

Name: Manas Mulchandani

Roll no: 64

Batch: T13

Experiment 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:

Continuous Integration (CI) involves automatically testing and building software in a shared repository, allowing developers to frequently integrate their work. In this context, Jenkins is used as the CI server to build, test, and deploy an application. The pipeline you will set up will utilize **Maven**, **Gradle**, or **Ant** to automate the process, and **Tomcat** will serve as the application server for deployment.

1. Understanding Jenkins Pipeline

A **Jenkins pipeline** is a series of automated processes that include building, testing, and deploying software. These processes are defined as **jobs** in Jenkins. A pipeline script can be created in **Jenkinsfile**, which can be version-controlled along with the source code.

Jenkins offers two types of pipelines:

1. **Declarative Pipeline:** A more structured and easier-to-understand format, ideal for most use cases.

2. **Scripted Pipeline:** Offers more flexibility but requires knowledge of Groovy scripting. It is more suited for complex workflows.

In this case, we will use a **Declarative Pipeline** script that will:

- Build the application using **Maven**, **Gradle**, or **Ant**.
 - Run tests to ensure the application is working as expected.
 - Deploy the application to a **Tomcat** server.
-

2. Jenkins Pipeline Stages

A typical Jenkins pipeline for building, testing, and deploying an application consists of the following stages: **a. Build Stage**

In this stage, Jenkins will use **Maven**, **Gradle**, or **Ant** to compile and package the application. The choice of tool depends on the project's build system.

- **Maven:** Uses the pom.xml file to manage dependencies and build lifecycle.
 - **Gradle:** Uses build.gradle to define tasks and dependencies. •
 - **Ant:** Uses build.xml to define tasks and configurations
-

b. Test Stage

After building the application, the next stage involves running tests to ensure the application behaves as expected. This can include unit tests, integration tests, or other forms of testing depending on the setup of your project.

- **Maven:** Run tests using the mvn test command.
- **Gradle:** Run tests using the gradle test command.
- **Ant:** Run tests using ant test.

c. Deploy Stage

The final stage of the pipeline is the **deployment** phase. In this phase, Jenkins will deploy the built application to a **Tomcat server**.

Tomcat is a popular open-source Java web server and servlet container, used to deploy web applications. Jenkins can deploy the application by copying the generated WAR file to the Tomcat webapps directory, where it will be automatically deployed.

Example for Deploying to Tomcat:

1. Install Tomcat Plugin in Jenkins (Optional but recommended):

- Jenkins provides a **Deploy to Container** plugin, which allows you to deploy your applications to a **Tomcat** server directly from Jenkins. To install it:
 - + Go to **Manage Jenkins > Manage Plugins**.
 - + Search for **Deploy to Container** plugin and install it.

2. Configure Deployment Credentials:

- In Jenkins, configure the Tomcat server's deployment credentials (username, password) under **Manage Jenkins > Configure System**. Enter the URL of your Tomcat server (e.g., `http://localhost:8080/manager/text`) and the credentials for the Tomcat Manager application.

3. Deploy WAR File:

- After configuring Jenkins, you can use the following code in your pipeline to deploy the WAR file to Tomcat.

4. Configuring Tomcat for Automatic Deployment

For **automated deployment** to **Tomcat**, ensure that:

- **Tomcat Manager** is enabled and configured to accept deployment commands (for example, via HTTP requests).
- You provide the correct **Tomcat Manager credentials** (username and password) to Jenkins for deployment.

Implementation:

1. Freestyle App


Dashboard > All > New Item

New Item


Enter an item name

FreeStyleApp


Select an item type




Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.




Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

OK

Dashboard > FreeStyleApp > Configuration

Configure

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

General

Enabled ☒

Description

FreeStyle App

Plain text: Preview

☐ Discard old builds ?

