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:

1. Introduction to CI/CD Pipelines

A CI/CD pipeline is a series of steps that software changes go through to be automatically built, tested, and deployed. The goal is to automate the software delivery process to reduce manual effort, minimize errors, and increase the speed and consistency of deployments.

In Jenkins, a pipeline is essentially a set of plugins and syntax that supports integrating and implementing continuous delivery pipelines. These pipelines are defined as code using a Jenkinsfile, enabling version control and better traceability.

2. Jenkins Pipelines: Concept and Importance

Unlike Freestyle projects, which are limited in flexibility, Jenkins Pipelines offer a powerful and extensible way to define the complete lifecycle of an application from build to production. Pipelines are especially useful when complex build and deployment processes need to be automated.

Types of Jenkins Pipelines:

- Declarative Pipeline: Uses a predefined structure and syntax; easier to understand and maintain.
- Scripted Pipeline: More flexible, written in Groovy, and suitable for advanced users.

Advantages of Jenkins Pipelines:

- Pipeline-as-code for versioning and rollback
- Better error handling and retry logic
- Parallel execution of stages
- Visualization of stages and logs

3. Build Tools: Maven, Gradle, and Ant Integration in Pipelines

Each build tool has a unique approach and syntax, but all can be easily integrated into Jenkins pipelines.

- Maven: Uses the pom.xml file to manage the lifecycle, dependencies, and plugins. Command: mvn clean install

- Gradle: Supports Groovy/Kotlin DSL for defining build scripts, with commands like gradle build or ./gradlew build

- Ant: XML-based, customizable with build targets specified in build.xml. Run via ant, ant test, or ant deploy

4. Installing and Configuring Tomcat Server for Deployment
Apache Tomcat is an open-source implementation of the Java Servlet and JavaServer Pages technologies. It is widely used to deploy and run Java web applications.

Steps for Setup:
- Download and extract the Tomcat server
- Set environment variables like CATALINA_HOME
- Place your .war file inside the /webapps directory
- Start the server using startup.sh or startup.bat
- Allowing Jenkins to Deploy on Tomcat:
- Create a user in tomcat-users.xml with roles: manager-script, admin-gui
- Jenkins will deploy via the Tomcat Manager API using credentials

5. Creating the Jenkins Pipeline Script
The goal here is to automate the build, test, and deployment process using a Jenkinsfile. This file contains all the stages needed to process the code and push it to a live Tomcat instance.

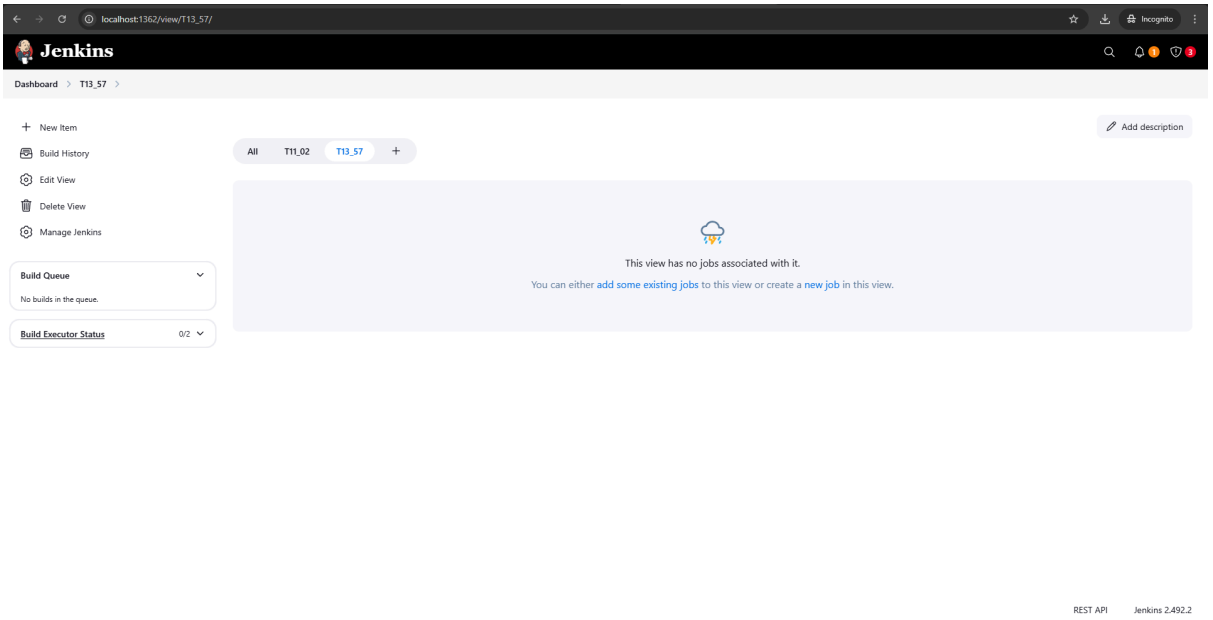
Typical Pipeline Stages:
- Checkout Source Code – Pull the latest code from Git
- Build Stage – Use Maven/Gradle/Ant to compile and package the application
- Testing Stage – Run unit tests and log results

