**Batch: A4 Experiment Number: 4**

**Roll Number: 1814128 Name: Rhutuja Thakur**

**Aim of the Experiment:** Programming with Jenkins

**Output/Result:**

Java Program:

public class Multiplier {

    public static void main(String[] args) {

        int num1, num2;

        num1 = Integer.parseInt(args[0]);

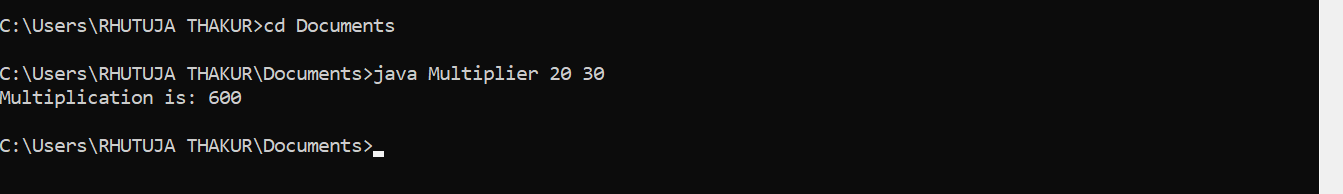
        num2 = Integer.parseInt(args[1]);

        System.out.println("Multiplication is: "+(num1\*num2));

