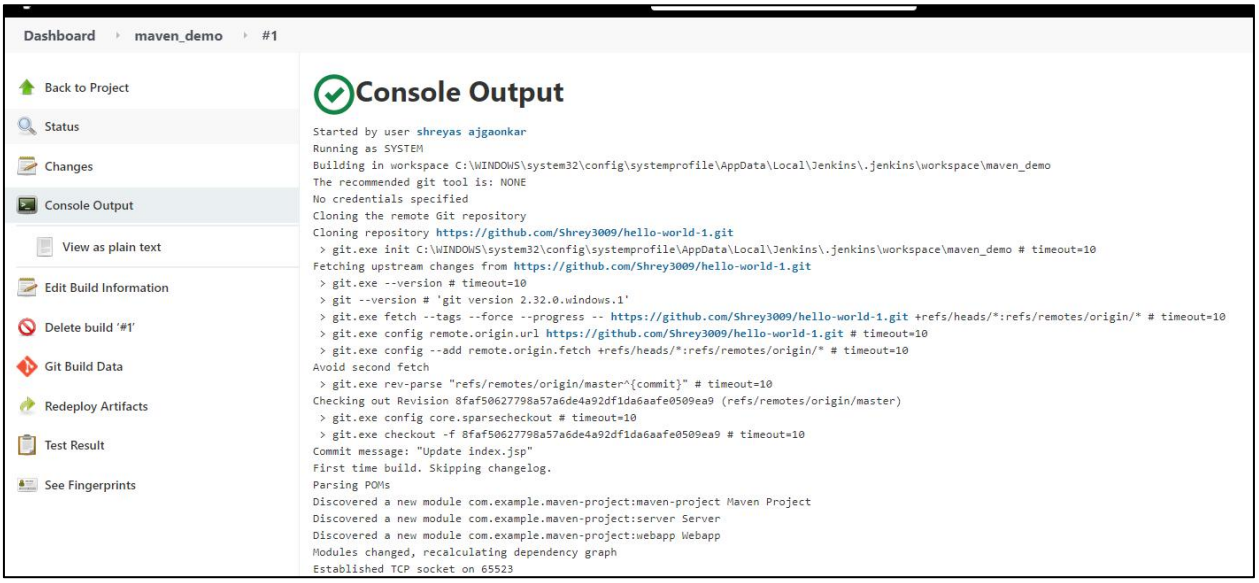
OUTPUT :

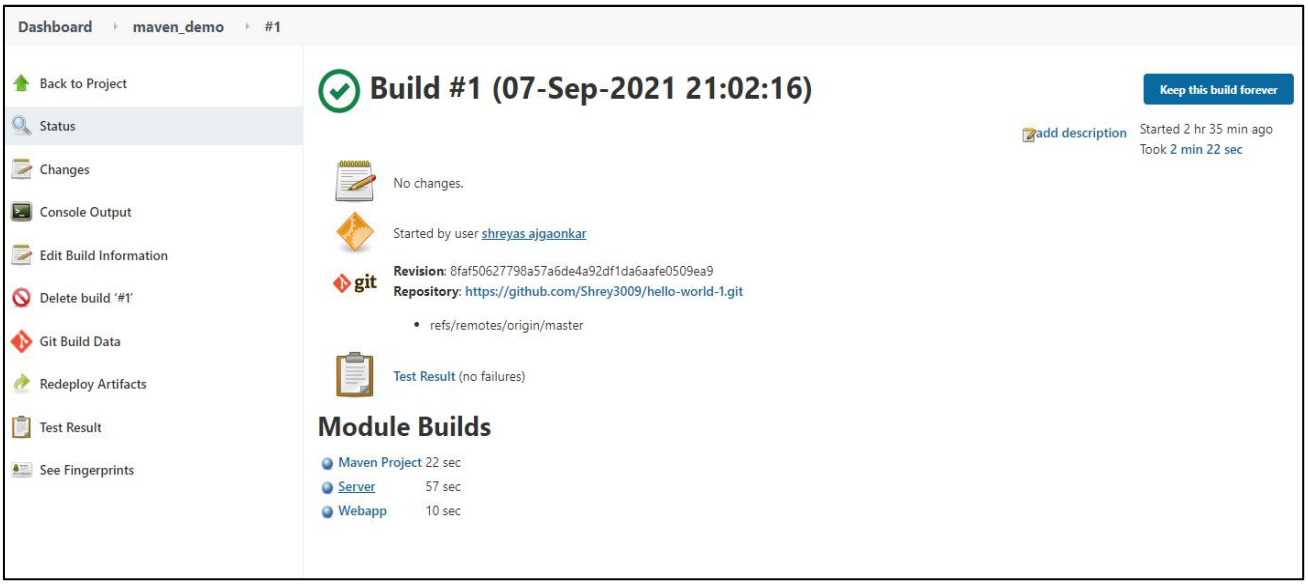
Installing Maven in Jenkins :