Deployment Stage – Send .war file to Tomcat via HTTP POST or SCP

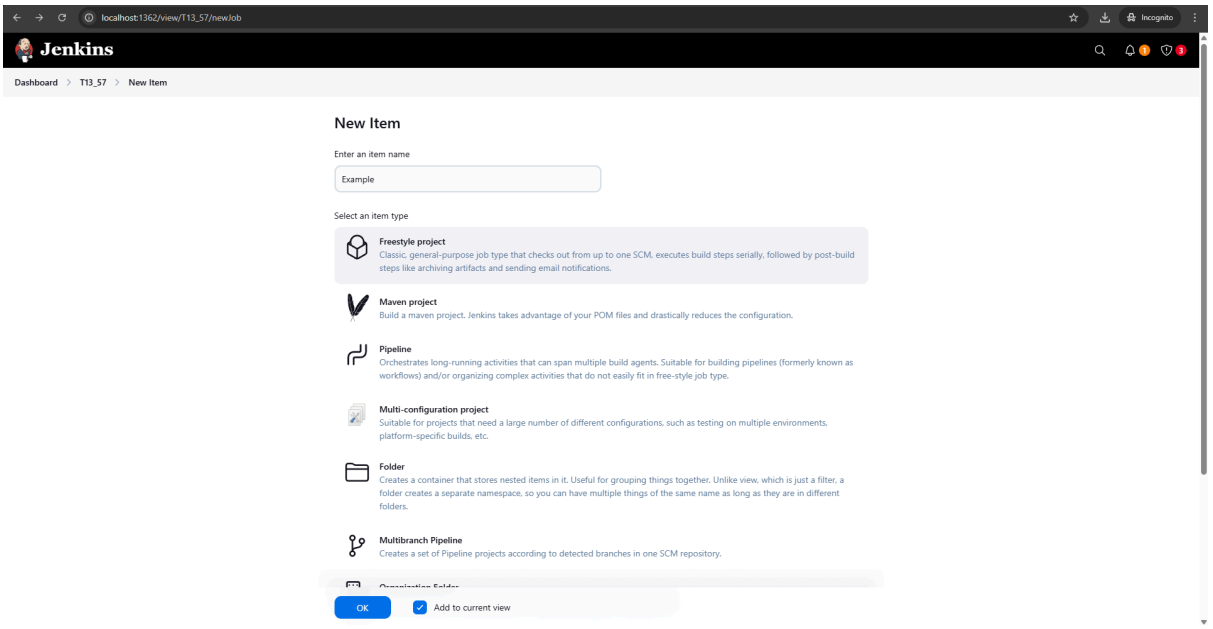
Implementation:

Example 1: Deploying a Freestyle App in Jenkins

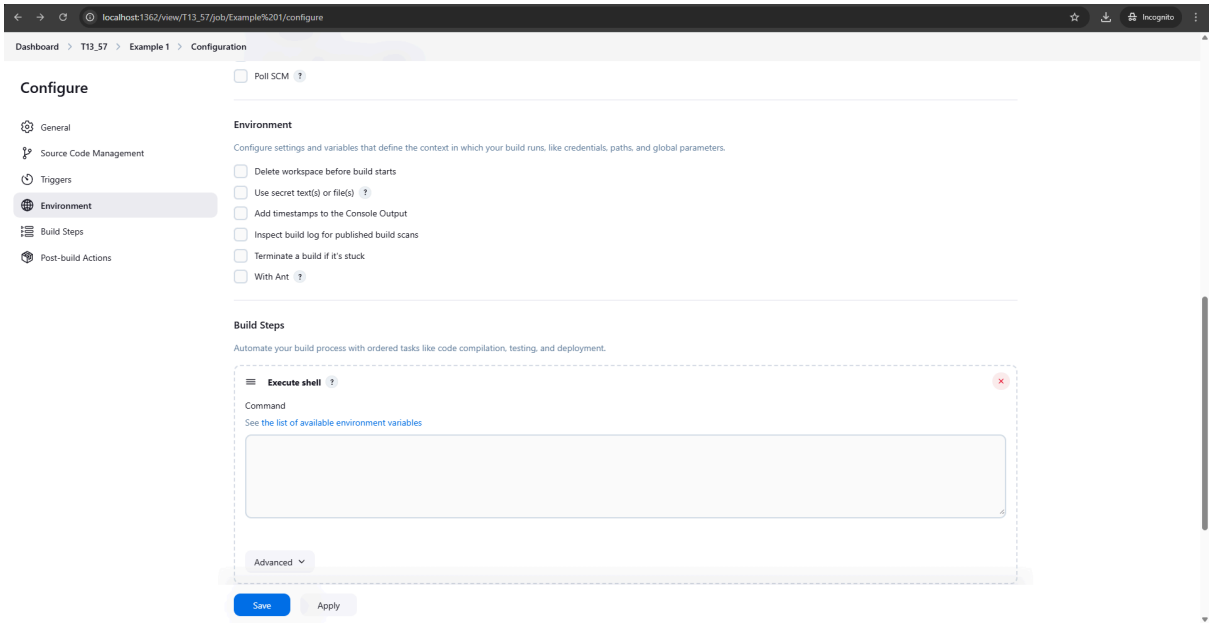
Step 1-: Click on Create new jobs.