    }

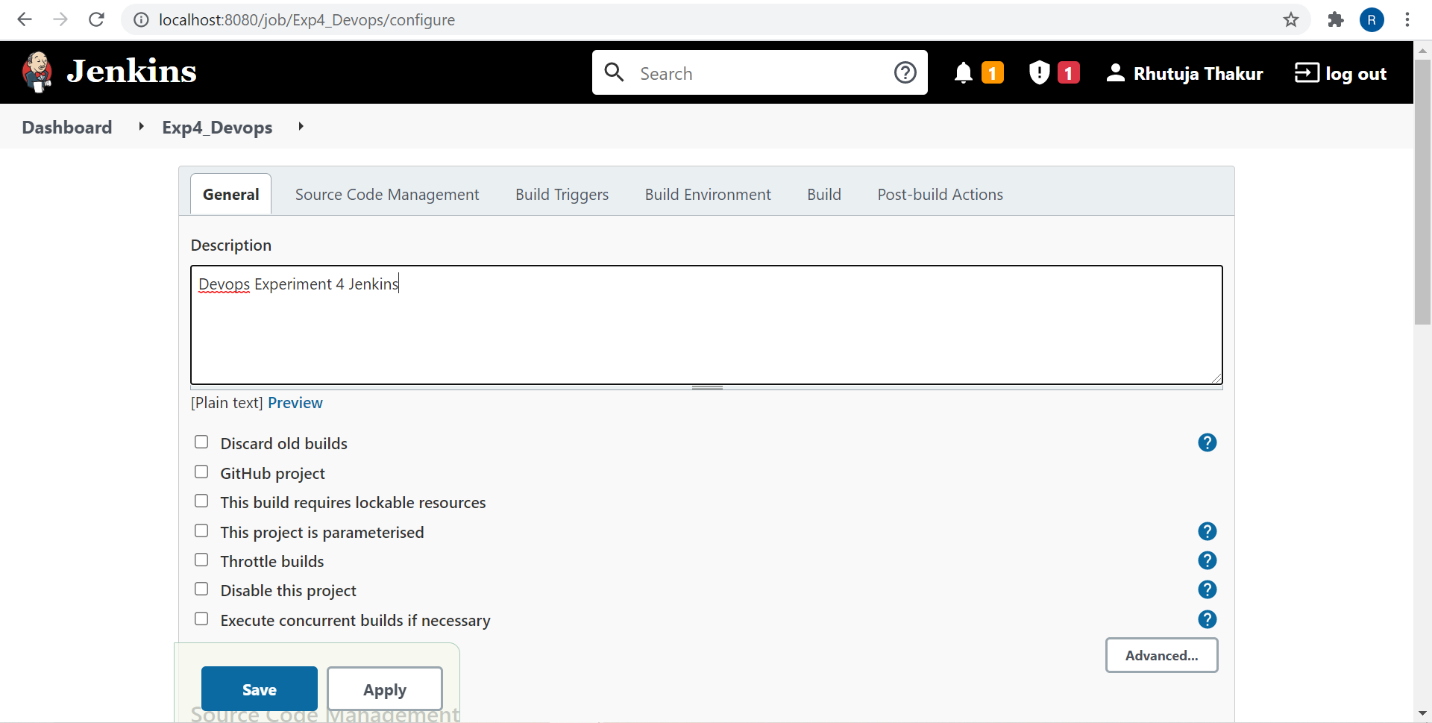
}

Execution of the Java program through command prompt on local machine

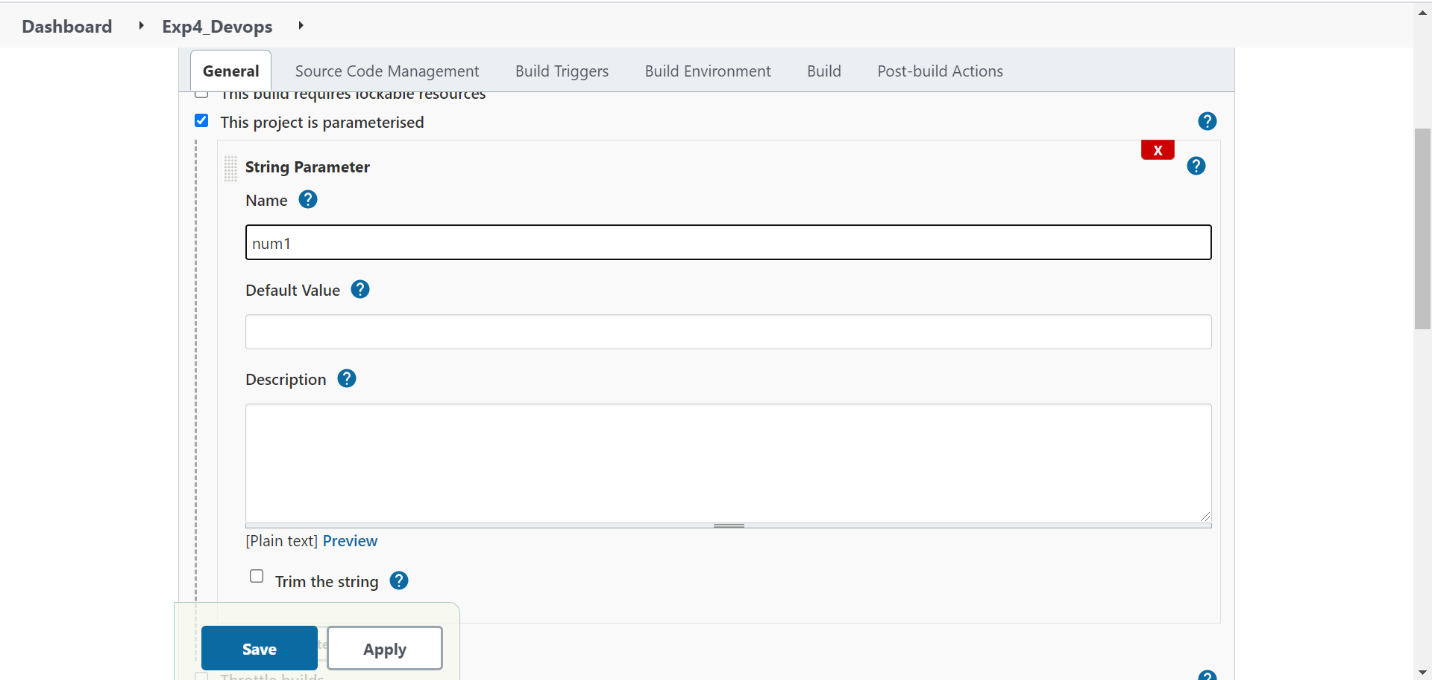


**JAVA Program using Jenkins:**

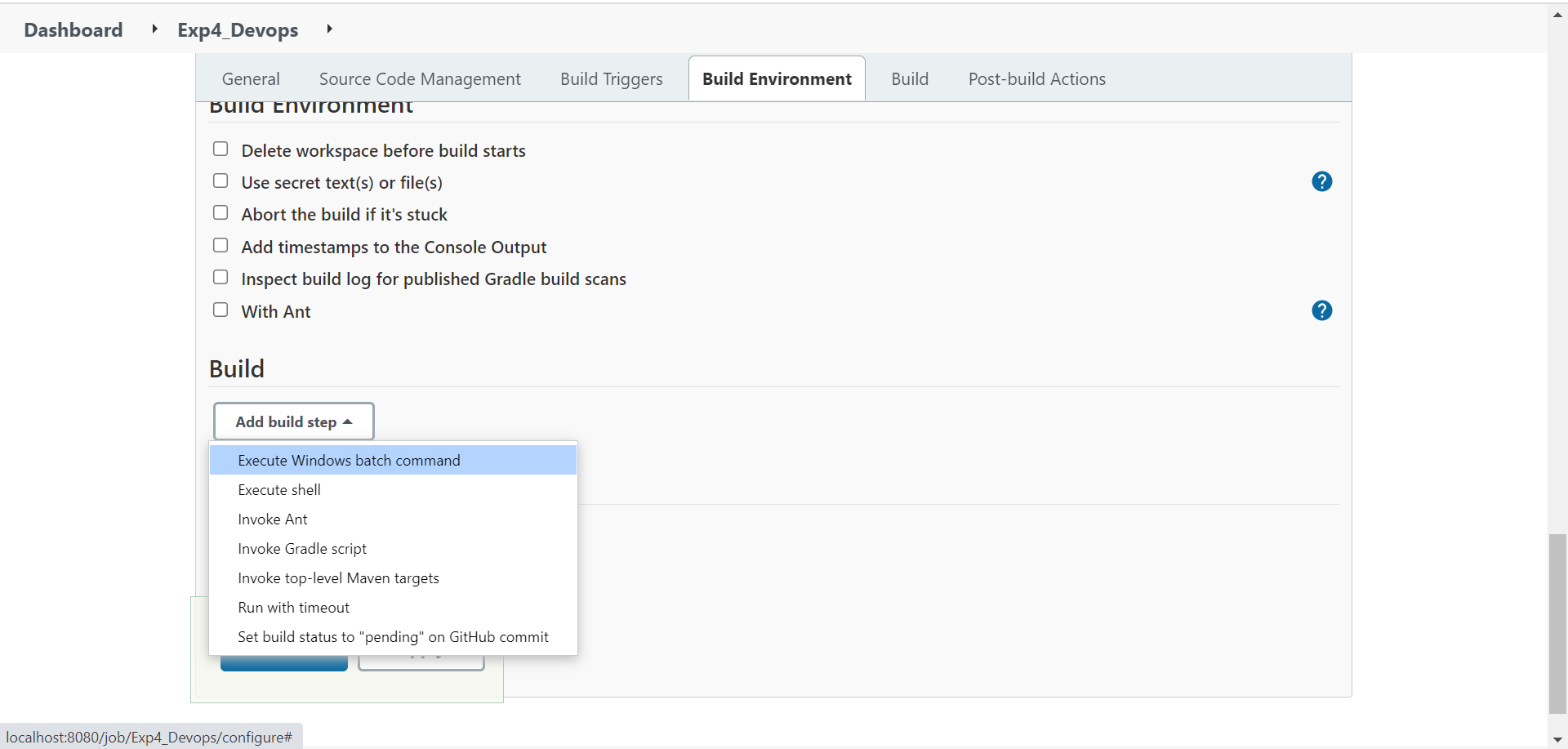
We created a new item as a parameterized program and added a general description for the same.



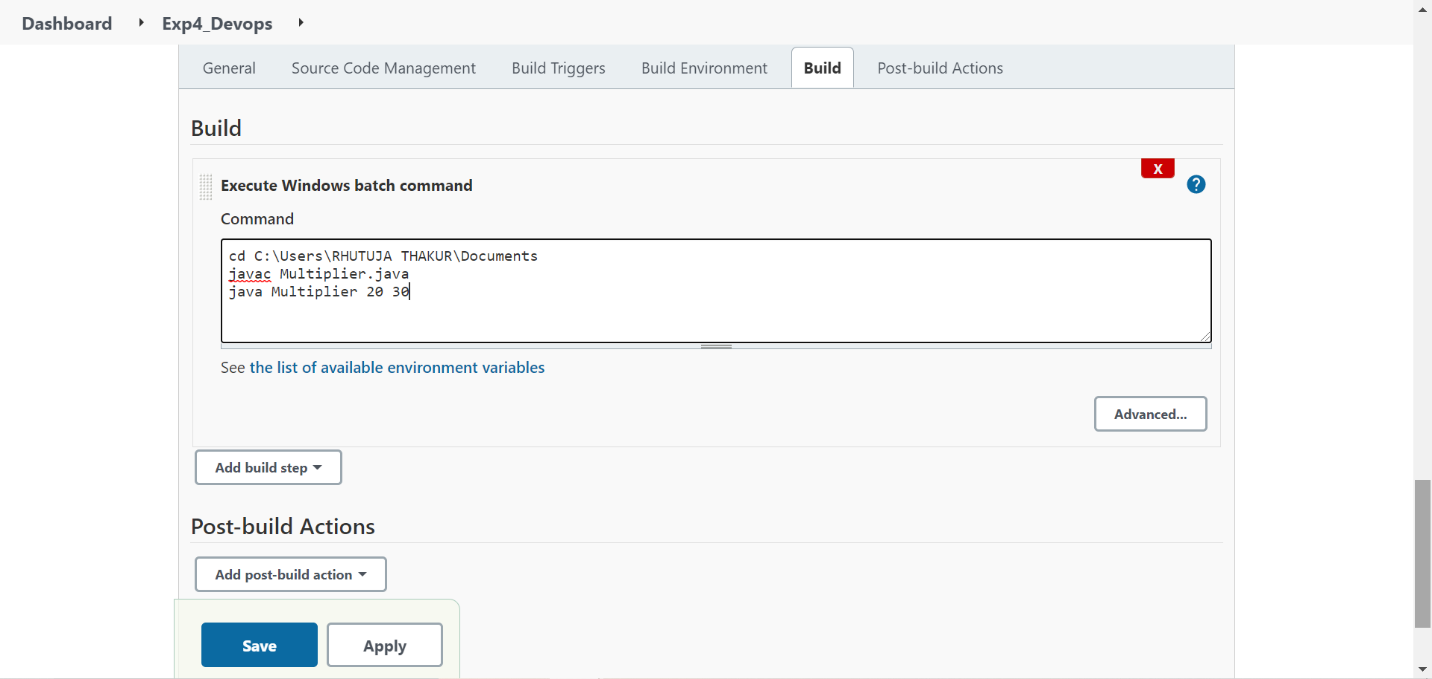
Then we select a string parameter for our program and name the variable as *num1 and num2*.



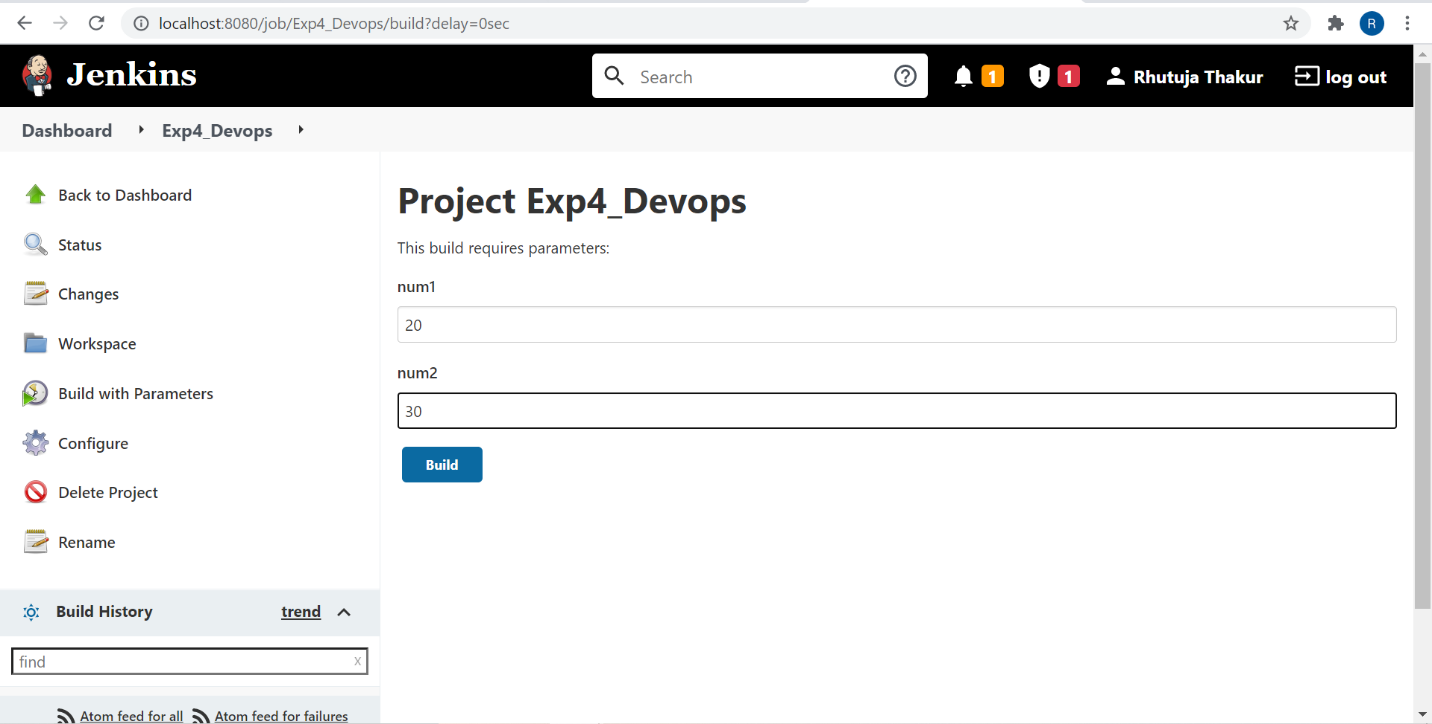
In the Build Environment tab, we select *Execute Windows Batch Command* option.



