UNIT III CONTINUOUS INTEGRATION USING JENKINS 6

Install & Configure Jenkins, Jenkins Architecture Overview, Creating a Jenkins Job, Configuring a Jenkins job, Introduction to Plugins, Adding Plugins to Jenkins, Commonly used plugins (Git Plugin, Parameter Plugin, HTML Publisher, Copy Artifact and Extended choice parameters). Configuring Jenkins to work with java, Git and Maven, Creating a Jenkins Build and Jenkins workspace.

INTRODUCTION TO PLUGINS

Plugins are extensions that enhance Jenkins functionality. They allow Jenkins to integrate with other tools, support new build steps, report formats, notifications, version control systems, and more. Essentially, plugins make Jenkins highly flexible and customizable. Plugins use their own set of Java Annotations and design patterns that define how the plugin is instantiated, extension points, the function of the plugin and the UI representation in the Jenkins Web UI

Benefits of using Plugins:

Following are the benefits that are observed when the Jenkins plugins are used

1. **Extended functionality:** Plugins can be used to add new features to jenkins. Many plugins are open-source and maintained by the Jenkins community. This means that the improvements are made by a larger user base.
2. **Task automation:** Plugins can be used to automate various tasks related to software development such as building, testing, deployment of library files to executable environment.
3. **Scalability:** We can add or remove the plugin as per the requirements of the project and thereby Jenkins meets the requirements of organization.
4. **Security:** Jenkins plugins are useful to make the job secure by iterating with security tools, vulnerability scanners.
5. **Continuous Improvement:** Jenkins is an automated tool. Developers continuously create new plugins or improve the existing ones. Thus continuous improvement is made with the latest technologies.
6. **Increased flexibility:** Plugins can be used to increase the flexibility of Jenkins. For example, Amazon EC2 plugin allows you to deploy the application from cloud platform or Git plugin allows you to deploy the application from GitHub repository. There are various ways by which the desired activity can be carried out in Jenkins.

Adding Plugins To Jenkins

- The simplest and most common way of installing plugins is through the Manage Jenkins → Plugins view, available to administrators of a Jenkins environment.

- Locate the desired plugin from the list or else we can search the plugin by typing plugins names. Suppose if we wish to install HTML Publisher plugin then we can either locate it in the list or we can simply type the name of that plugin in the search window.
- Click install button

Thus the plugin is installed

COMMONLY USED PLUGINS

Following is a list of some popularly used plugins are:

1. **Git Plugin:** The git plugin provides fundamental git operations for Jenkins projects. It can poll, fetch, checkout, branch, list, merge, tag and push repositories.
2. **Docker Plugin:** The jenkins cloud plugin for Docker is the most effective solution for Devops engineers to integrate Jenkins with Docker.
3. **Amazon EC2Plugin:** Amazon EC2 plugin lets Jenkins start up new EC2 on demand and shut down them when they are no longer needed.
4. **SonarQube plugin:** SonarQube is an open source tool used for continuous code quality inspection. The Jenkins monitoring plugin allows us to integrate SonarQube intoJenkins so that we can easily analyze a code while running a Jenkins job that comes with SonarQube execution.
5. **Jira Plugin:** Jira plugin is one of the most popular plugins. It is an open-source plugin that integrates Jenkins with the Atlassian Jira Software (both Cloud and Server versions), enabling the DevOps teams more visibility into the development pipeline.

GIT PLUGIN

The Git plugin is used to perform fundamental git operations for Jenkins project. When the Git plugin is installed in Jenkins then we can perform pull, fetch, branch, list, merge or push operations.

How to install the Git Plugin?

For installing the Git plugin just login to the Jenkins and click the Manage Jenkins. Open the plugins section and install the GitHub plugin.

Once the Jenkins Git Plugin is installed and configured, our Jenkins build jobs can poll any local or remote repositories for new commits. The cron expression is used for job scheduling periodically.