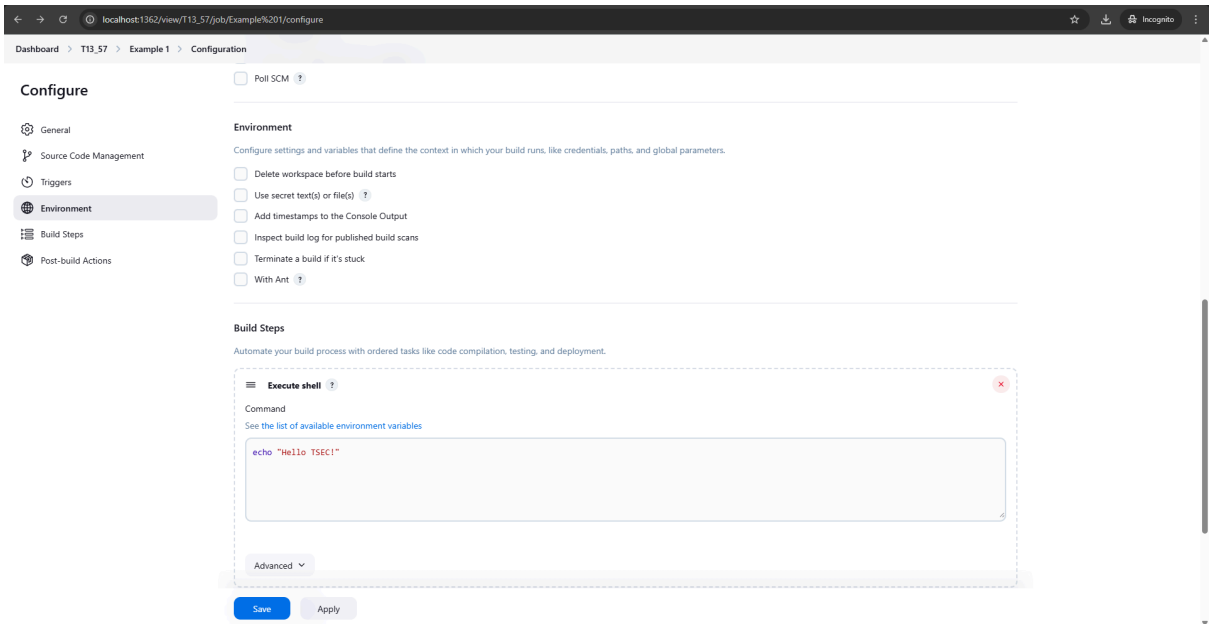
Step 2-: Now Specify name to the project as “Example1”, select Option “Free style project “ and click on OK button



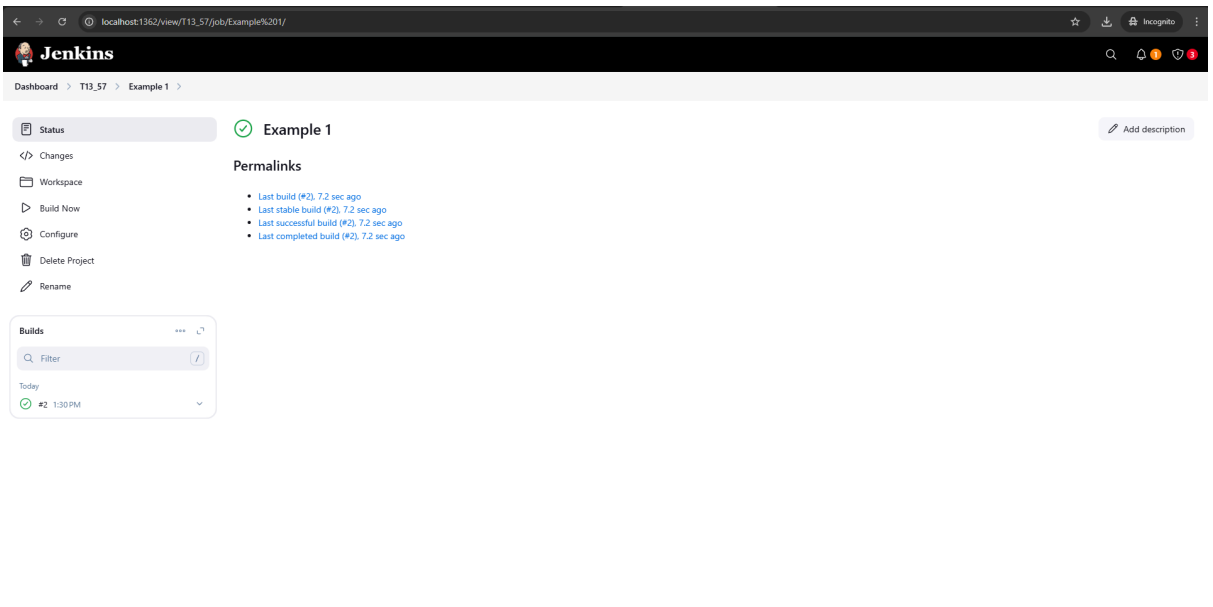
Step 3-: In this project we are going to learn how to run simple shell script on Jenkins. So, Click on Build option select Execute script from dropdown menu