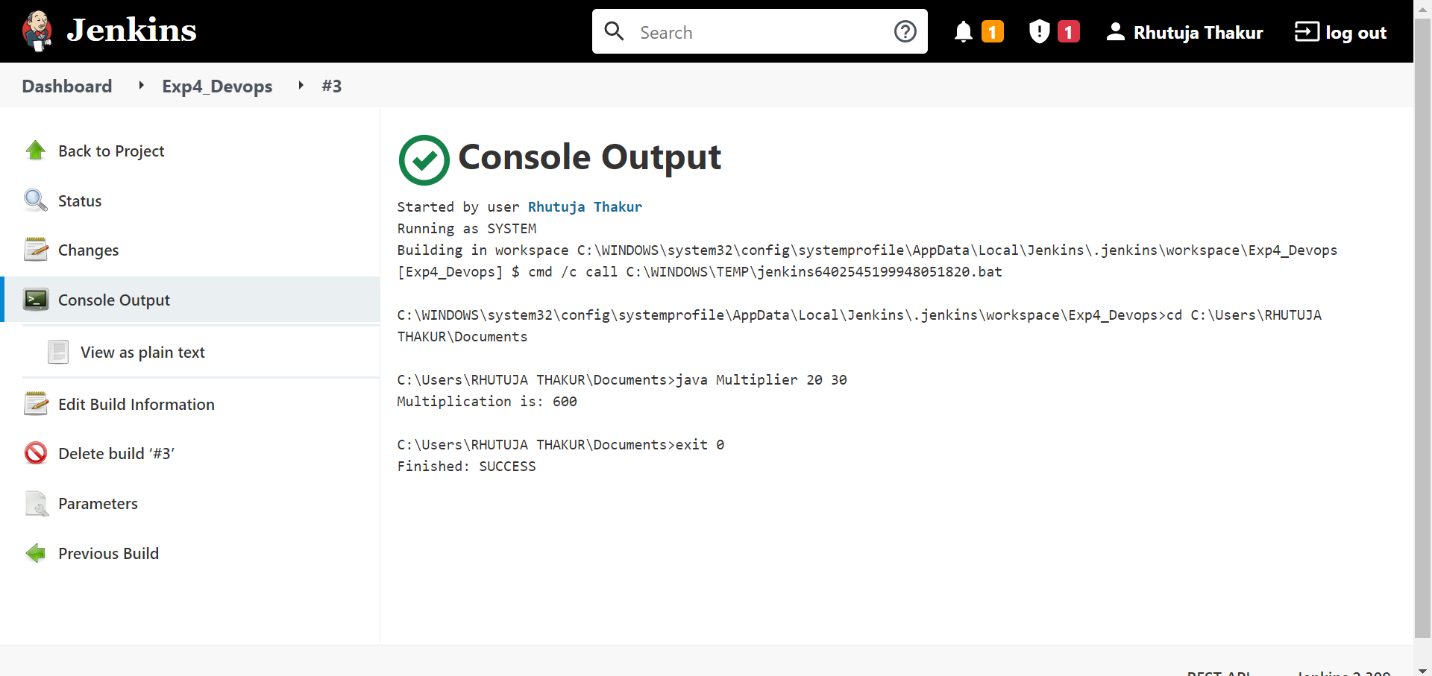
Now, we mention the batch commands that need to be executed via Jenkins, which start with changing the directory to the location where the Java file is present, then we run the *javac* command to create a *Class File*, which then is executed to get the output.



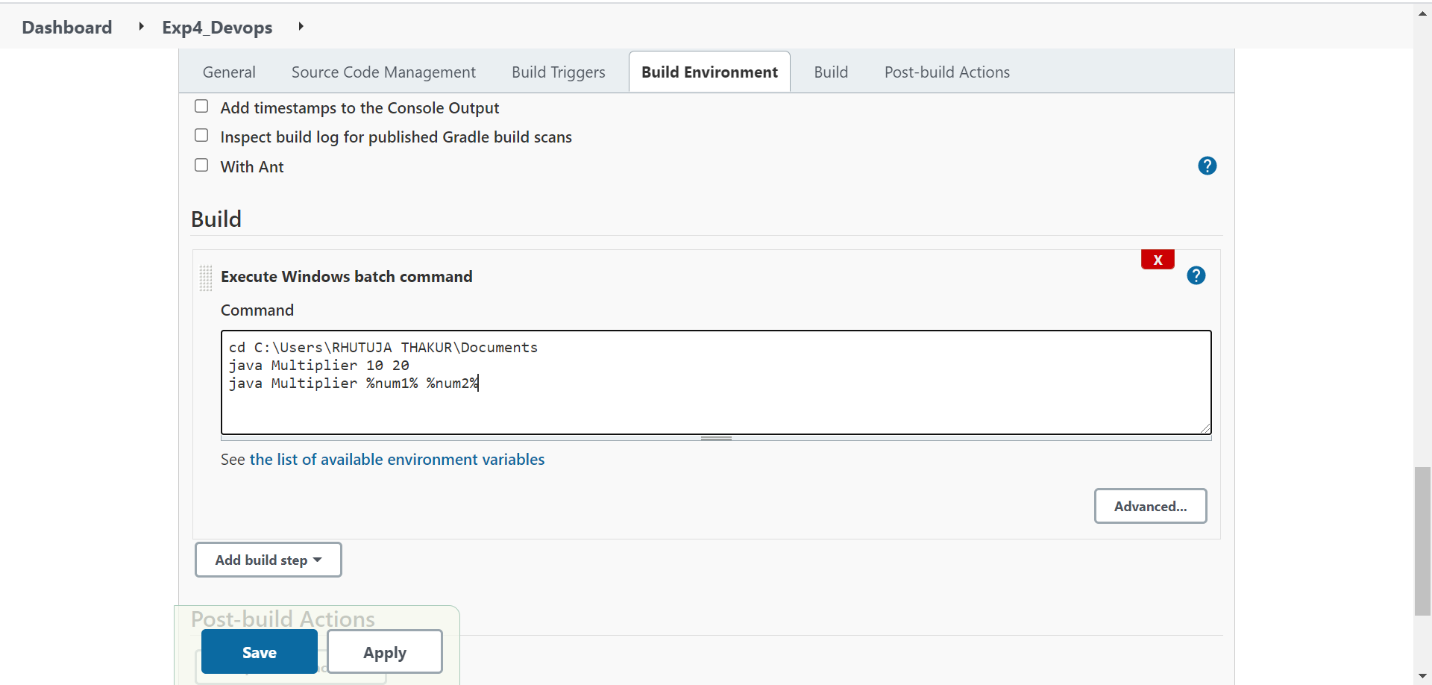
Next up we give values to the parameters we had created. We have assigned num1 to be 20 and num2 to be 30 this time. Next, we *Build* the project.



