Name: Vinay Dawani

Roll no: 22

Batch: T11

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.

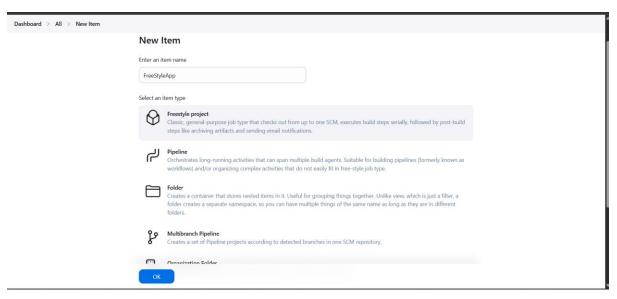
4. Configuring Tomcat for Automatic Deployment

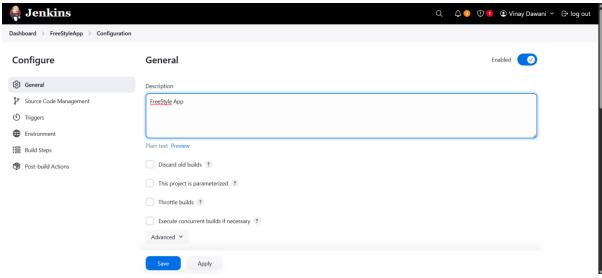
For **automated deployment** to **Tomcat**, ensure that:

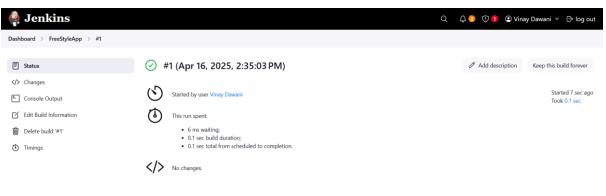
- Tomcat Manager is enabled and configured to accept deployment commands (for example, via HTTP requests).
- You provide the correct Tomcat Manager credentials (username and password) to Jenkins for deployment.

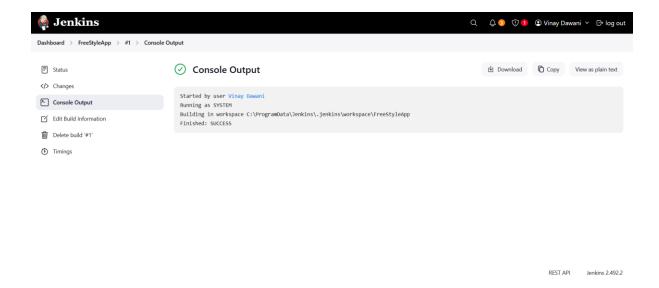
Implementation:

1. Freestyle App

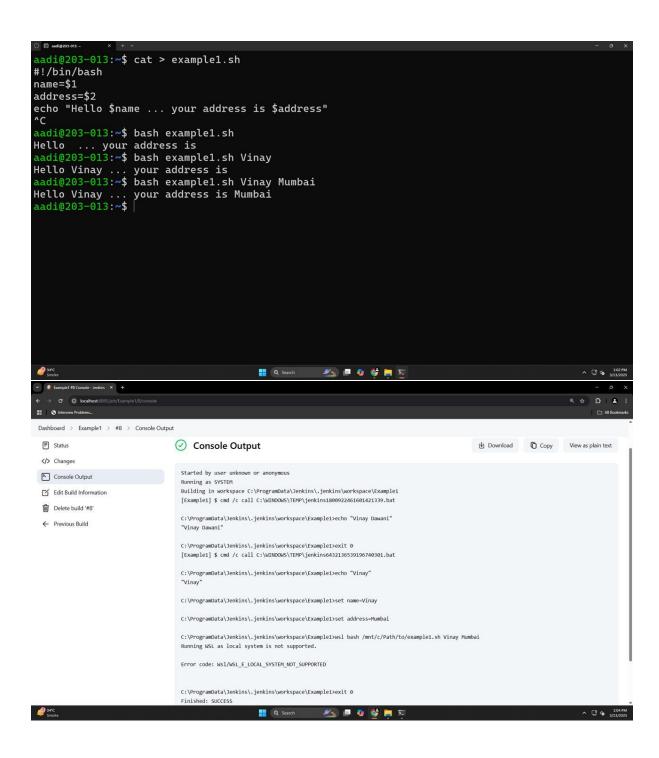




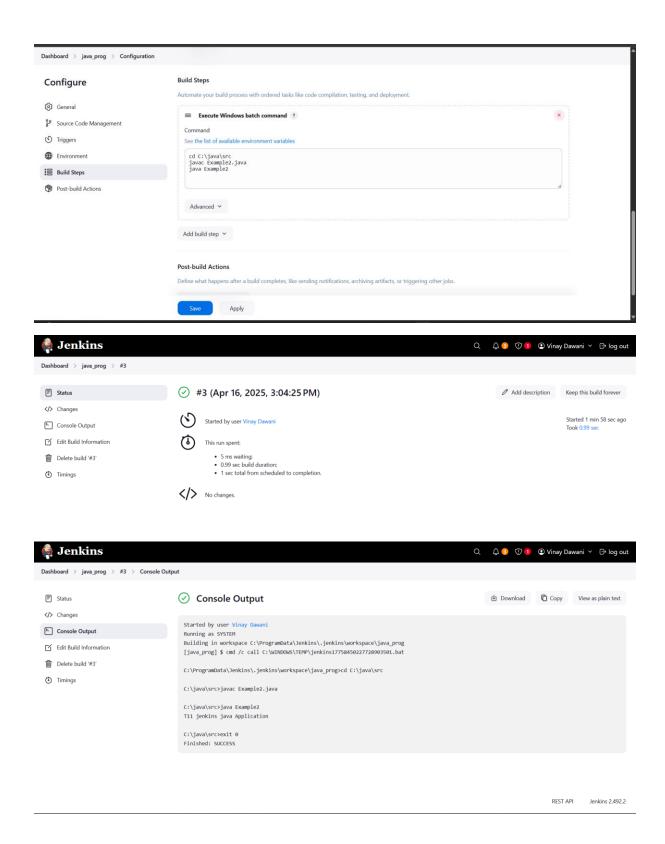




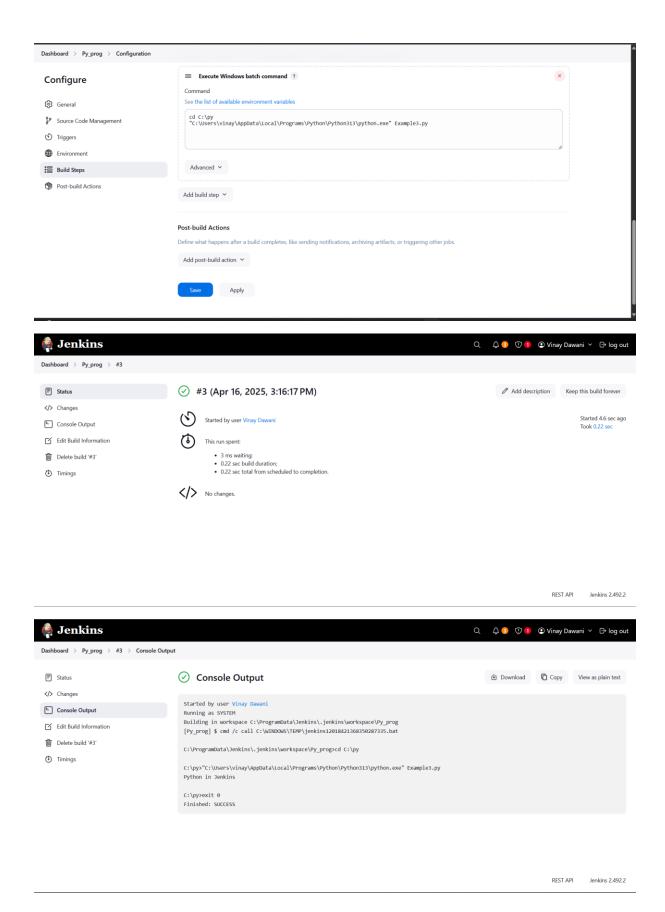
2. Taking Parameters



3. Running java program



4. Running Python script



Conclusion:

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