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced ▾

Save Apply

Dashboard > FreeStyleApp > #1

Status


</> Changes

Console Output


Edit Build Information


Delete build '#1'


Timings



#1 (Apr 16, 2025, 2:35:03 PM)


 Add description

 Keep this build forever



Started by user [Vinay Dawani](#)


Started 7 sec ago



This run spent:

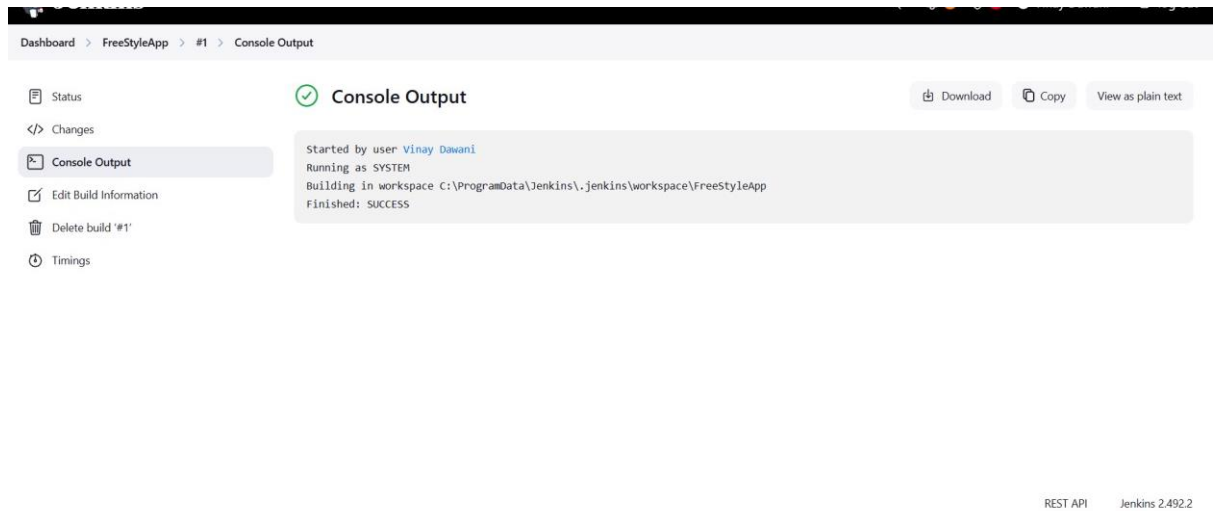
- 6 ms waiting;
- 0.1 sec build duration;
- 0.1 sec total from scheduled to completion.

Took 0.1 sec

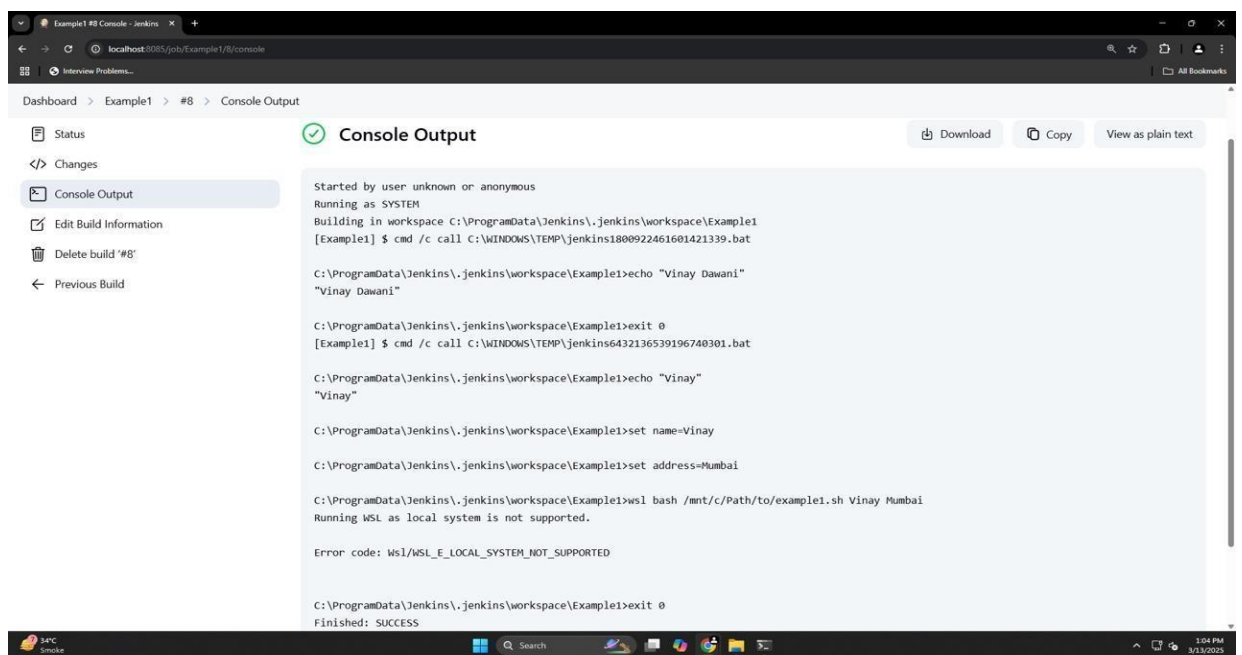


No changes.