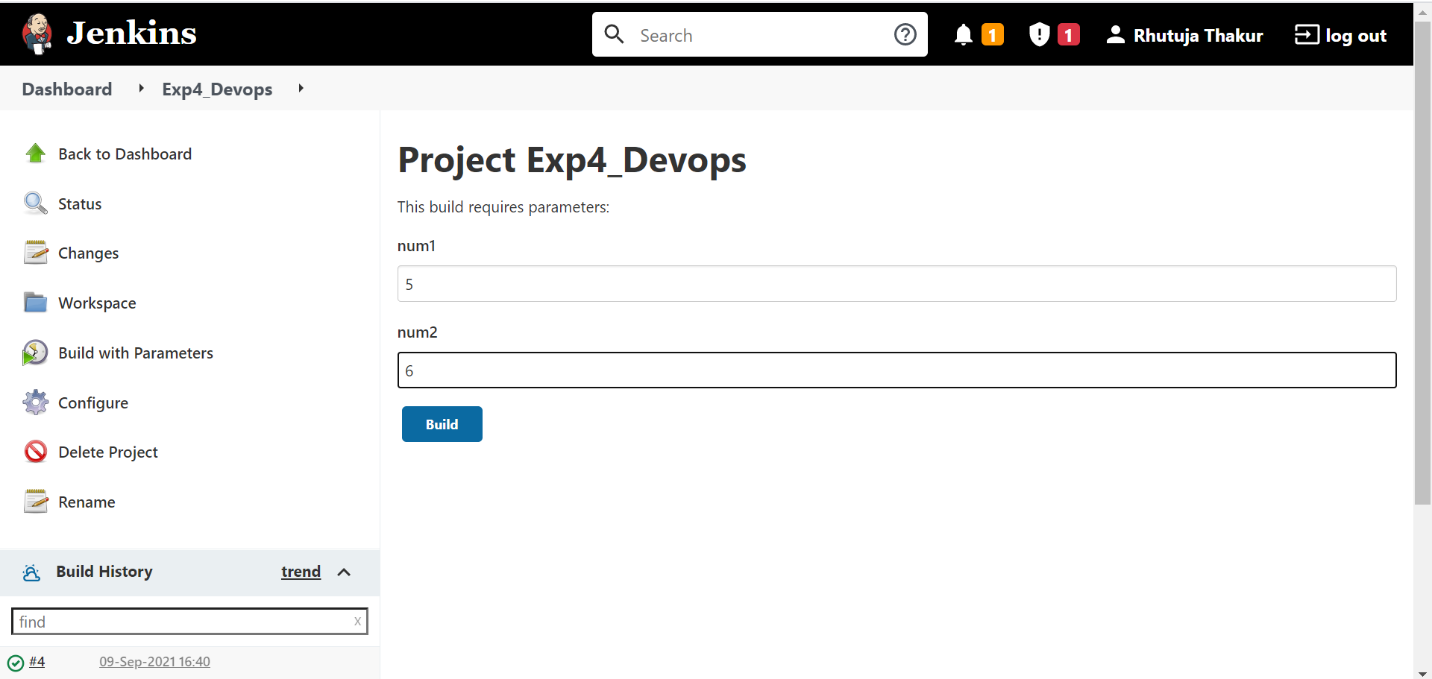
After hitting build, we get the Console Output, which in our case is ‘Multiplication is: 600’



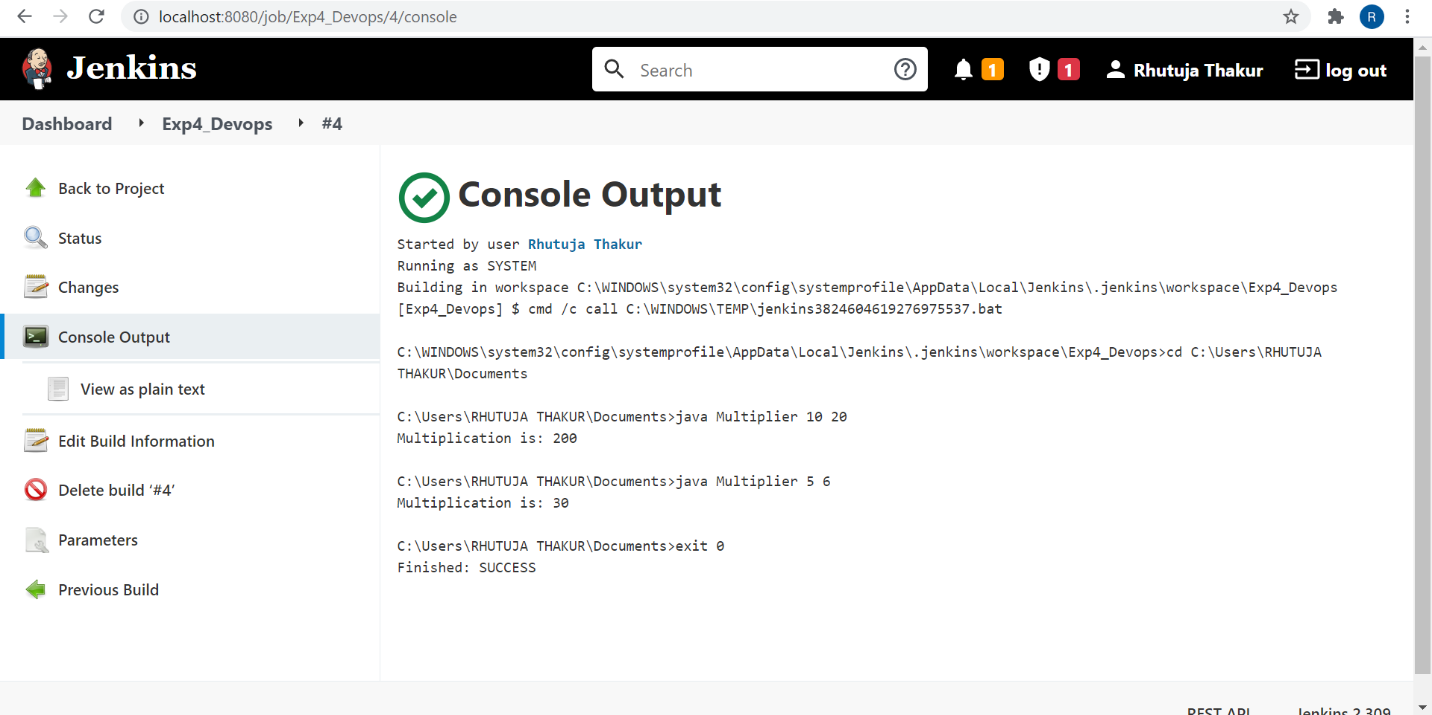
We changed the batch command so that it can take values from the passed arguments, as well as read them from stored variables in *num1 and num2*.



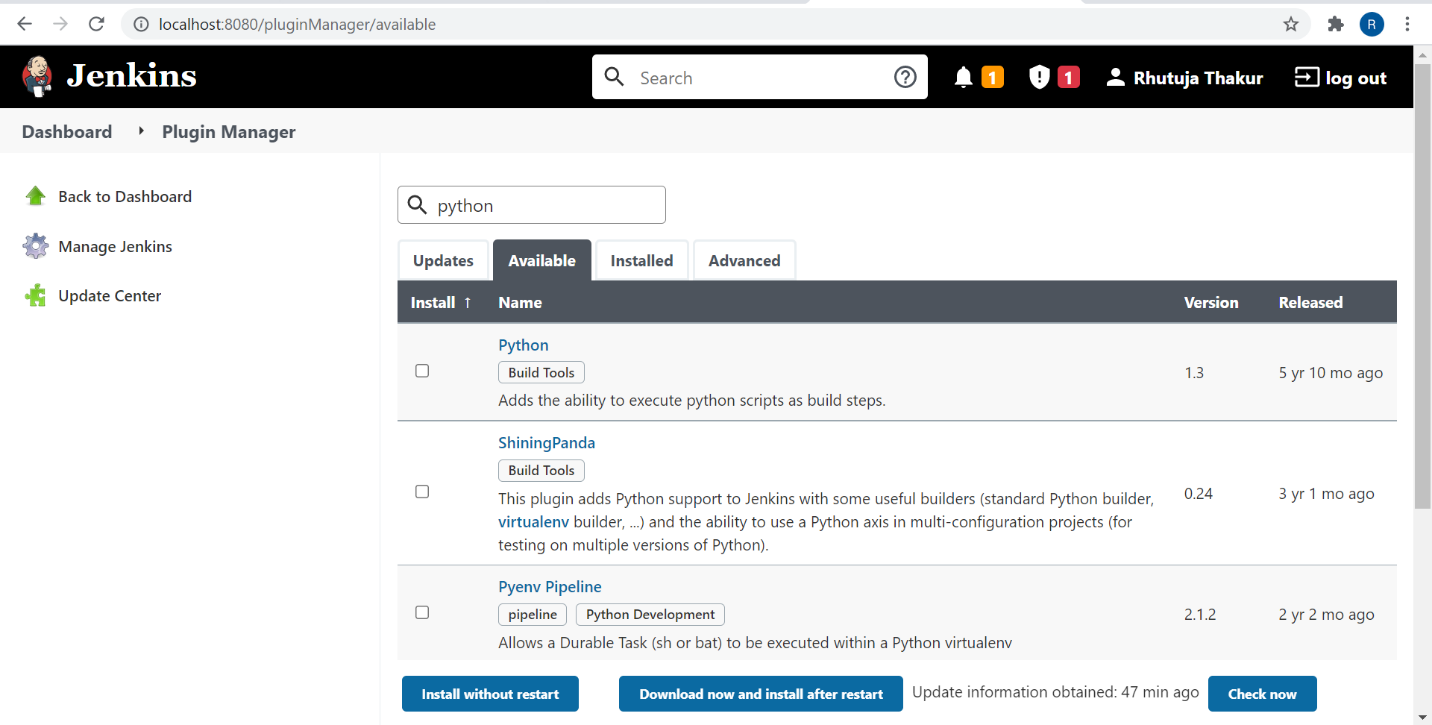
Here, we give values of 5 & 6 to num1 and num2 respectively.



