# Theory:

Programming in Jenkins:

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.” In simple way, Continuous integration (CI) is the practice of frequently building and testing each change done to your code automatically.

Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Our first job will execute the shell commands. The freestyle project provides enough options and features to build the complex jobs that you will need in your projects.

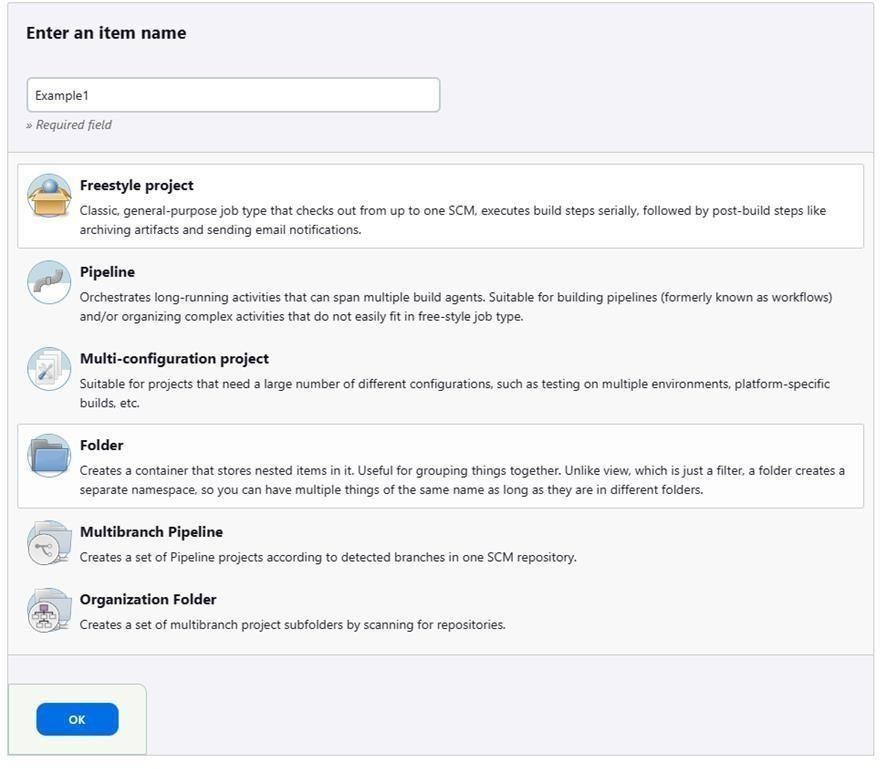
# Example 1

Example 1.1: Deploying a freestyle app in Jenkins

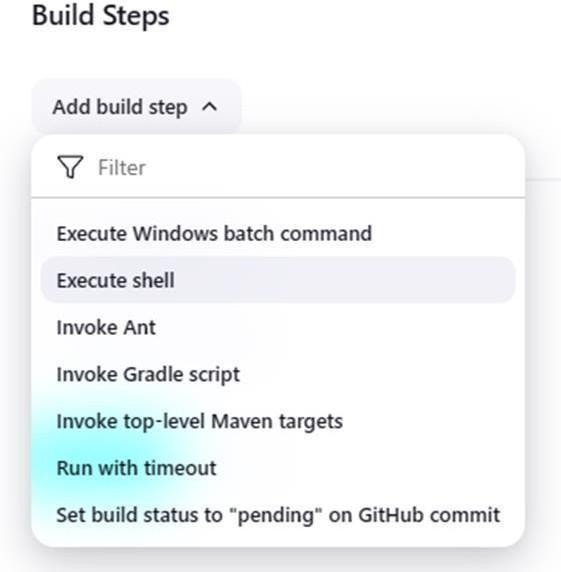
Creating a job:



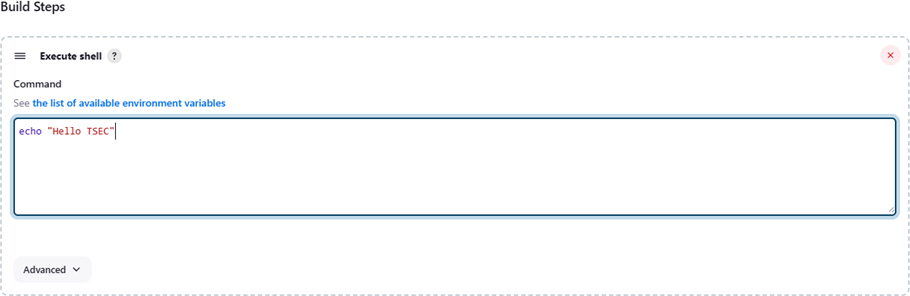
Naming the job and setting it as freestyle:



Selecting build type as “Execute shell”:



Entering a simple command for the shell execution:



Applying and saving the project configuration:

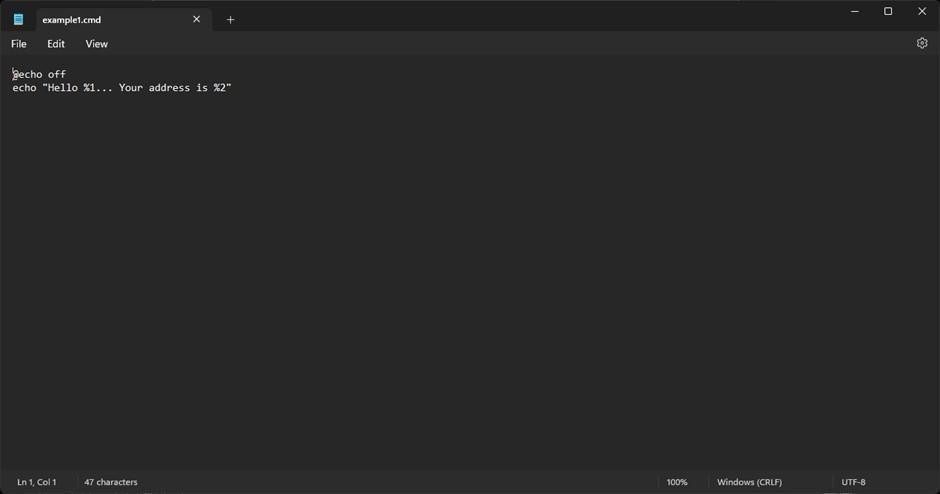


Building the project:

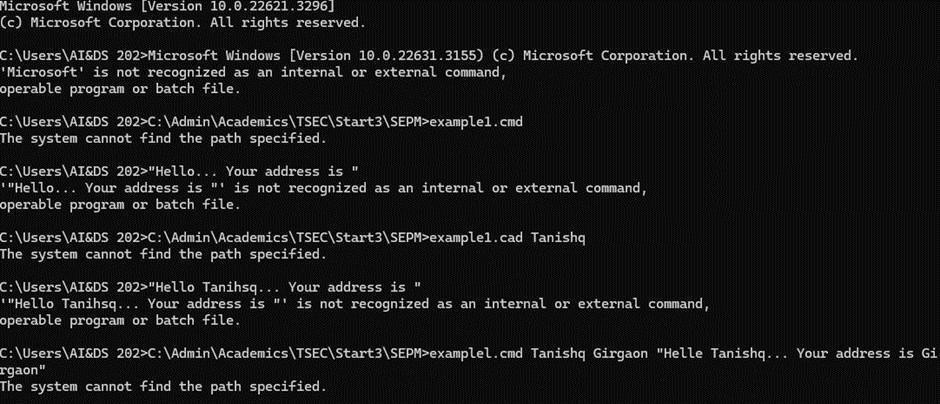


Console output (after building):