Example Demo

Step 1: Create a simple Java program. I have created a folder named MyJavaPrograms and inside it created a simple Java program as follows -

test.java

```
public class test
{
	public static void main(String args[])
	{
		for(int i=1; i<=5; i++)
		{
			System.out.println("Welcome Anuradha");
		}
	}
}
```

Step 2: Open the command prompt, switch to that folder and execute the above Java program. It is illustrated by the following screenshot.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

E:\MyJavaPrograms>javac test.java
E:\MyJavaPrograms>java test
Welcome Anuradha
Welcome Anuradha
Welcome Anuradha
Welcome Anuradha
Welcome Anuradha

E:\MyJavaPrograms>
```

Step 3: Now we will create a Git repository and push this repository on GitHub. First of all we will initialise the Git repository by using git init command.

```
C:\Windows\System32\cmd.exe
E:\MyJavaPrograms>git init
Initialized empty Git repository in E:/MyJavaPrograms/.git/
E:\MyJavaPrograms>git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.class
```

test.java
nothing added to commit but untracked files present (use "git add" to track)
E:\MyJavaPrograms>
then add the java and class files to git repository -
C:\Windows\System32\cmd.exe
E:\MyJavaPrograms>git add .

E:\MyJavaPrograms>git status
On branch master
No commits yet
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: test.class
new file: test.java
E:\MyJavaPrograms>
Now we will commit the changes -
C:\Windows\System32\cmd.exe
E:\MyJavaPrograms>git commit -m "First Commit For Java program"
[master (root-commit) 4c6040b] First Commit For Java program
2 files changed, 6 insertions(+)
create mode 100644 test.class
create mode 100644 test.java
E:\MyJavaPrograms>
Now we will Create a repository on GitHub and push the above committed git repository on the GitHub.
Open the browser and Go to GitHub.com and log in.
Click “New Repository”.
Enter the repository name, for example:
MyJavaPrograms

- Choose Public or Private as desired.