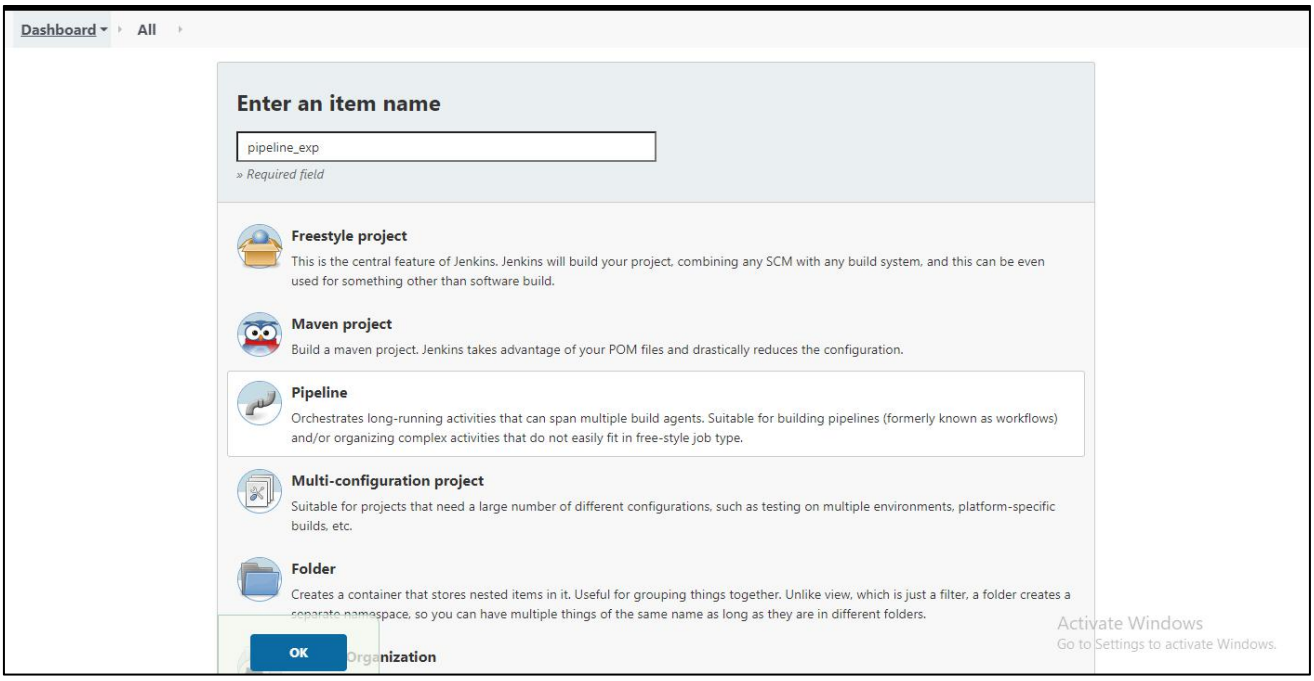



Saved
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Root POM
pom.xml
Goals and options
clean install
Advanced...
Post Steps
Run only if build succeeds
Run only if build succeeds or is unstable
Run regardless of build result
Should the post-build steps run only for successful builds, etc.
Add post-build step
Build Settings
E-mail Notification
Post-build Actions
Add post-build action
Save
Apply