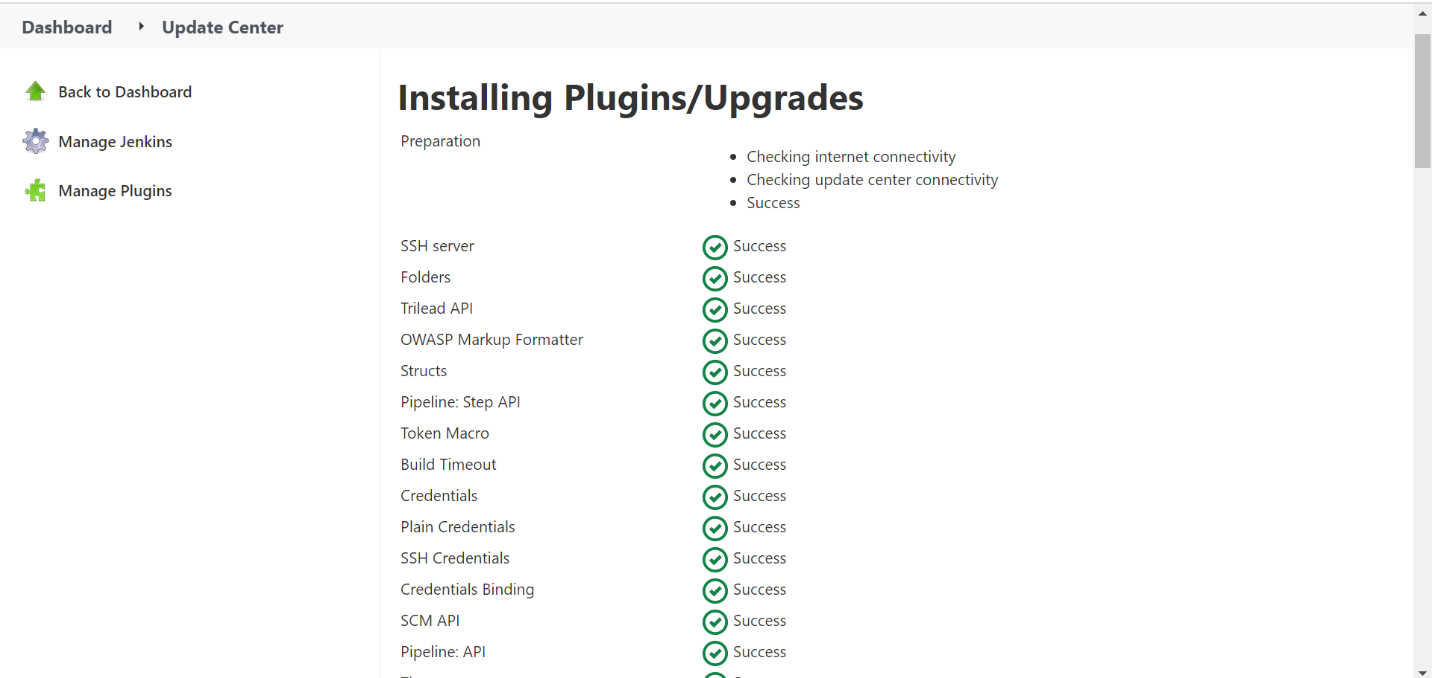
When we Build the item, we get a new console output showcasing that one output is ‘Multiplication is: 200’, which was for passed values via command line and the other is ‘Multiplication is: 30’ which was for values taken via the stored variables in Jenkin.



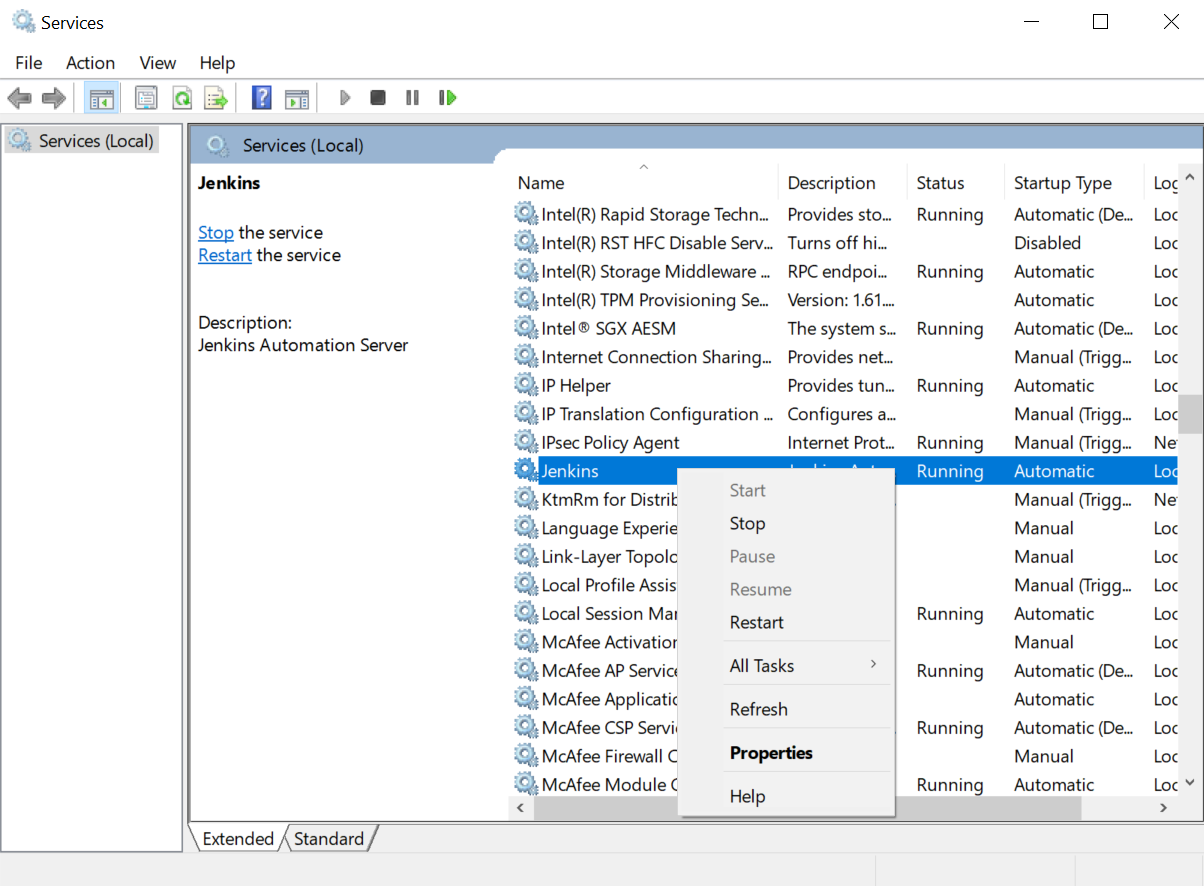
**Python Program using Jenkins:**

Dashboard > Manage Jenkins > Manage Plugins > Search in Available “Python” , then we *install without restart* the Python plugin for Jenkins

Here, we have successfully installed the Python plugin.