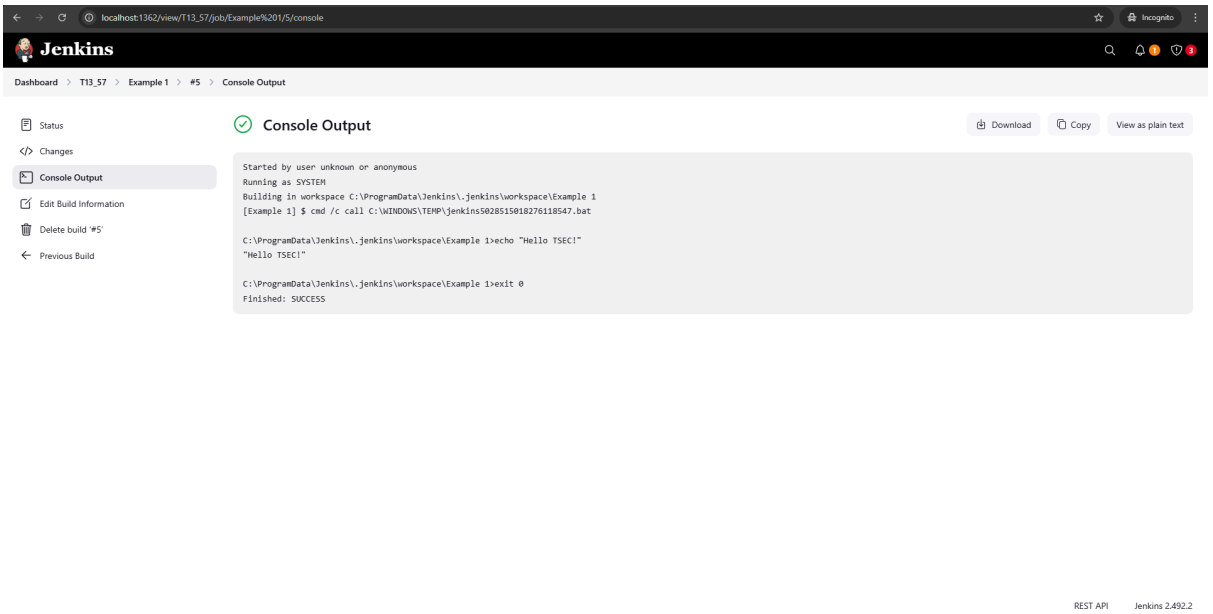


Step 4-: Now Write a Simple Shell command to print the text as Like given below.



Step 5-: No Build a project to see the output Click on our first build “1” followed by console output to see the output