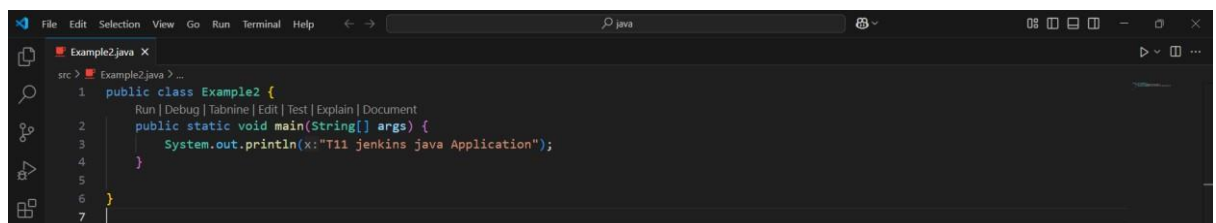
REST API Jenkins 2.492.2



2. Taking Parameters



3. Running java program



Dashboard > java_prog > Configuration

Configure

- General
- Source Code Management
- Triggers
- Environment
- Build Steps**
- Post-build Actions

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Execute Windows batch command ?

Command

[See the list of available environment variables](#)

```
cd C:\java\src
javac Example2.java
java Example2
```

Advanced ▾

Add build step ▾

Post-build Actions


Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.


Save Apply

Dashboard > java_prog > #3


Status

- </> Changes
- Console Output
- Edit Build Information
- Delete build '#3'
- Timings


 **#3 (Apr 16, 2025, 3:04:25 PM)**

 Add description


Keep this build forever

 Started by user [Vinay Dawani](#)

Started 1 min 58 sec ago
Took 0.99 sec

 This run spent:


- 5 ms waiting;
- 0.99 sec build duration;
- 1 sec total from scheduled to completion.


 No changes.


Dashboard > java_prog > #3 > Console Output

Status

- </> Changes
- Console Output**
- Edit Build Information
- Delete build '#3'
- Timings

 **Console Output**

 Download

 Copy

View as plain text

Started by user [Vinay Dawani](#)

Running as SYSTEM

Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\java_prog

[java_prog] \$ cmd /c call C:\WINDOWS\TEMP\jenkins17758450227728903501.bat

C:\ProgramData\Jenkins\jenkins\workspace\java_prog>cd C:\java\src

C:\java\src>javac Example2.java

C:\java\src>java Example2

T11 jenkins java Application

C:\java\src>exit 0

Finished: SUCCESS

REST API Jenkins 2.492.2

4. Running Python script

Dashboard > Py_prog > Configuration

Configure

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

Execute Windows batch command ?

Command

See the list of available environment variables

cd C:\py
"C:\Users\vinay\AppData\Local\Programs\Python\Python313\python.exe" Example3.py

Advanced ▾

Add build step ▾

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

Add post-build action ▾

Save

Apply

Dashboard > Py_prog > #3

Status

</> Changes

Console Output

Edit Build Information

Delete build '#3'

Timings

✓ #3 (Apr 16, 2025, 3:16:17 PM)

Add description

Keep this build forever

Started by user Vinay Dawani

Started 4.6 sec ago
Took 0.22 sec

This run spent:

- 3 ms waiting;
- 0.22 sec build duration;
- 0.22 sec total from scheduled to completion.

</> No changes.

REST API

Jenkins 2.492.2

Dashboard > Py_prog > #3 > Console Output

Status

</> Changes

Console Output

Edit Build Information

Delete build '#3'

Timings

✓ Console Output

Download

Copy

View as plain text

Started by user Vinay Dawani

Running as SYSTEM

Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Py_prog

[Py_prog] \$ cmd /c call C:\WINDOWS\TEMP\jenkins12018421368350287335.bat

C:\ProgramData\Jenkins\jenkins\workspace\Py_prog>cd C:\py

C:\py>"C:\Users\vinay\AppData\Local\Programs\Python\Python313\python.exe" Example3.py

Python in Jenkins

C:\py>exit 0

Finished: SUCCESS

REST API

Jenkins 2.492.2

Conclusion:

Thus, we have successfully studied continuous integration and installed, configured, and understood programming with Jenkins.