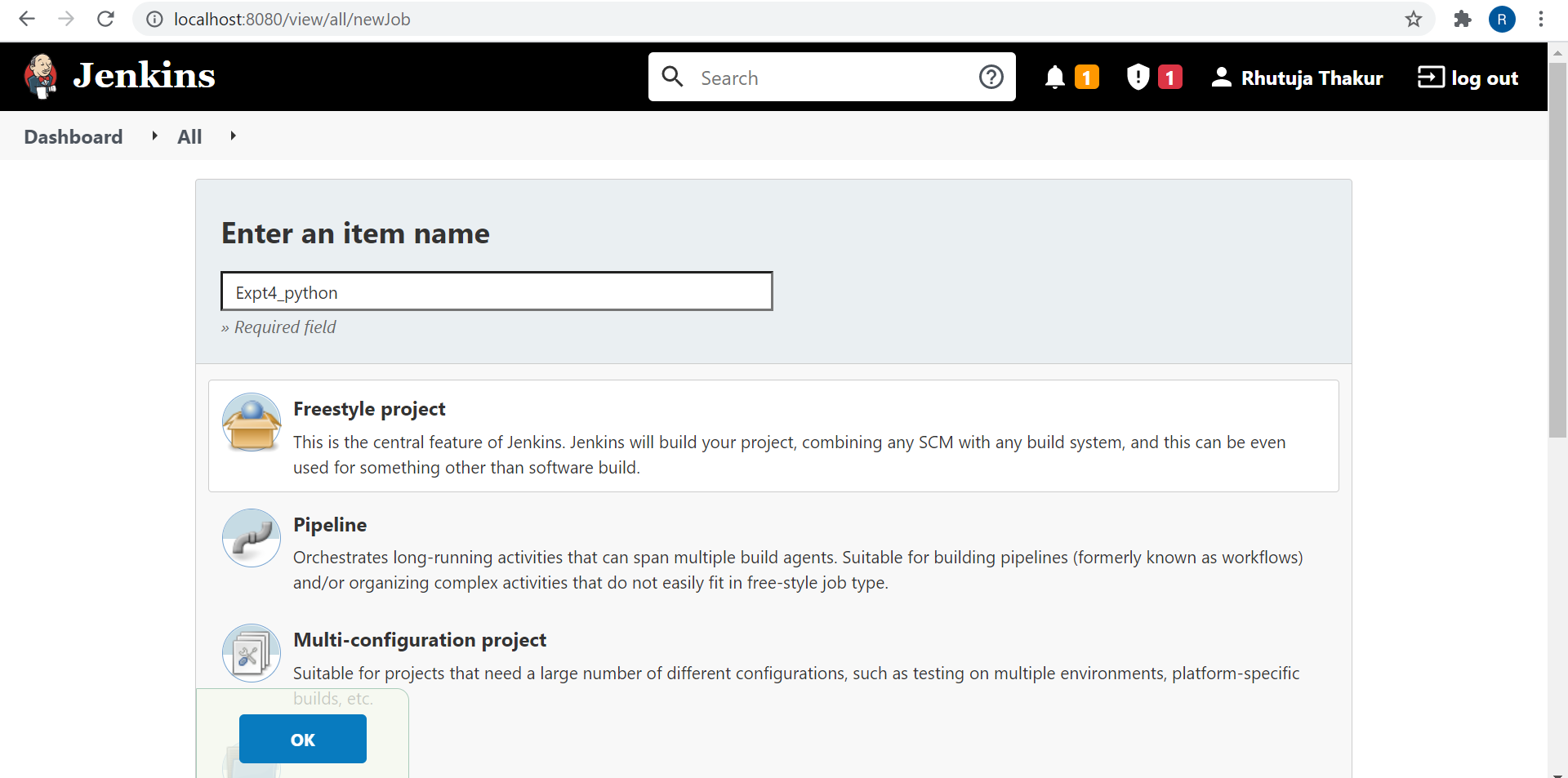


By going to the services.msc file by running Windows + R, we Restart by right clicking on Jenkins.

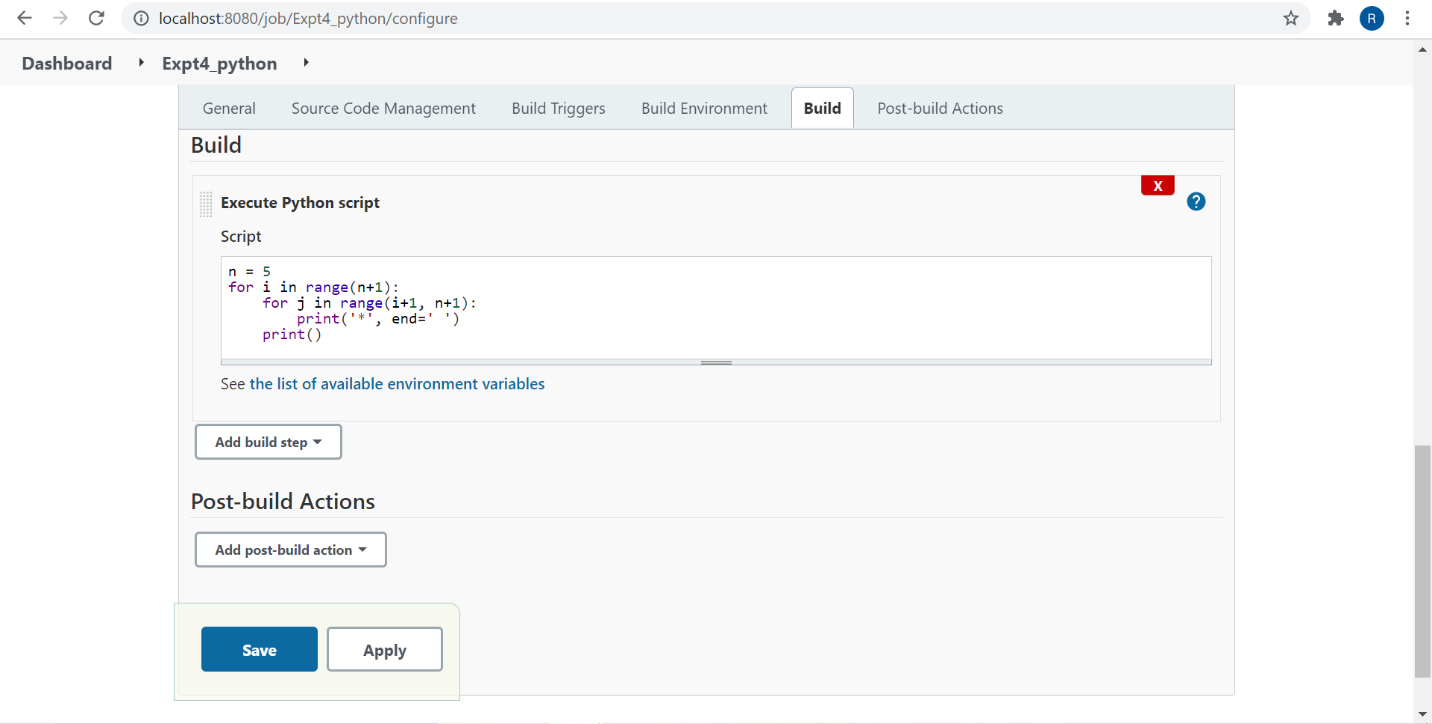


**Implicit Programming**

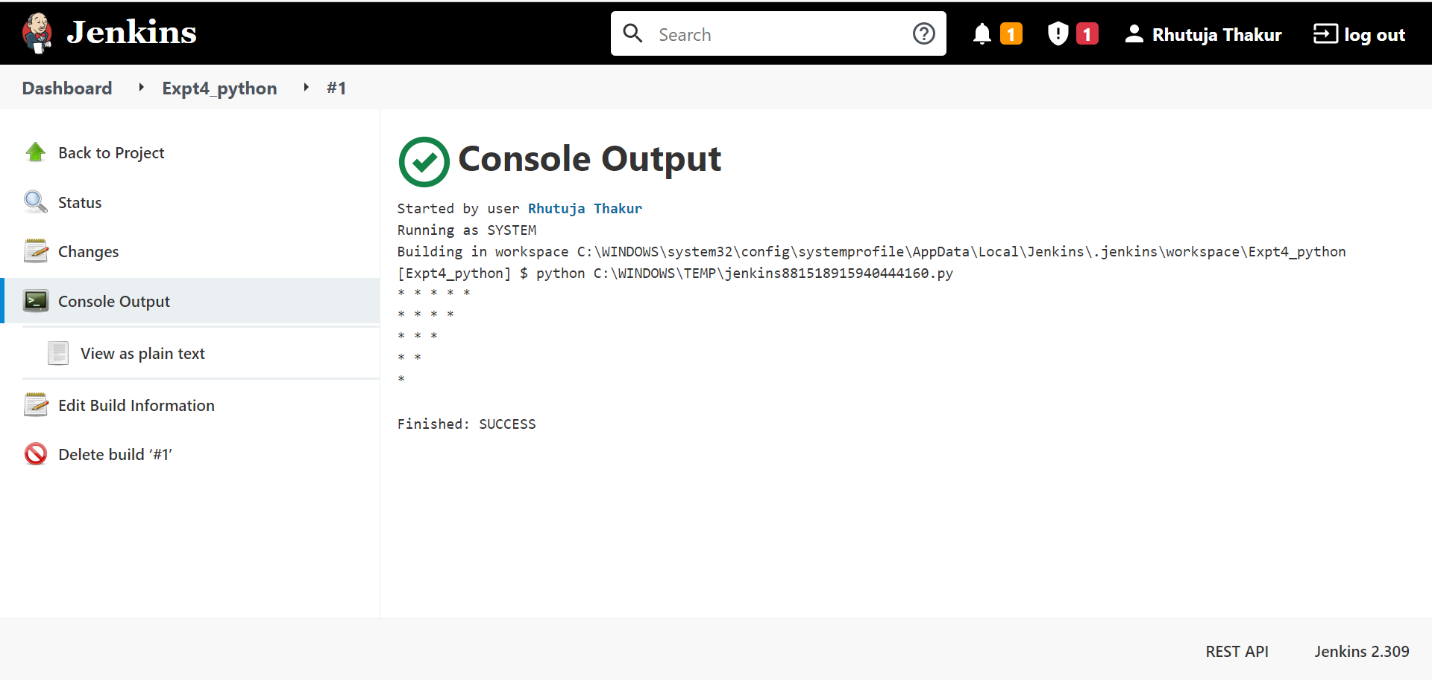
We create a new item as a Freestyle project, named *Expt4\_python*

****

Next up, in the Build tab, we choose the option to ‘Execute Python Script’ wherein we put our Python script which we wish to execute.