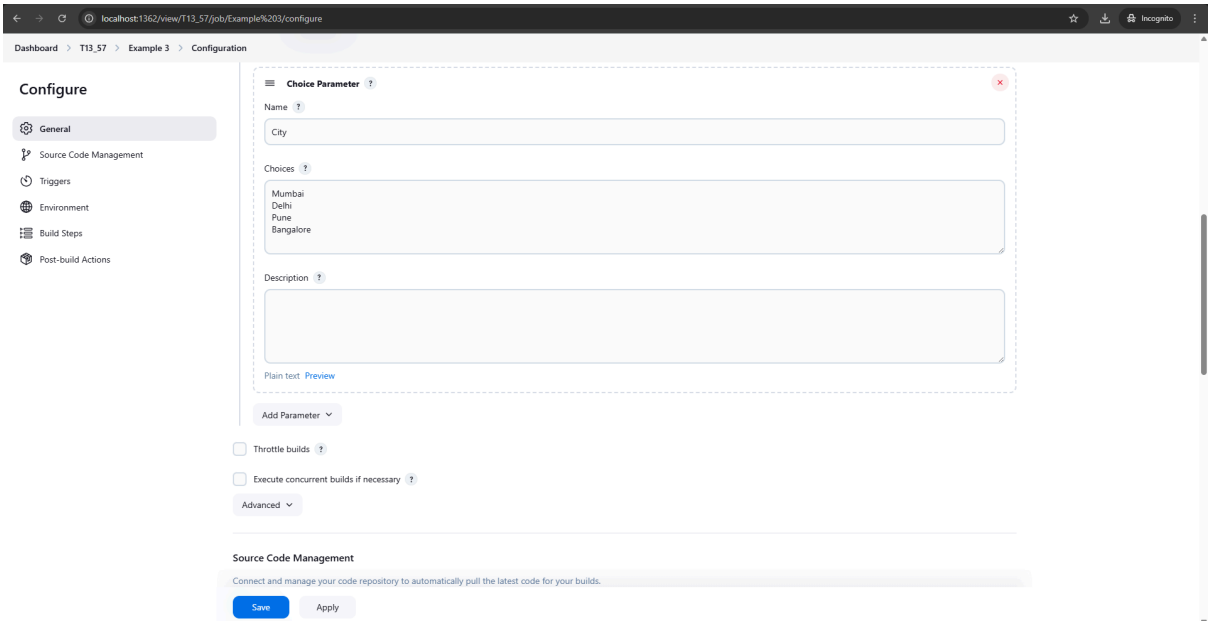




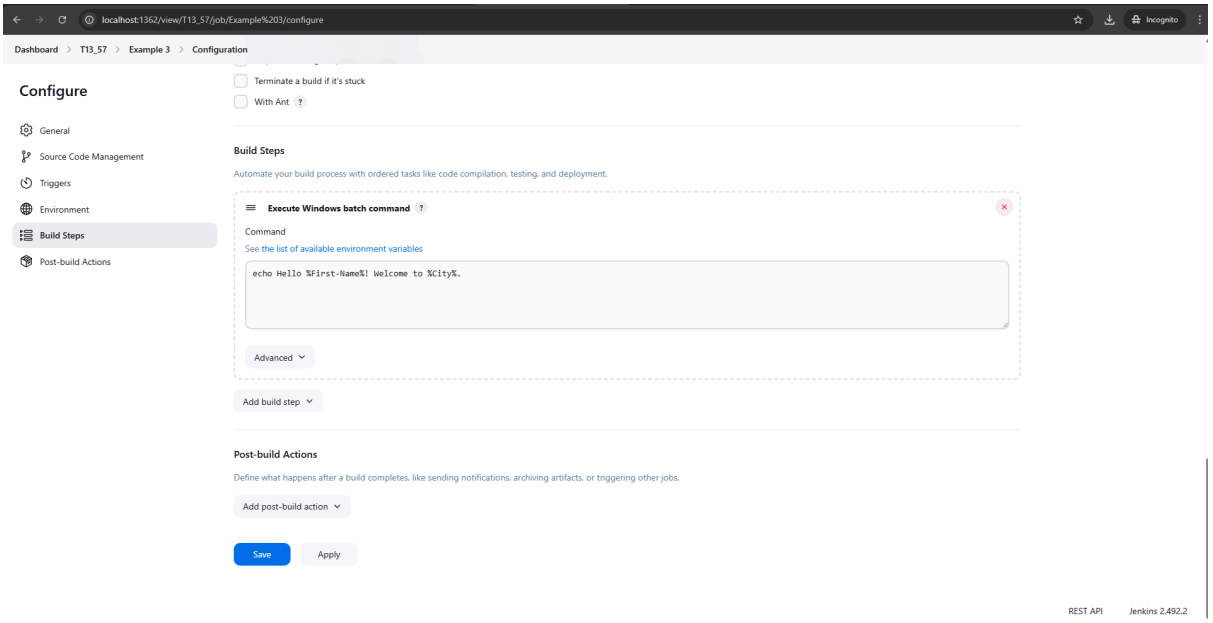
Example 2: Parameterize Build

In this program we are going to see how to provide parameters during runtime to your shell script or java program.

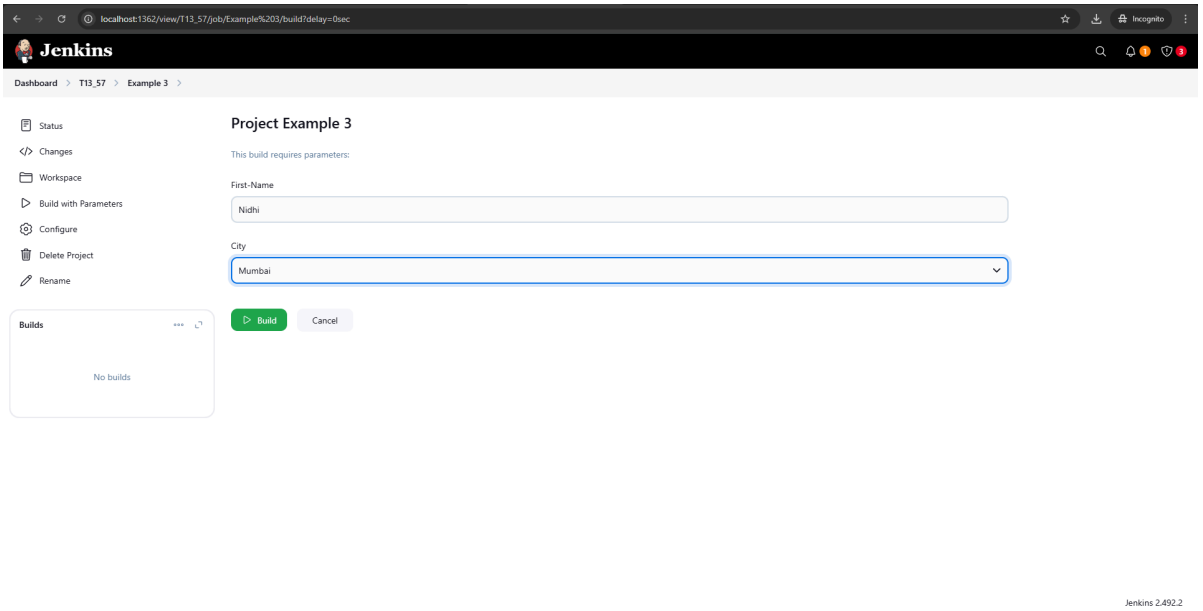
Step 1-: Create a free style project example3 by clicking on new item followed by specifying project