Dashboard
maven_demo
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Git Build Data
Redeploy Artifacts
Test Result
See Fingerprints
Console Output
Started by user shreyas ajgaonkar
Running as SYSTEM
Building in workspace C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\maven_demo
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Shrey3009/hello-world-1.git
> git.exe init C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\maven_demo # timeout=10
Fetching upstream changes from https://github.com/Shrey3009/hello-world-1.git
> git.exe --version # timeout=10
> git --version # 'git version 2.32.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/Shrey3009/hello-world-1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/Shrey3009/hello-world-1.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 8faf50627798a57a6de4a92df1da6aafe0509ea9 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 8faf50627798a57a6de4a92df1da6aafe0509ea9 # timeout=10
Commit message: "Update index.jsp"
First time build. Skipping changelog.
Parsing POMs
Discovered a new module com.example.maven-project:maven-project Maven Project
Discovered a new module com.example.maven-project:server Server
Discovered a new module com.example.maven-project:webapp Webapp
Modules changed, recalculating dependency graph
Established TCP socket on 65523



Pipeline of jobs in Jenkins :



General | Build Triggers | Advanced Project Options | Pipeline

Description

pipeline example

[Plain text] Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] This project is parameterised
- [ ] Throttle builds

General | Build Triggers | Advanced Project Options | **Pipeline**

Definition

Pipeline script

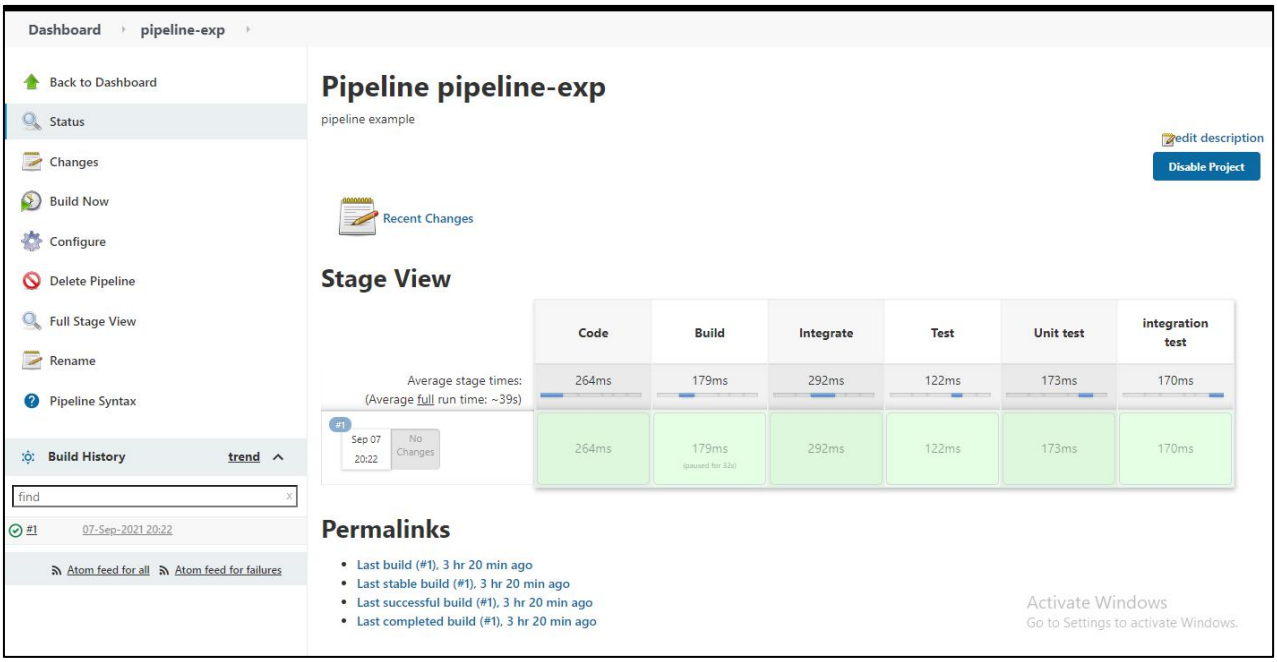
Script

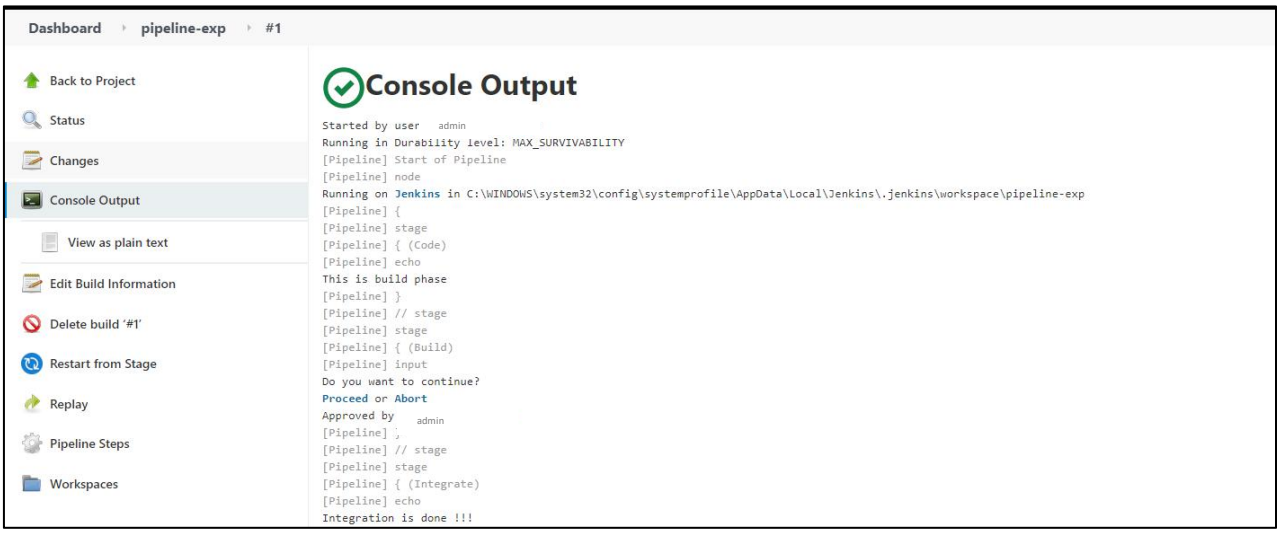
```
pipeline {
    agent any

    stages {
        stage('Code') {
            steps{
                    echo 'This is build phase'
            }