Next, we save the item and then hit the ‘Build’ option, and get the following output on the Console.



**Explicit Python:**

Python program:

n = 5

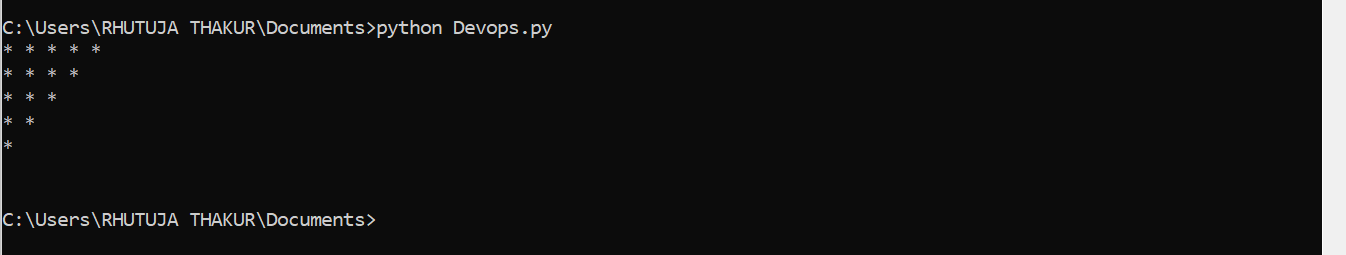
for i in range(n+1):

    for j in range(i+1, n+1):

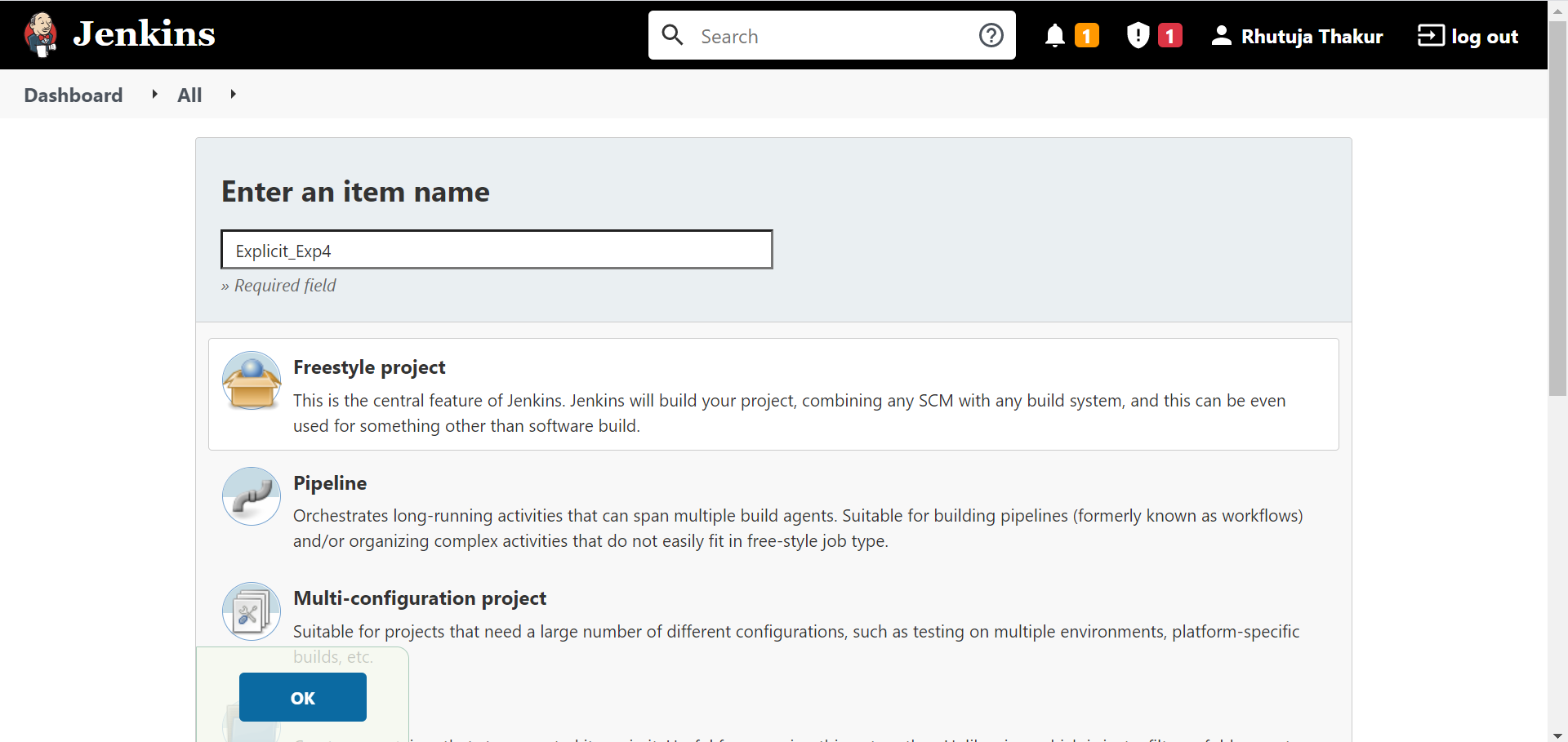
        print('\*', end=' ')

    print()

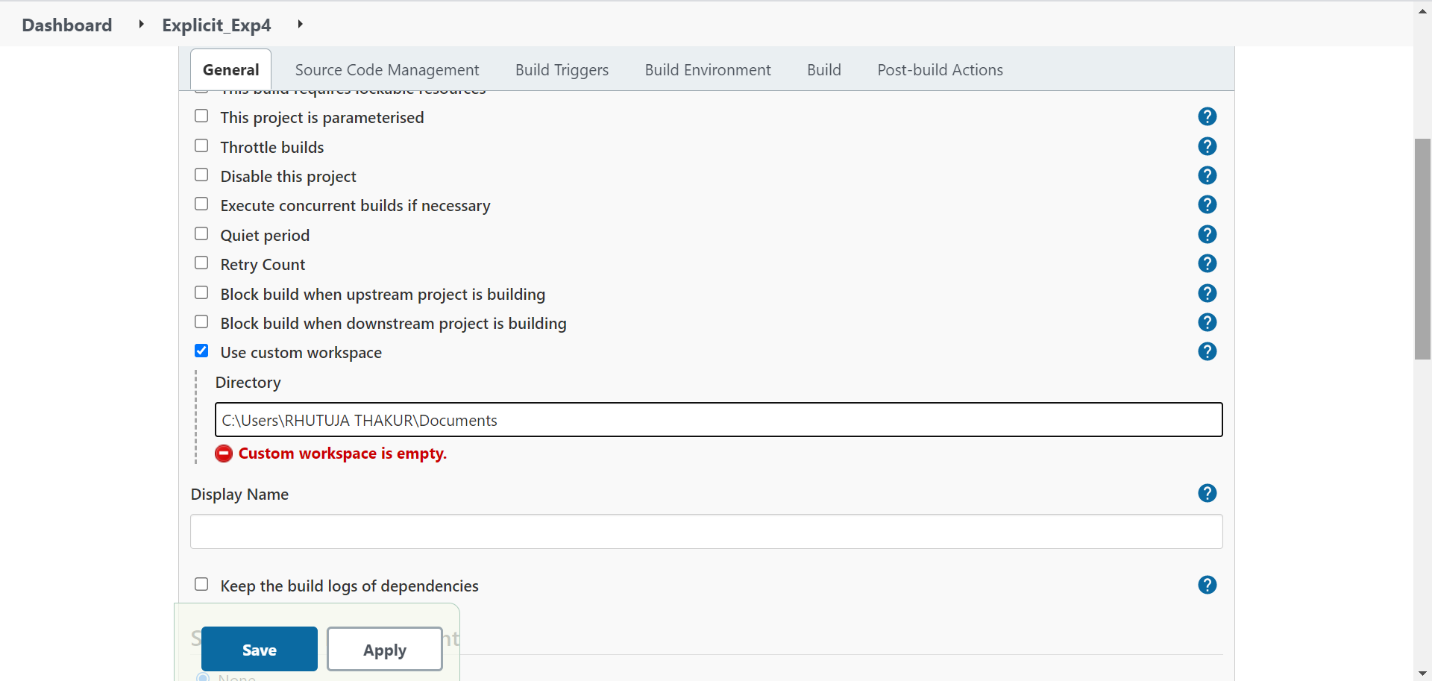
Execution of the Python program through command prompt on local machine:



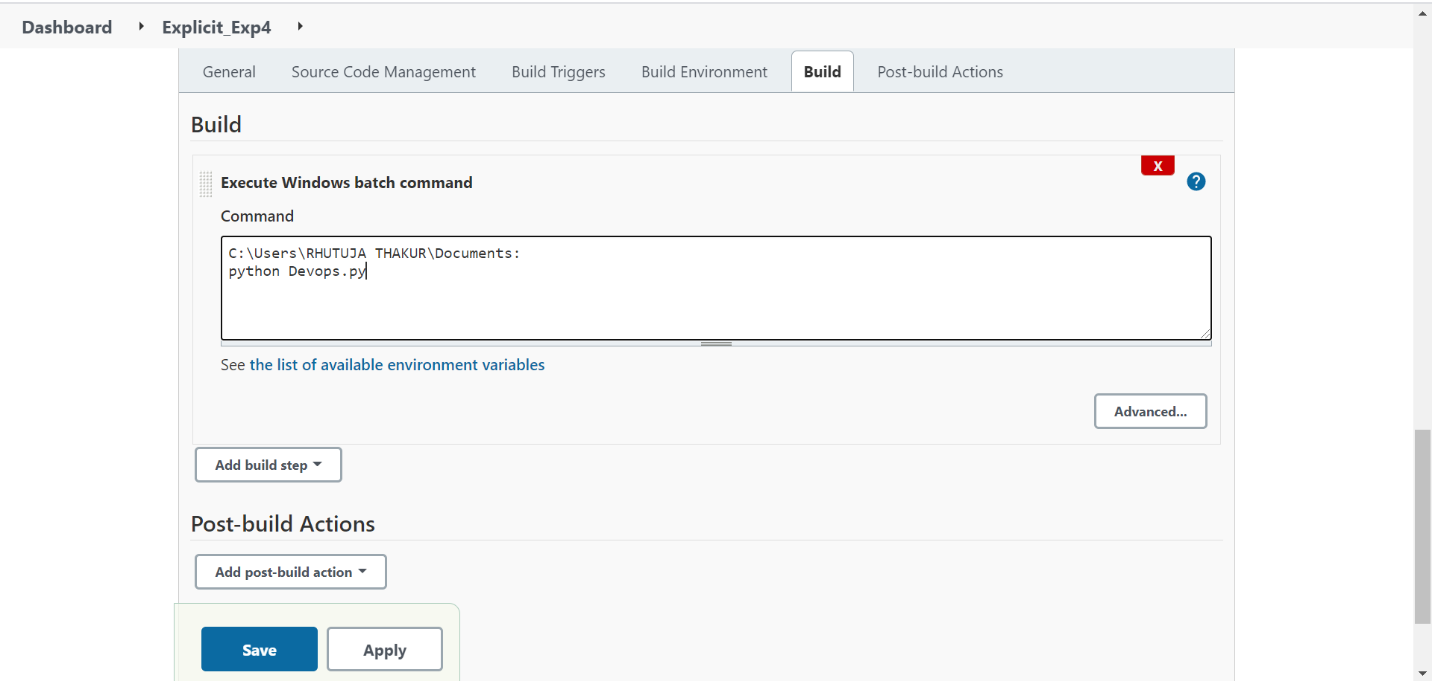
We create a new item as a Freestyle project, named *Explicit\_Exp4*



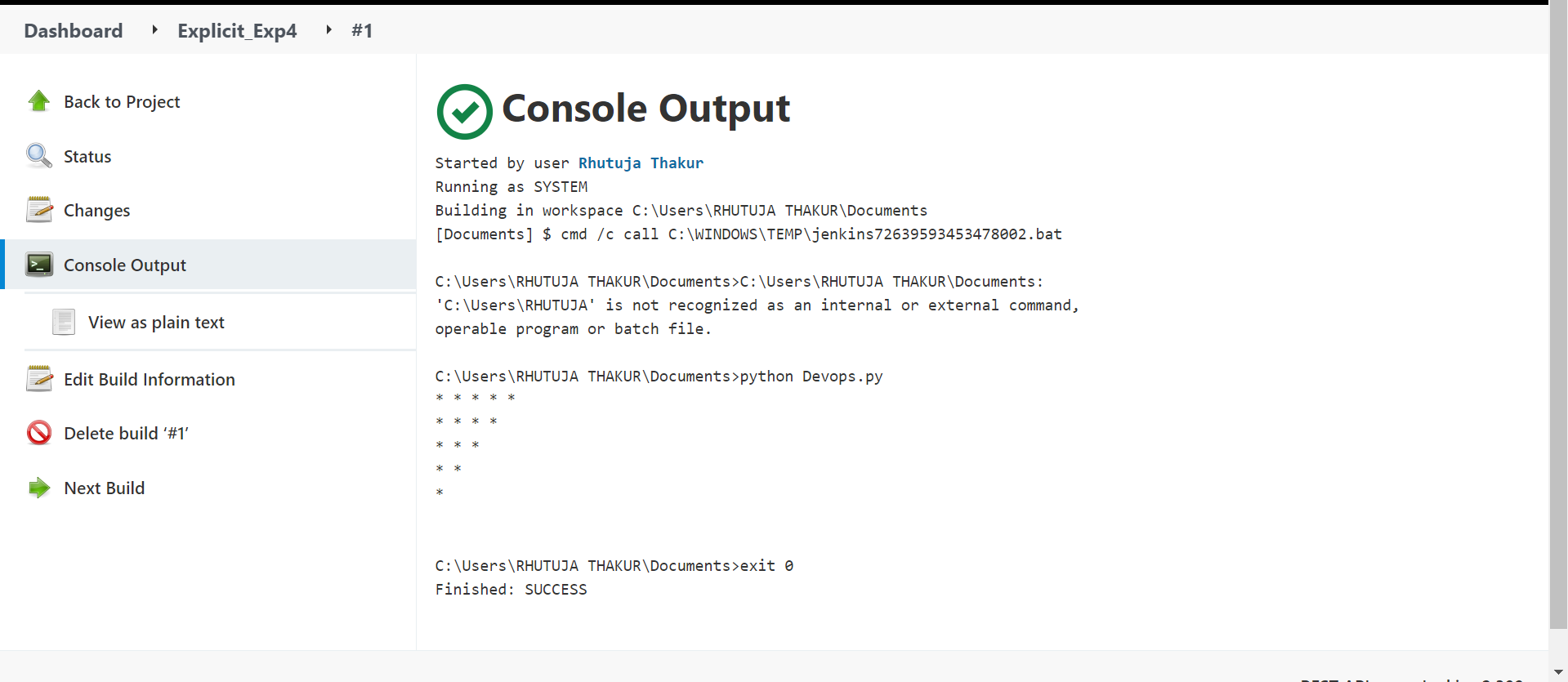
We now select the ‘Use custom workspace’ option in the General tab. Here we put up the path to the directory where we have stored the Python file.



In the Build tab, we choose the ‘Execute Windows Batch Command’ option and put the command to run the Python file, which is *python Devops.py*



And then we hit build and observe the following Console output.



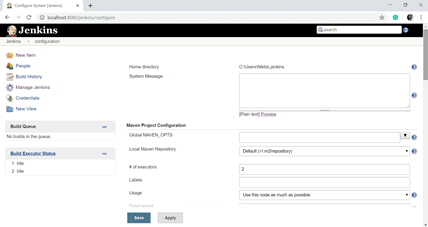
**Post Lab Question-Answers:**

1. **Explain the difference between system configuration and global tool configuration.**

Ans:

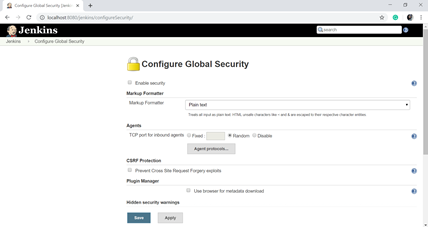
System Configuration

In this, we can manage paths to the various tools to use in builds, such as the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. Jenkins will add the required configuration fields dynamically when new plugins are installed.



Global Tool Configuration

Configure Global Security option provides the ability to set up users and their relevant permissions on the Jenkins instance. By default, you will not want everyone to be able to define builds or other administrative tasks in Jenkins. So Jenkins provides the ability to have a security configuration in place.



**Outcomes:**

CO3 – Comprehend the effective code building and testing process.

**Conclusion (based on the Results and outcomes achieved):**

In this experiment, we learnt about Java and python programming with Jenkins and explored the creation of parametrized and non parametrized projects using new item feature, we also installed the python Plugin and implemented the Python script both implicitly and explicitly.