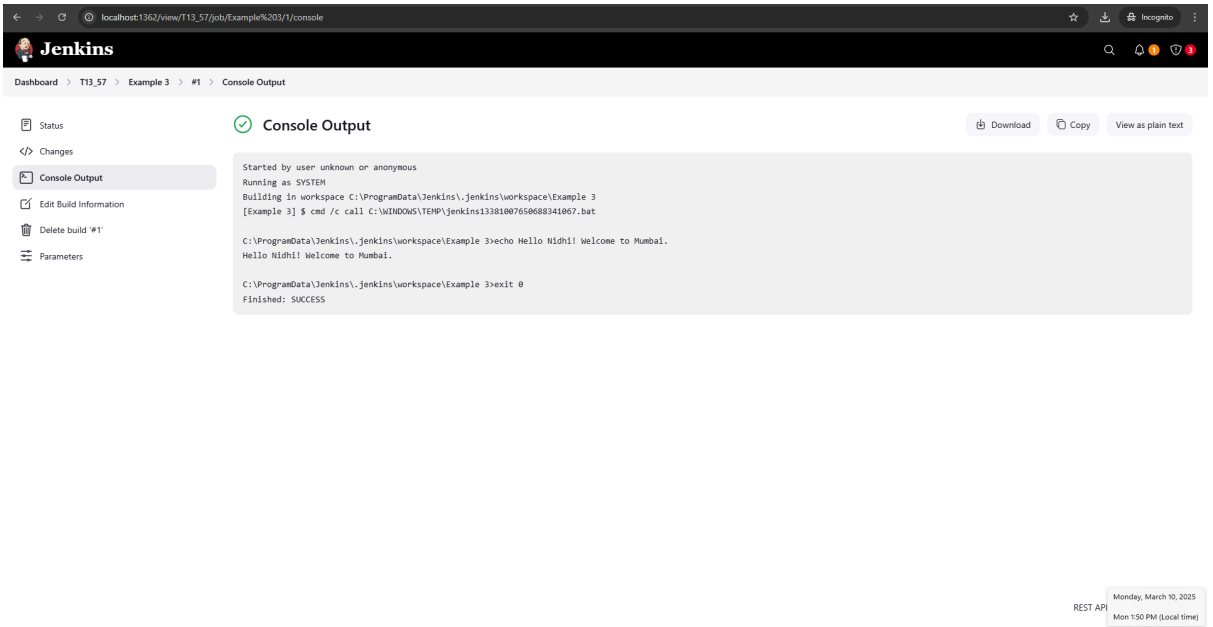


Step 2-: Now under general menu, select option this project is parameterize



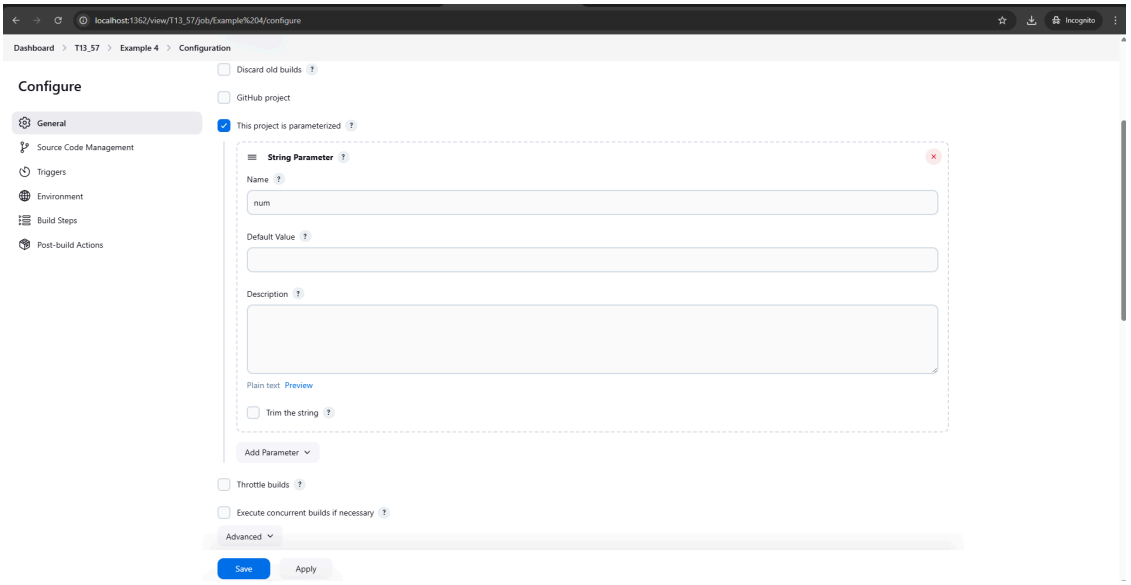
Select String parameter and specify name as “First-Name”