        }
        stage('Build'){
            steps{
                input('Do you want to continue?')
            }
        }

        stage('Integrate')  {
            when{
                not{
                    branch "master"
                }
            }
            steps{
                echo "Integration is done !!!"
            }
        }
        stage('Test'){
            parallel{
            stage('Unit test'){
                steps{
                        echo"test done"
                }
            }
           stage('integration test'){
               steps{
```


Dashboard
pipeline-exp
Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
find
#1
07-Sep-2021 20:22
Atom feed for all
Atom feed for failures
Pipeline pipeline-exp
pipeline example
edit description
Disable Project
Recent Changes
Stage View
Code
Build
Integrate
Test
Unit test
integration test
Average stage times:
(Average full run time: ~39s)
264ms
179ms
292ms
122ms
173ms
170ms
#1
Sep 07
20:22
No Changes
Permalinks
Last build (#1), 3 hr 20 min ago
Last stable build (#1), 3 hr 20 min ago
Last successful build (#1), 3 hr 20 min ago
Last completed build (#1), 3 hr 20 min ago
Activate Windows
Go to Settings to activate Windows.


Dashboard
pipeline-exp
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Restart from Stage
Replay
Pipeline Steps
Workspaces
Console Output
Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\pipeline-exp
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] echo
This is build phase
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] input
Do you want to continue?
Proceed or Abort
Approved by admin
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Integrate)
[Pipeline] echo
Integration is done !!!

CONCLUSION : Thus we successfully studied about pipeline of jobs using Maven / Gradle / Ant in Jenkins, created a pipeline script to Test and deploy an application over the tomcat server.