Example 1.2: Taking parameters through files Contents of script example1.cmd:



Executing script example1.cmd on the terminal:



Modifying the Jenkins project to execute the script while supplying required parameters:



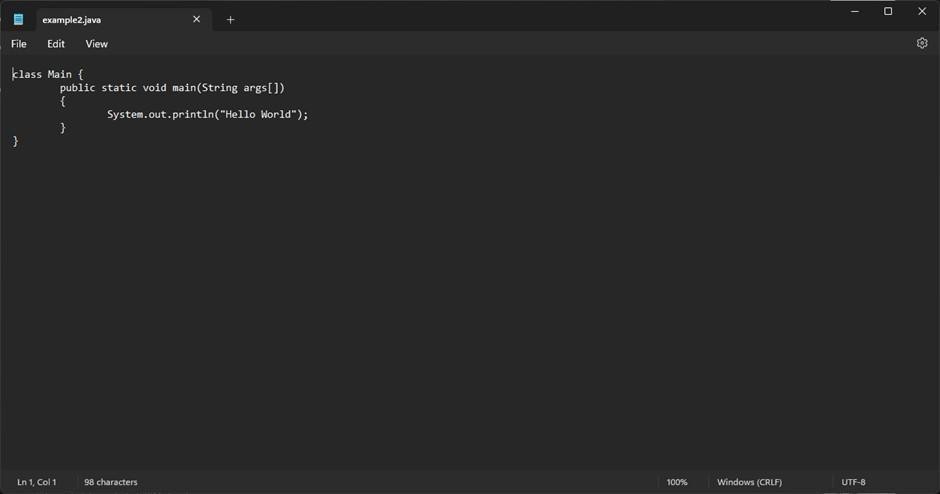
Console output after building the modified project:



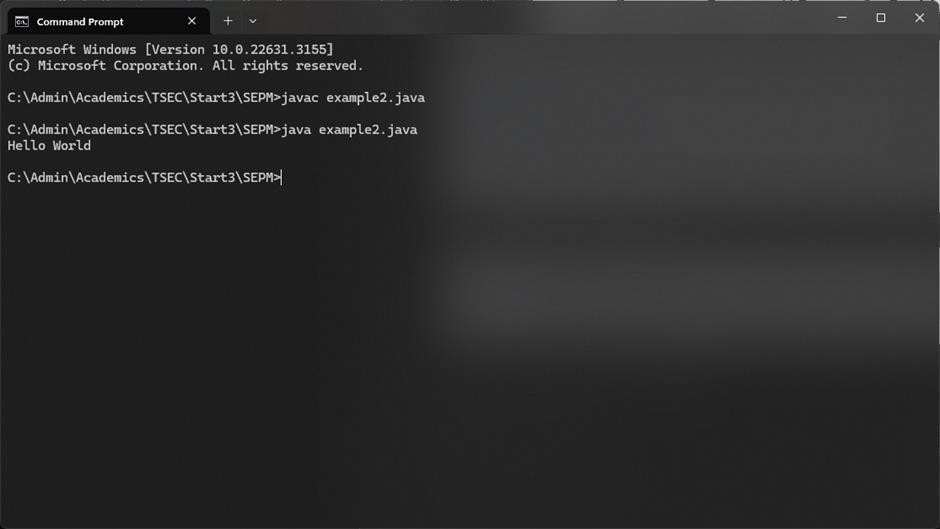
# Example 2

Example 2.1: Running a Java program under Jenkins

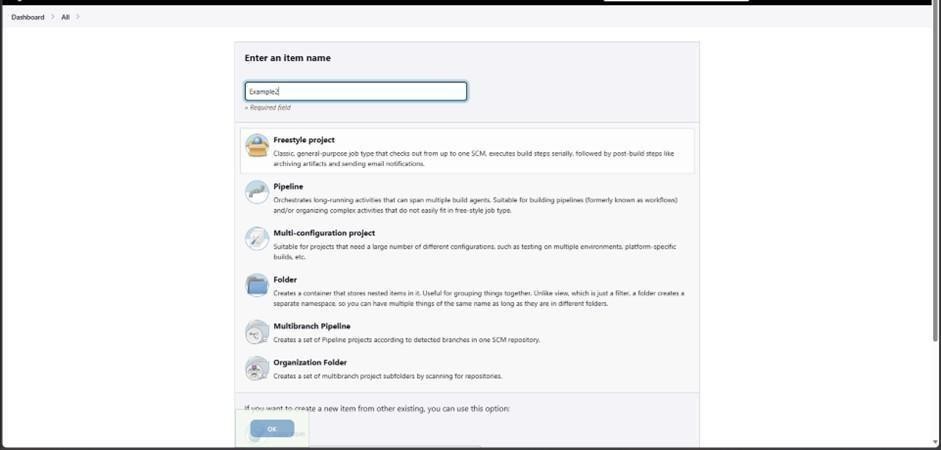
Creating a simple Java program:



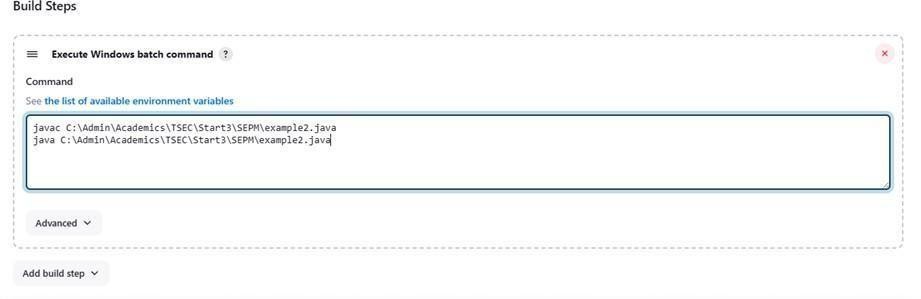
Compiling and running the program on the terminal:



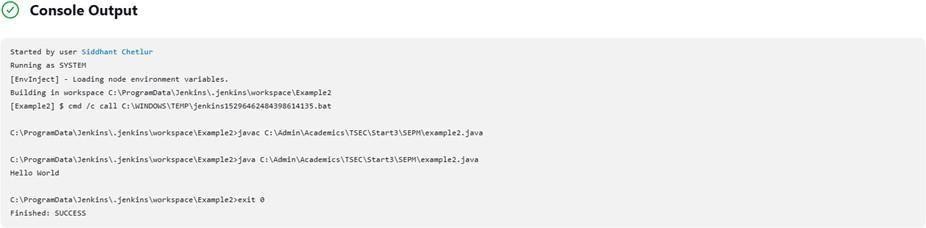
Creating a new freestyle project:



Configure new project:



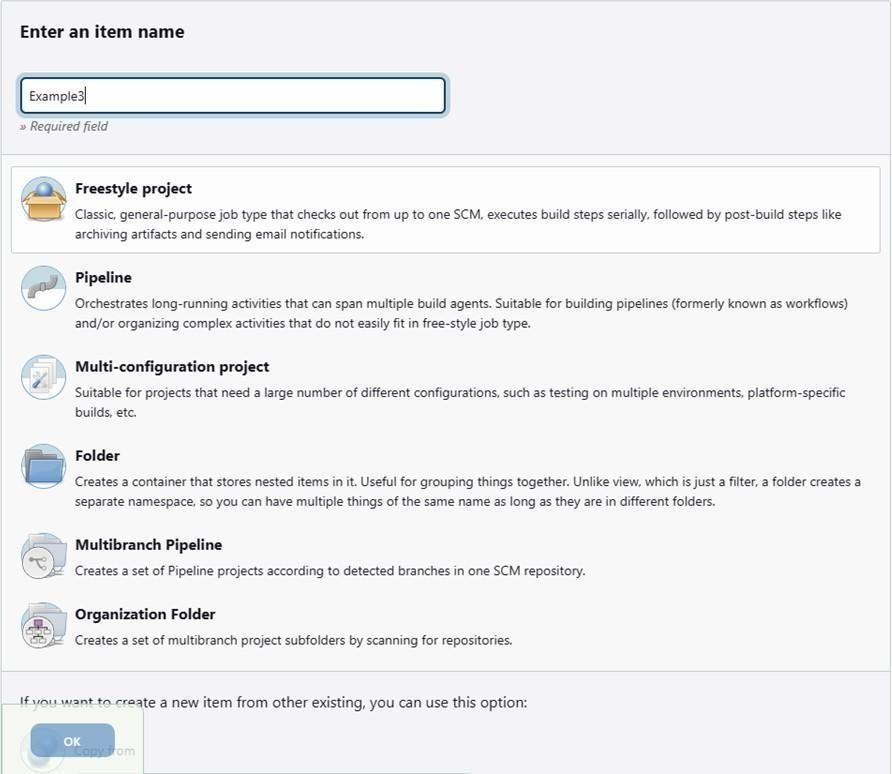
Console output after building:



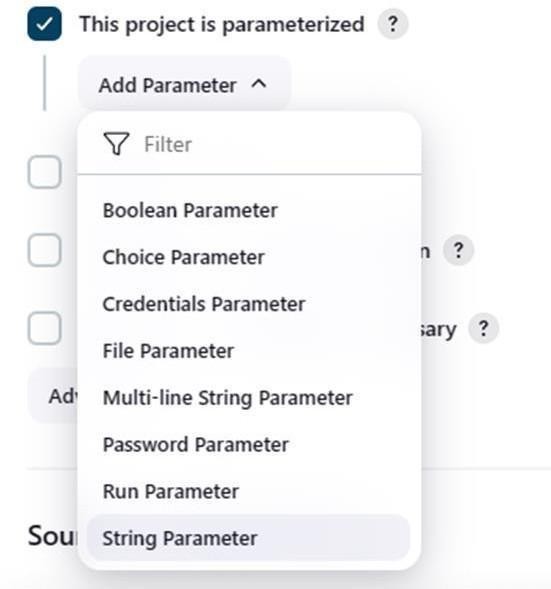
# Example 3

Example 3.1: Parameterise build

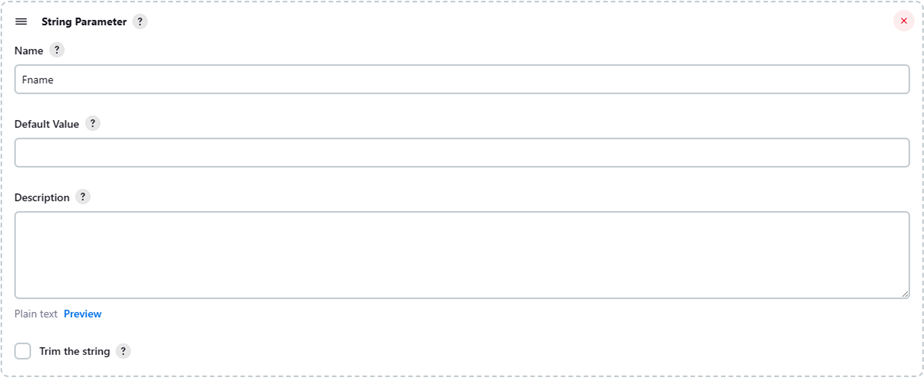
Creating a new freestyle project:



Enabling parameterisation and adding a String parameter:



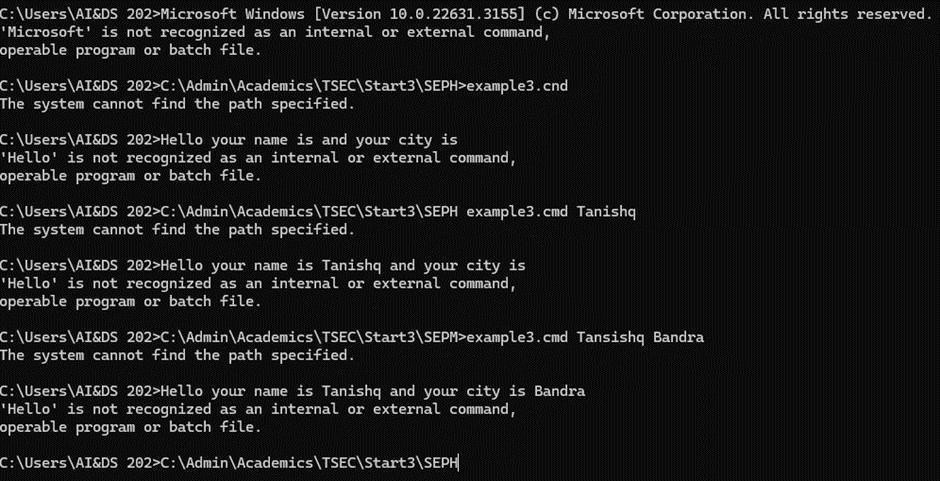
Configuring the string parameter as Fname:



Adding a choice parameter and configuring it as City with the following choices:



Creating a script which takes 2 arguments for name and city:



Configuring build steps:



Entering parameters for build:

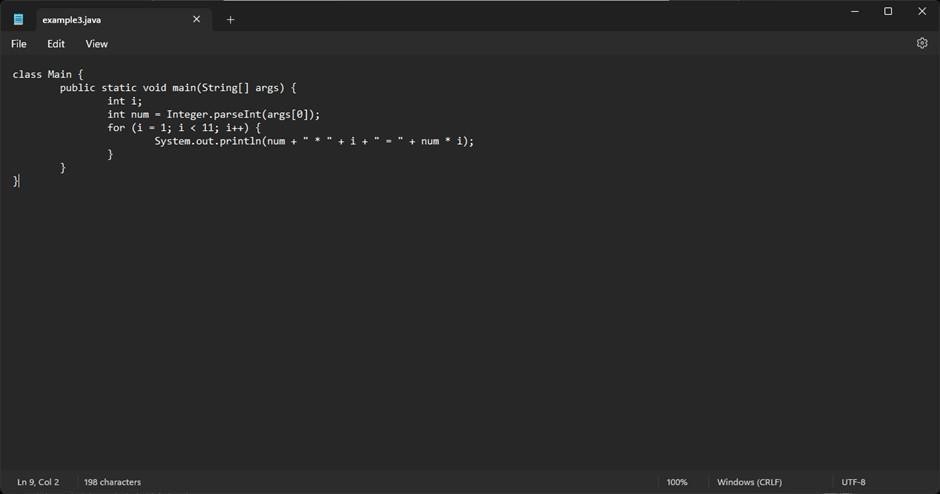


Console output after building:

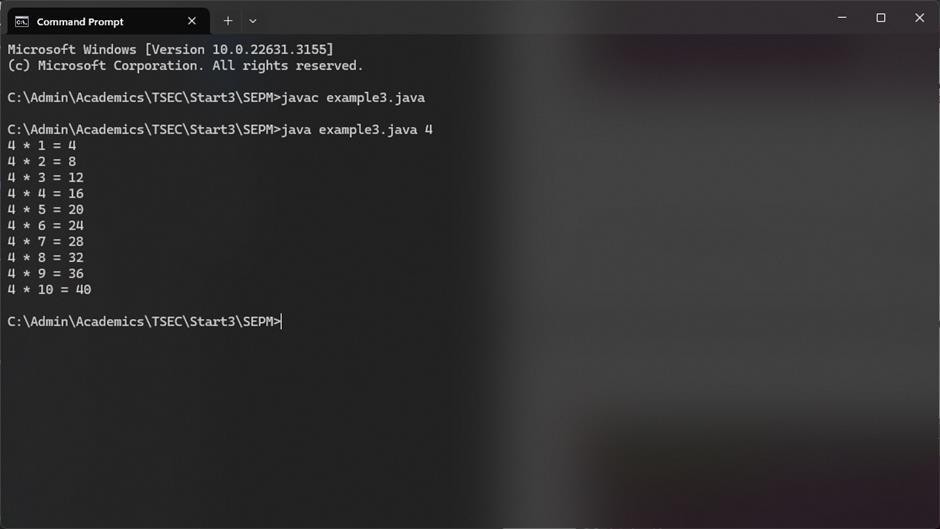


Example 3.2: Running a Java program with parameters

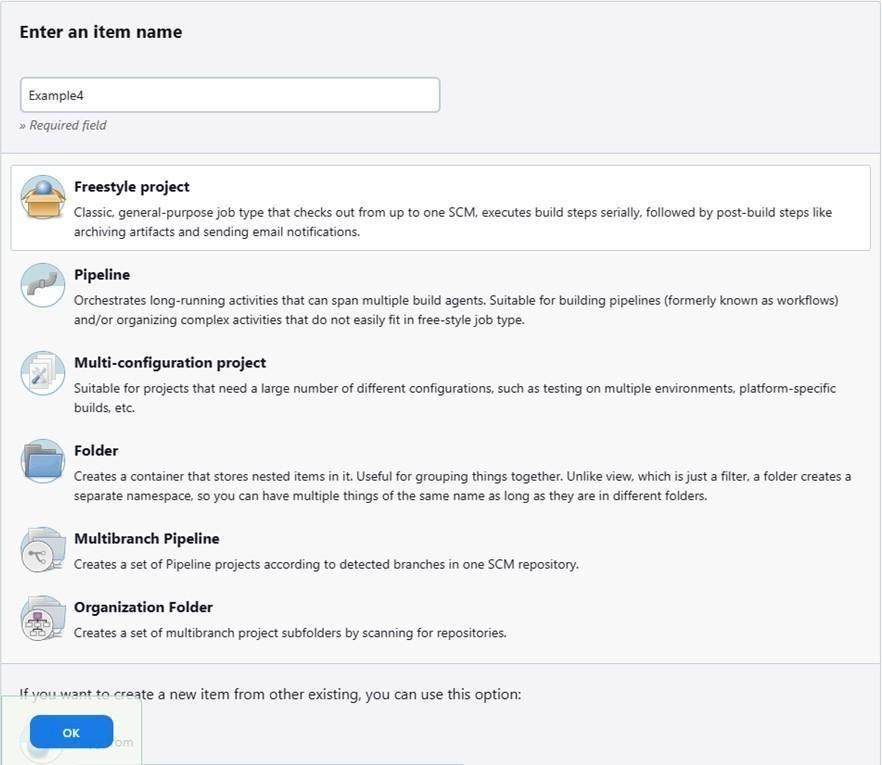
Creating a Java program with an input argument:



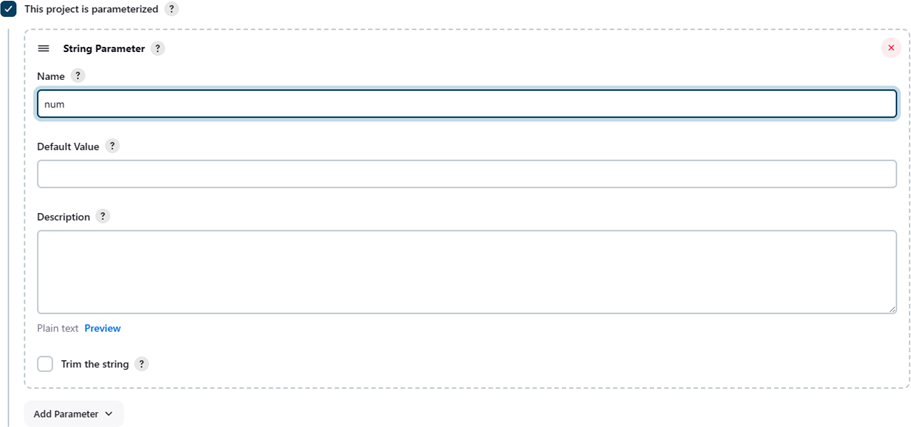
Testing the program on the terminal:



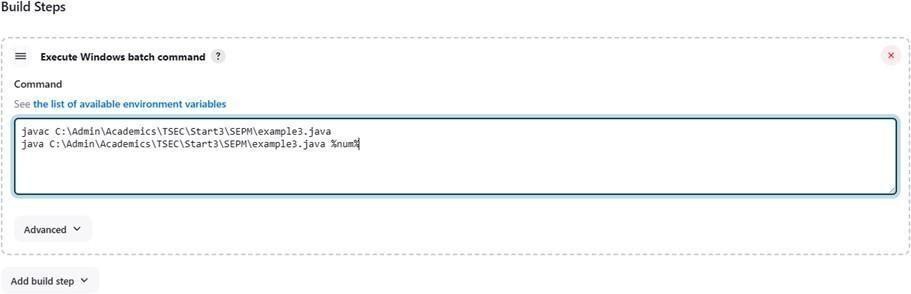
Creating a new freestyle project:



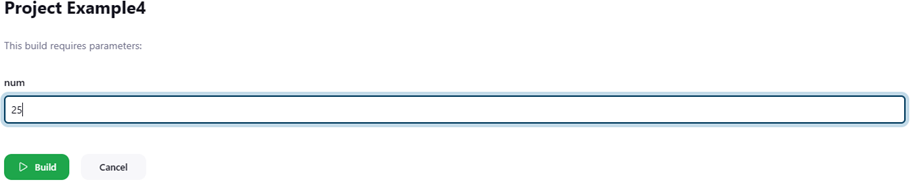
Parameterise the project by adding a string parameter as follows:



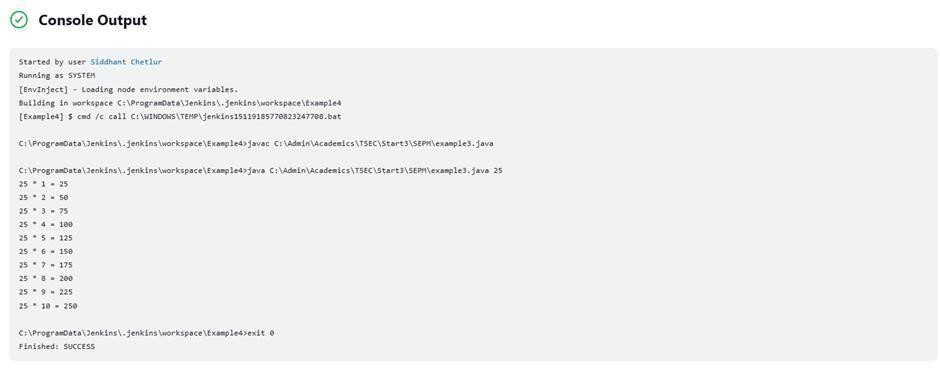
Configure the build steps:



Entering the parameter for the build:



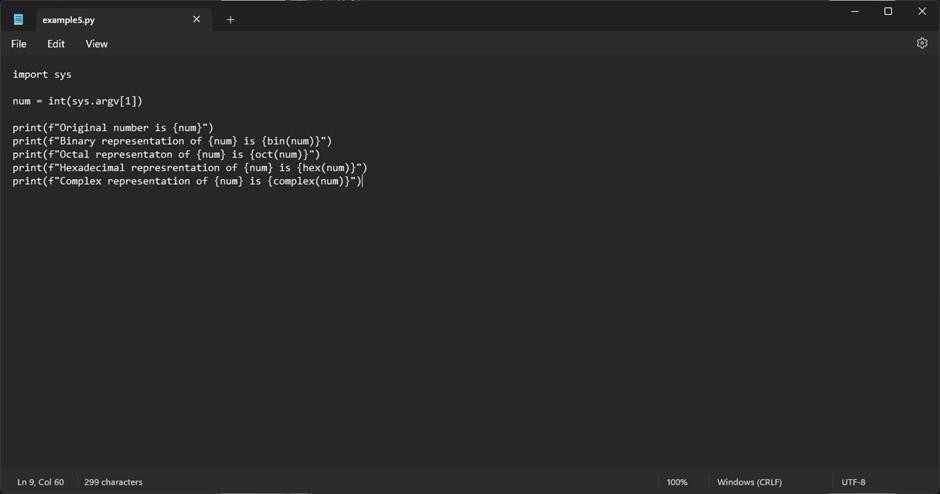
Console output after building:



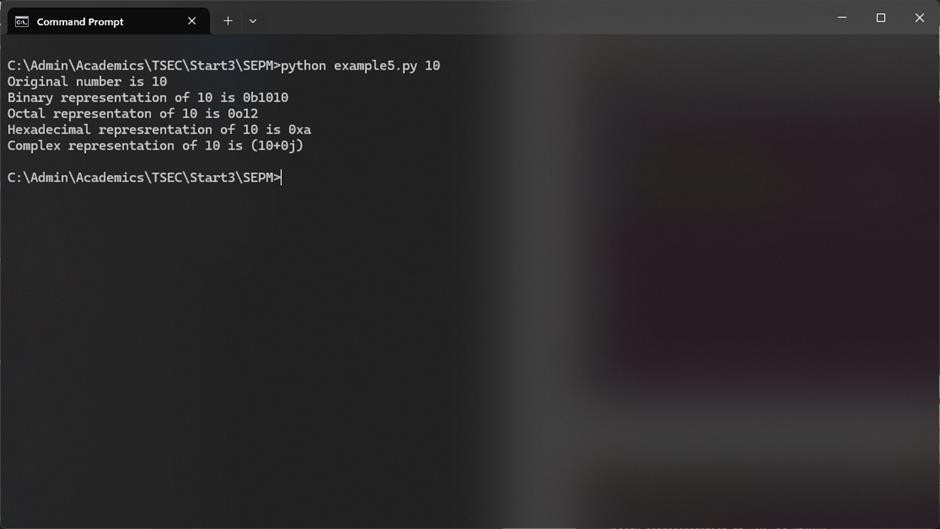
# Example 5

Example 5.1: Running a Python program

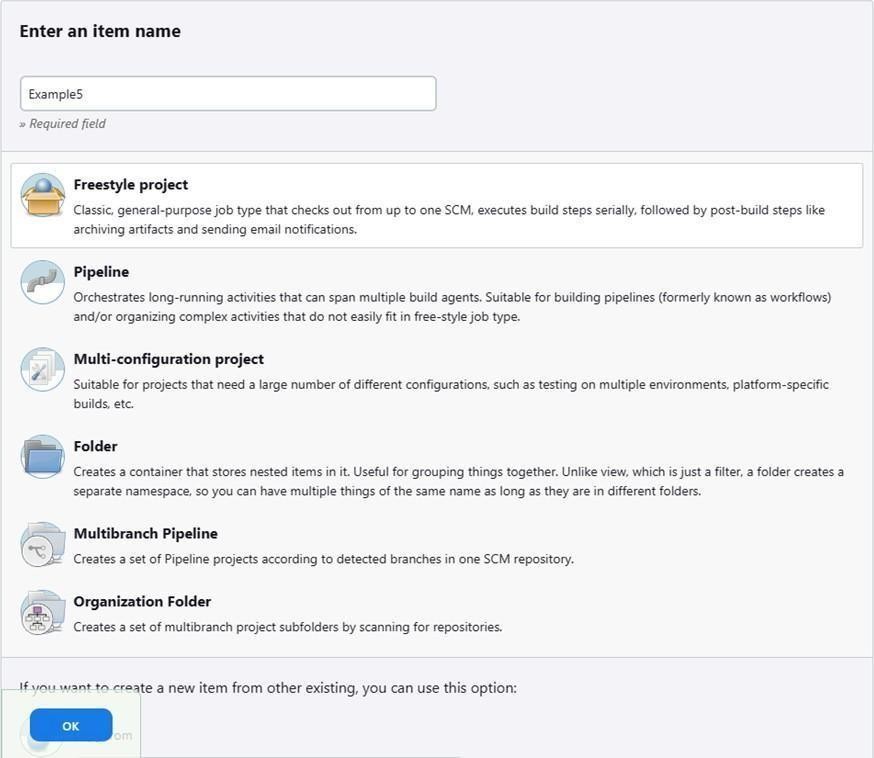
Creating a simple Python script:



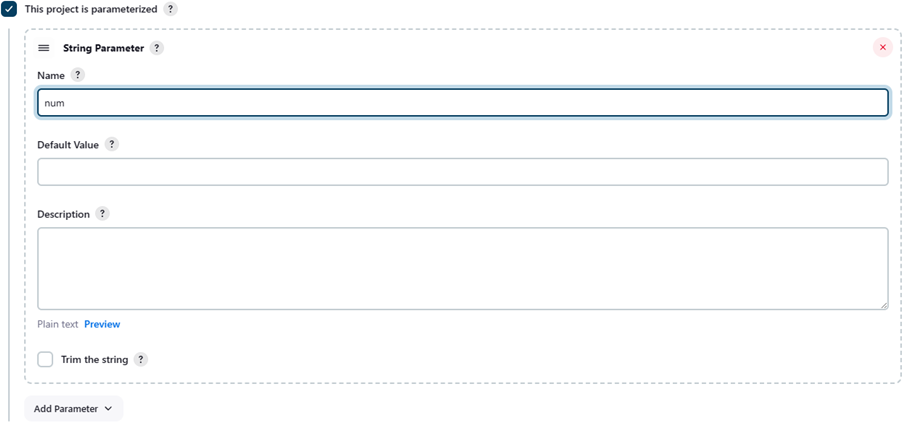
Running the Python script on the terminal:



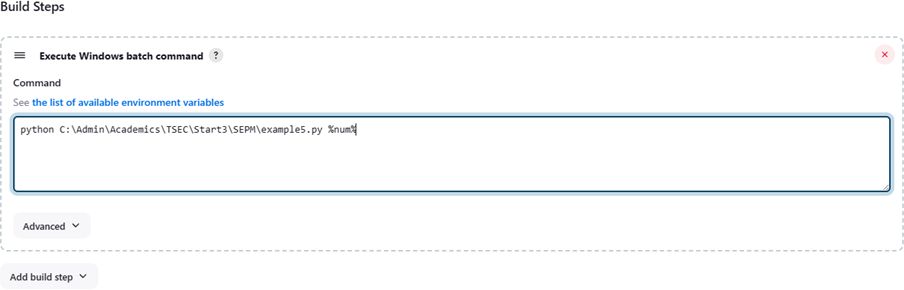
Creating a new freestyle project:



Parameterising the project with a string parameter as follows:



Configuring the build steps:



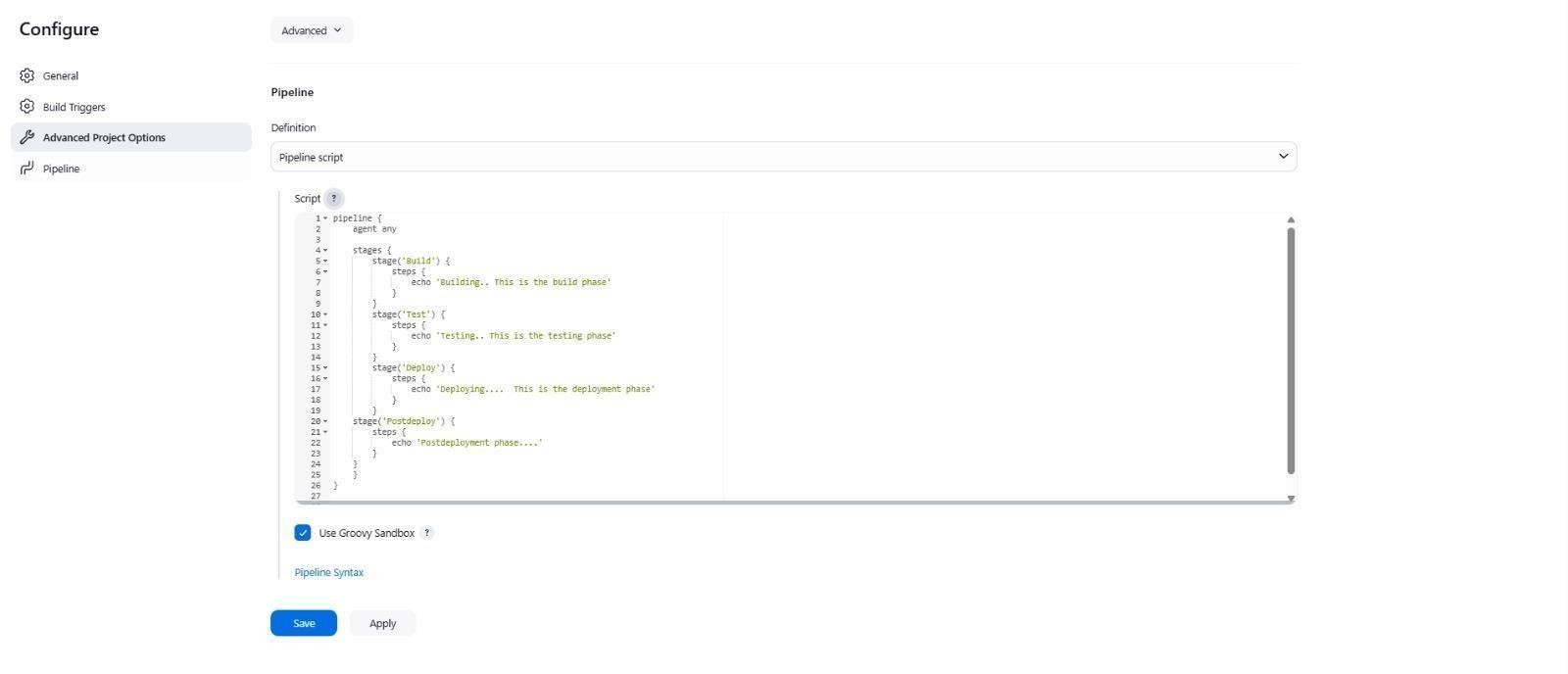
Setting the parameter for the build:

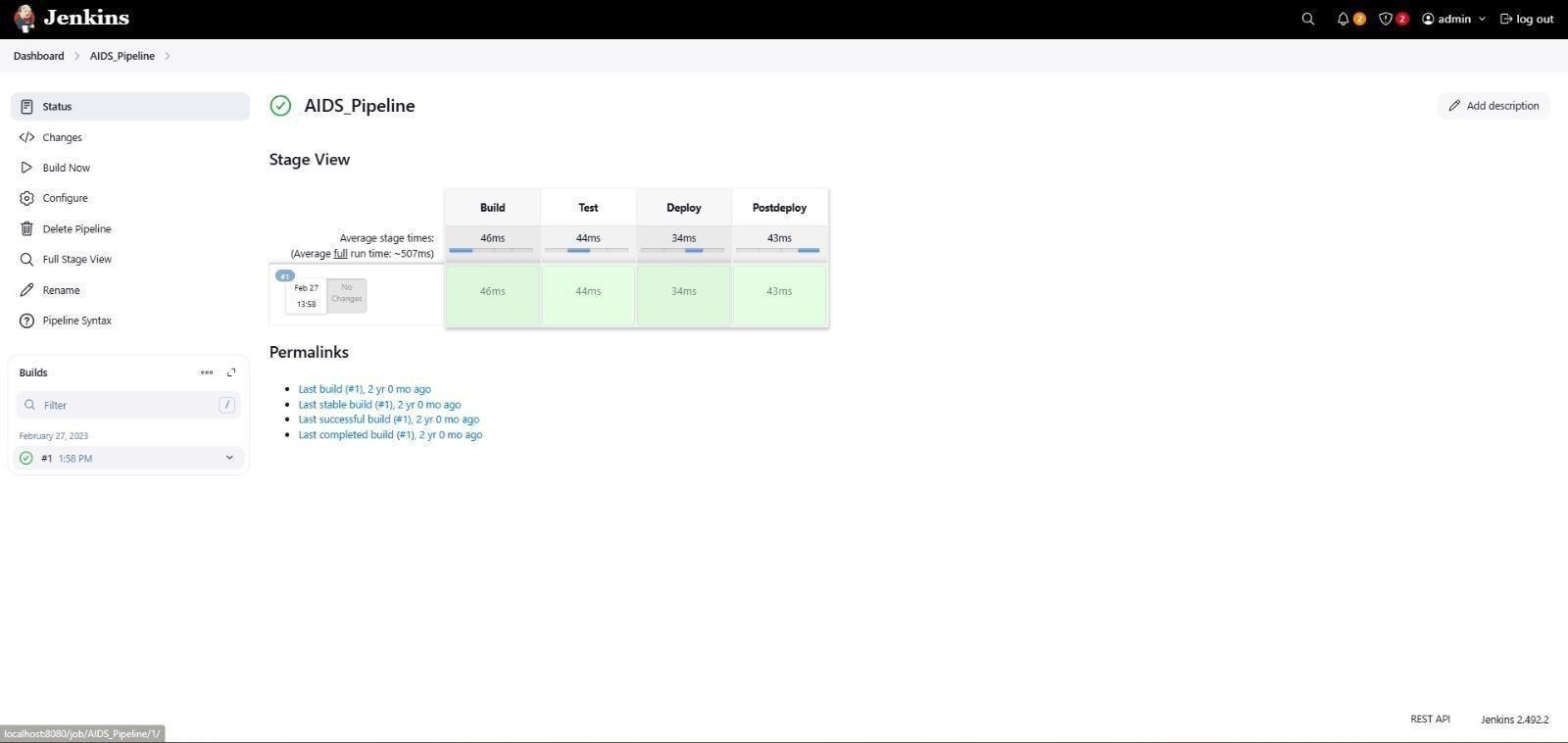


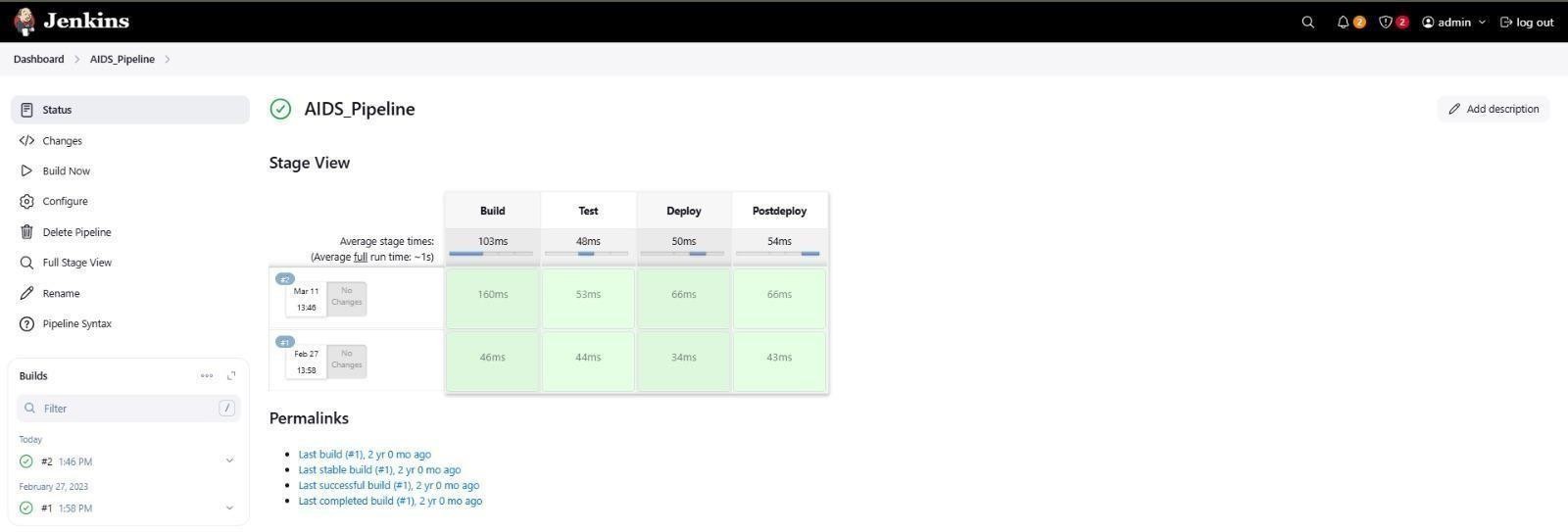
Console output after building:

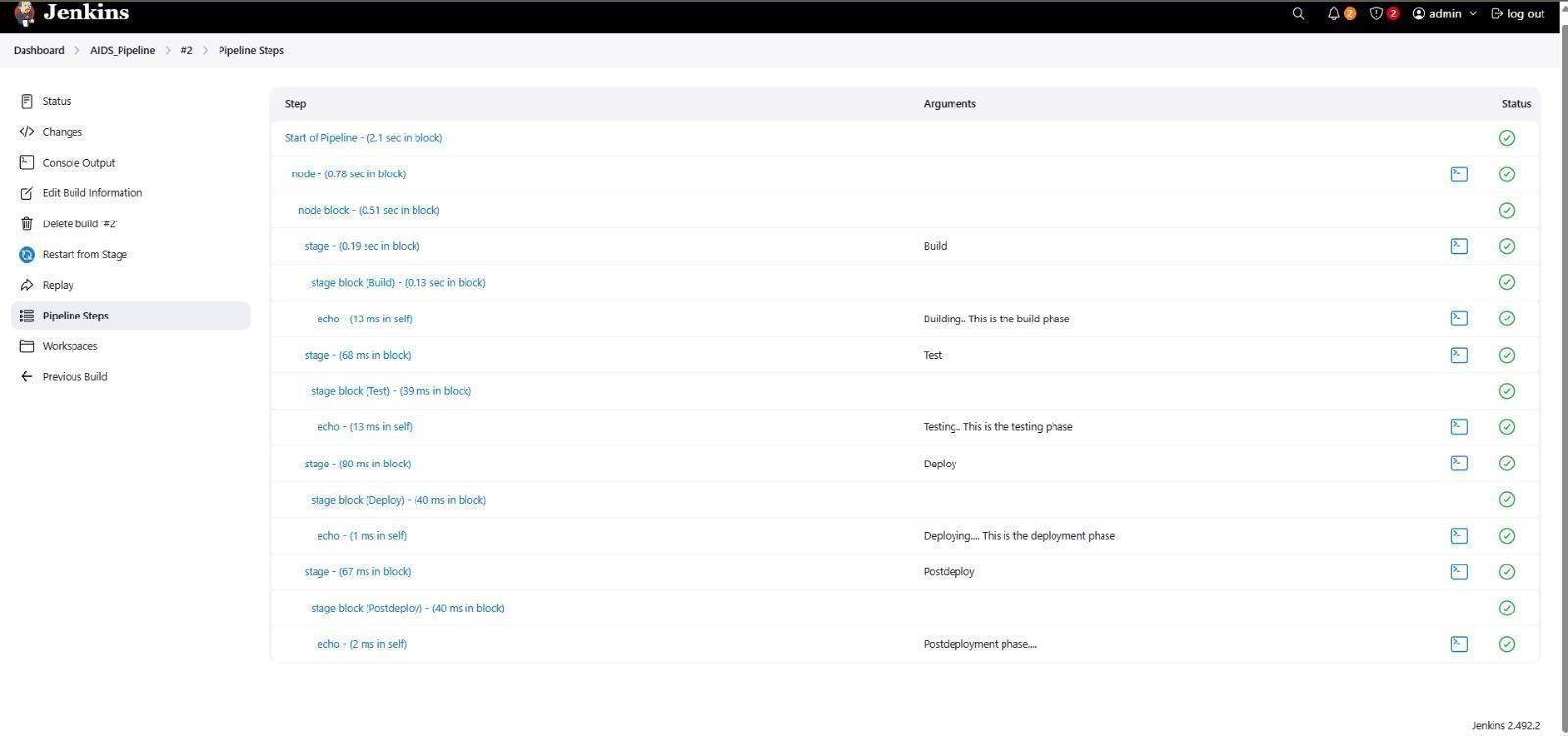


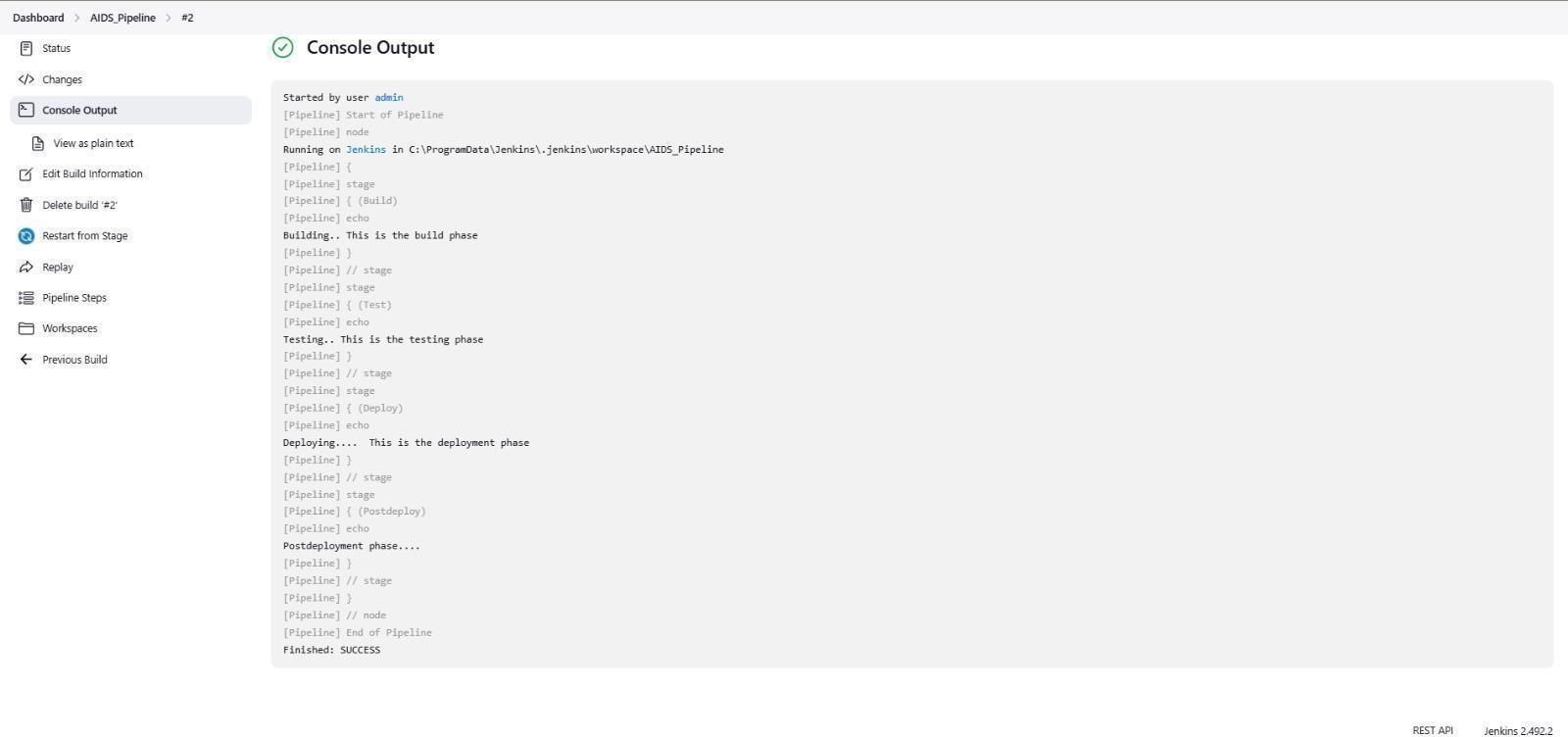
Some Screenshots:

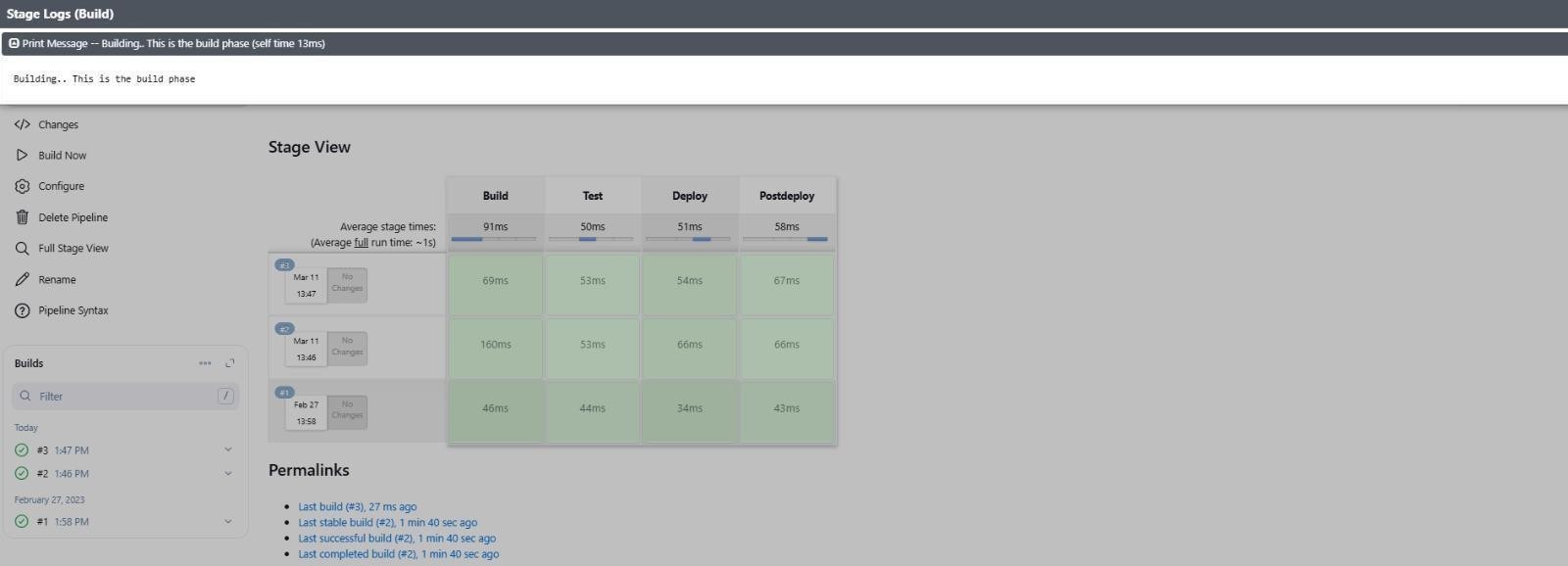












**Conclusion: Thus, we have successfully Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, created a pipeline script to Test and deploy an application over the tomcat server.**