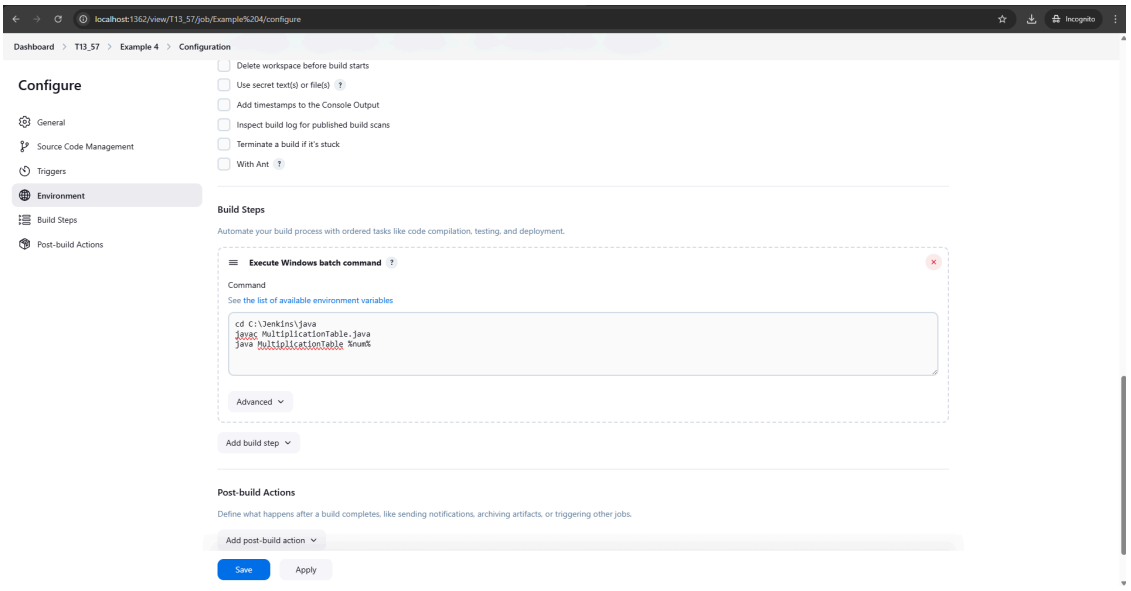


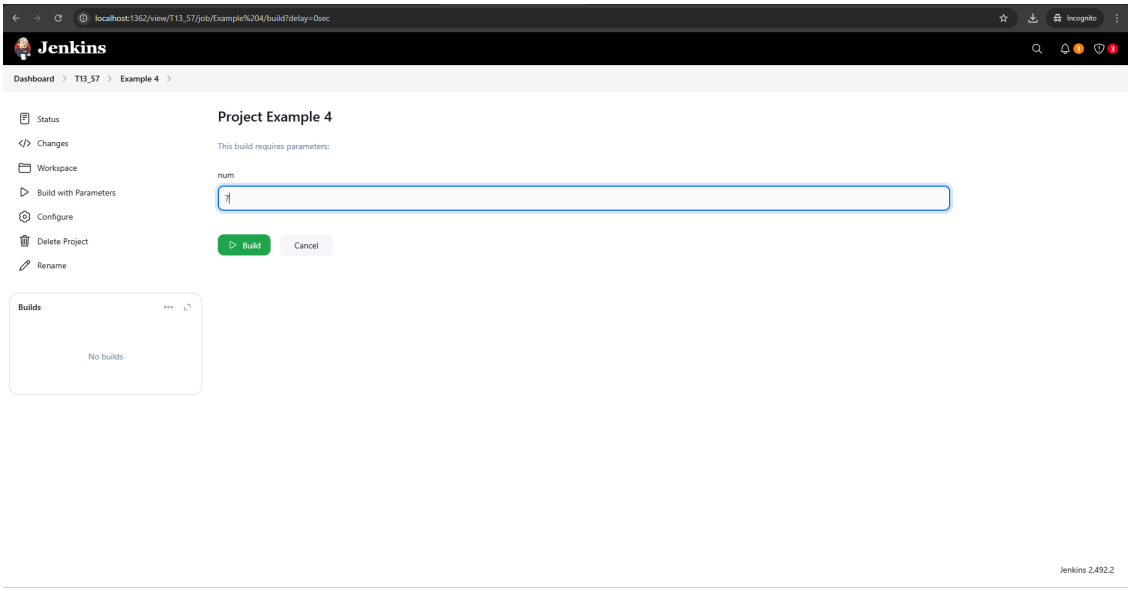


Example 4: Java Program with Parameters:

1. Create a Java file locally:
   Create a file named MultiplicationTable.java with this content:
   Create a new Freestyle project:
2. Name it "Example4"
   Check "This project is parameterized"
   Add a String Parameter named "num"
3. Add build step:
   Select "Execute Windows batch command"
4. Save and build with parameters

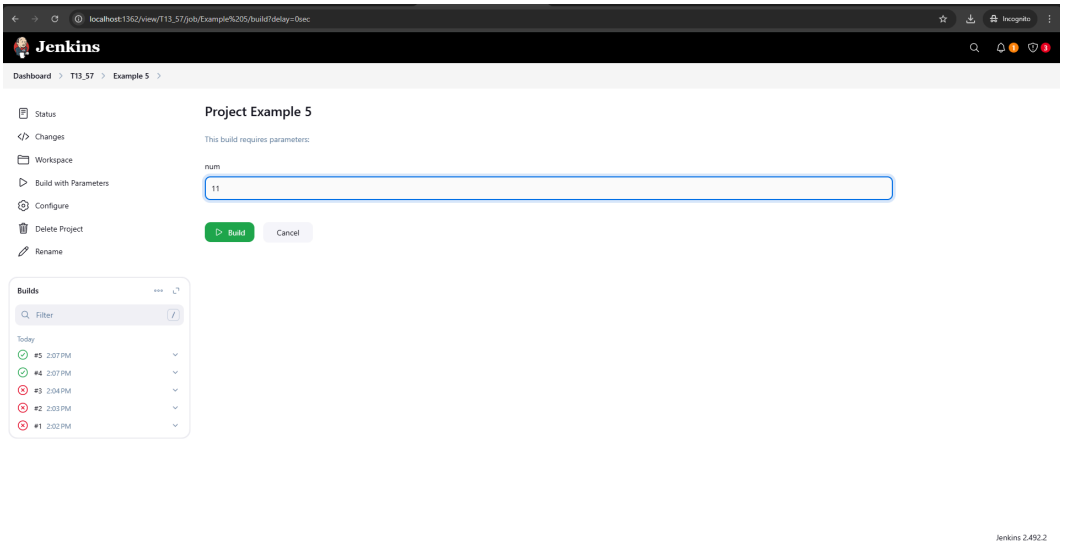

localhost:1362/view/T13_57/job/Example%204/configure
Dashboard > T13_57 > Example 4 > Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Discard old builds
GitHub project
This project is parameterized
String Parameter
Name
num
Default Value
Description
Plain text Preview
Trim the string
Add Parameter
Throttle builds
Execute concurrent builds if necessary
Advanced
Save
Apply


localhost:1362/view/T13_57/job/Example%204/configure
Dashboard > T13_57 > Example 4 > Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
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:\Jenkins\java
javac MultiplicationTable.java
java MultiplicationTable %num%
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

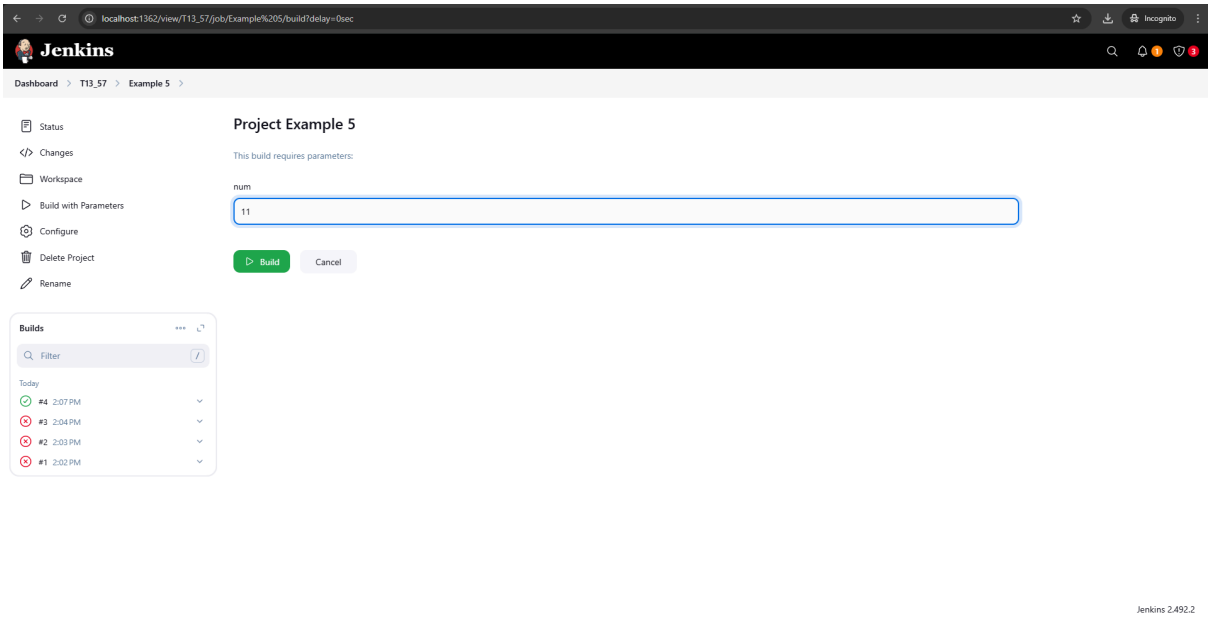

localhost:1362/view/T13_57/job/Example%204/build?delay=0sec
Jenkins
Dashboard > T13_57 > Example 4 >
Status
Changes
Workspace
Build with Parameters
Configure
Delete Project
Rename
Builds
No builds
Project Example 4
This build requires parameters:
num
7
Build
Cancel
Jenkins 2.492.2

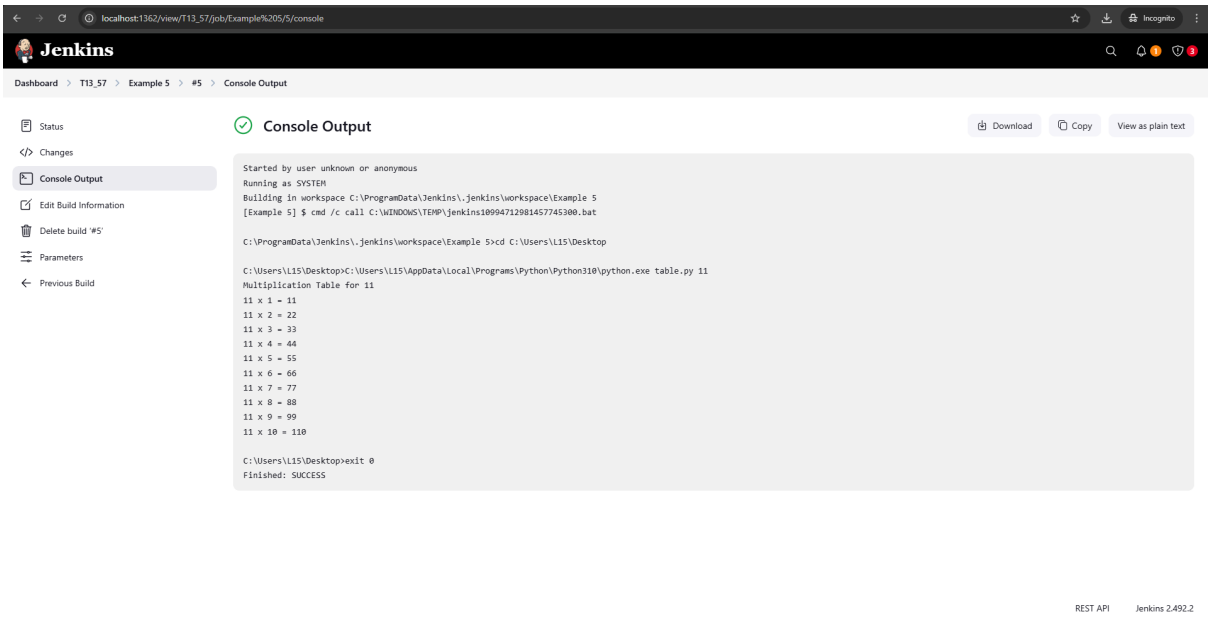
Example 5: Python Program

1. Create a Python file locally:
2. Create a new Freestyle project:
   Name it "Example5"
   Check "This project is parameterized"
   Add a String Parameter named "num"
3. Add build step:
   Select "Execute Windows batch command"
   Adjust the path to match where you saved your Python file



4. Save and build with parameters





Conclusion: Hence we have successfully Built the pipeline of jobs using Maven / Gradle / Ant in Jenkins, created a pipeline script to Test and deployed an application over the tomcat server.