  Click “Create a new repository”.
  Copy the repository URL
  After creation, copy the HTTPS URL:
  https://github.com/AnuradhaP/MyJavaPrograms.git
- Link local repository to GitHub
  Open the command prompt and navigate to your local project directory (MyJavaPrograms).

Add the remote repository:
git remote add origin https://github.com/AnuradhaP/MyJavaPrograms.git

- Push code to GitHub
  Push your local commits to the remote repository:
  git push -u origin master
- Refresh your GitHub repository page and we now see the uploaded files, such as:
  test.java
  test.class
- The commit message (e.g., "First Commit for Java program") will also appear beside the files
- The next step is to create a job in Jenkins that uses the GitHub plugin to access the GitHub repository.

PARAMETERIZED PLUGIN

A Parameterized Job in Jenkins allows you to create jobs that take parameters (like strings, numbers, or boolean values) when triggered. This helps you customize the job's behavior based on the values provided at runtime. It's useful for jobs where the same task may need to be executed with different configurations.

Use Cases:

- Dynamic Builds – Run the same job with different inputs without modifying the configuration manually.
- Improved Reusability – Use a single Jenkins job for multiple environments, configurations, or test cases.
- Time-Saving Automation – Execute builds, tests, and deployments automatically with varying parameters instead of creating multiple jobs.
- Efficient CI/CD Pipelines – Makes software testing and application deployment more flexible and scalable.

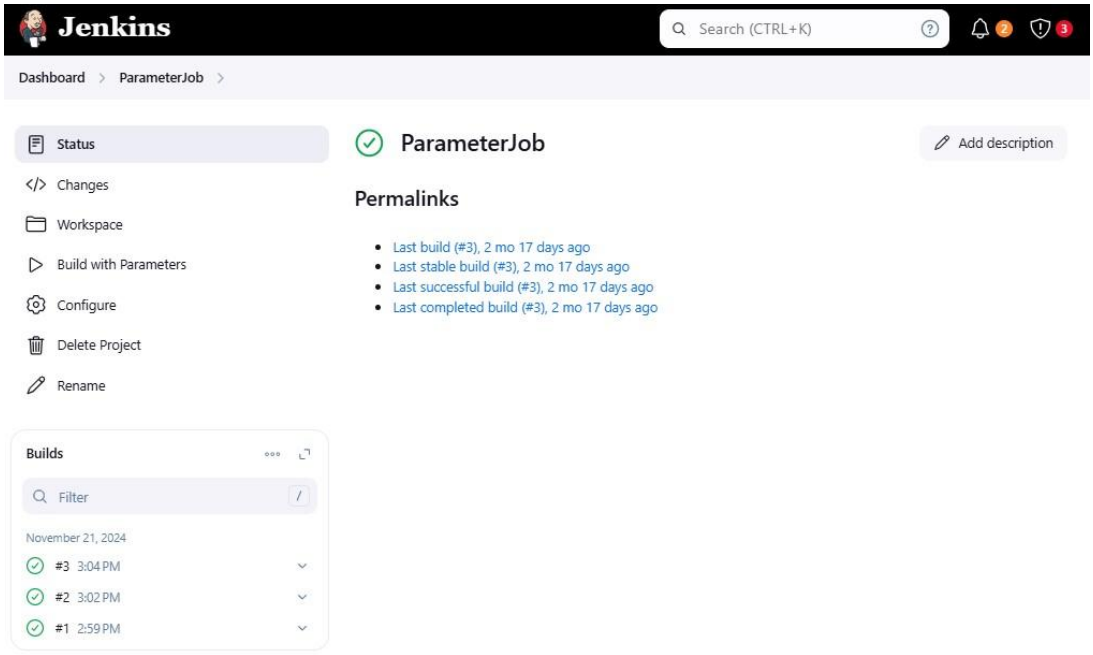
Here are the steps to create a Parameterized Job in Jenkins:

Step 1: Select the Jenkins Job to Add Parameters

Select the Job which you are planning to add parameters:
jenkins job list

Step 2: Configure the Jenkins Job

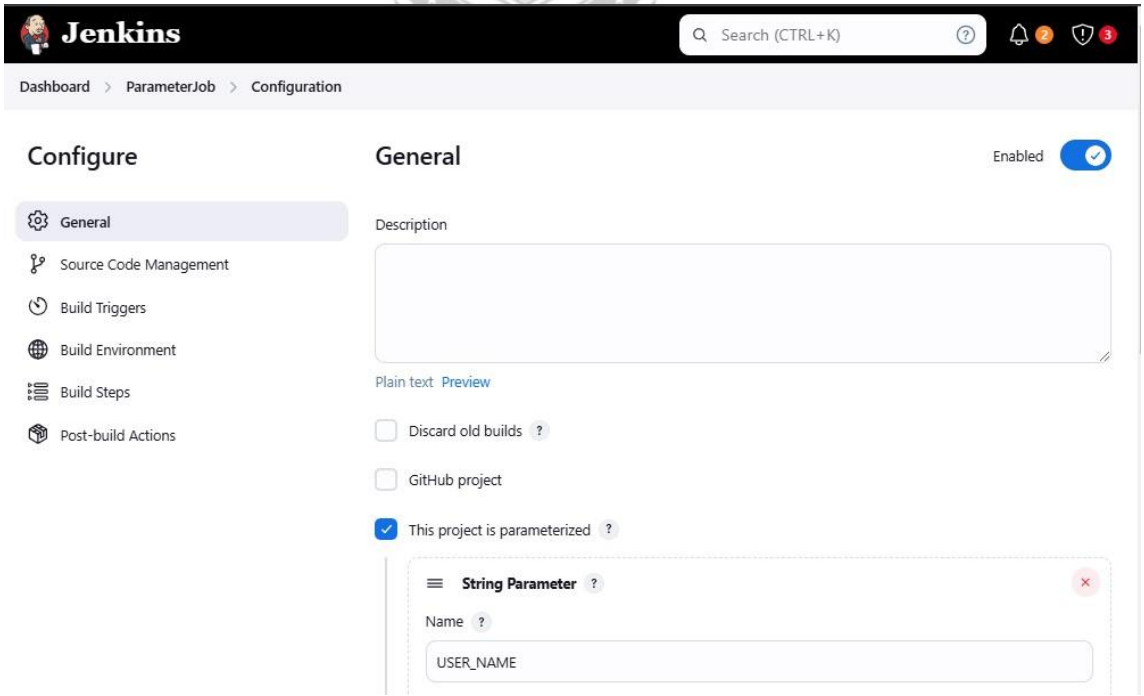
You can Configure the Job with the left Side of the job list which shown below.



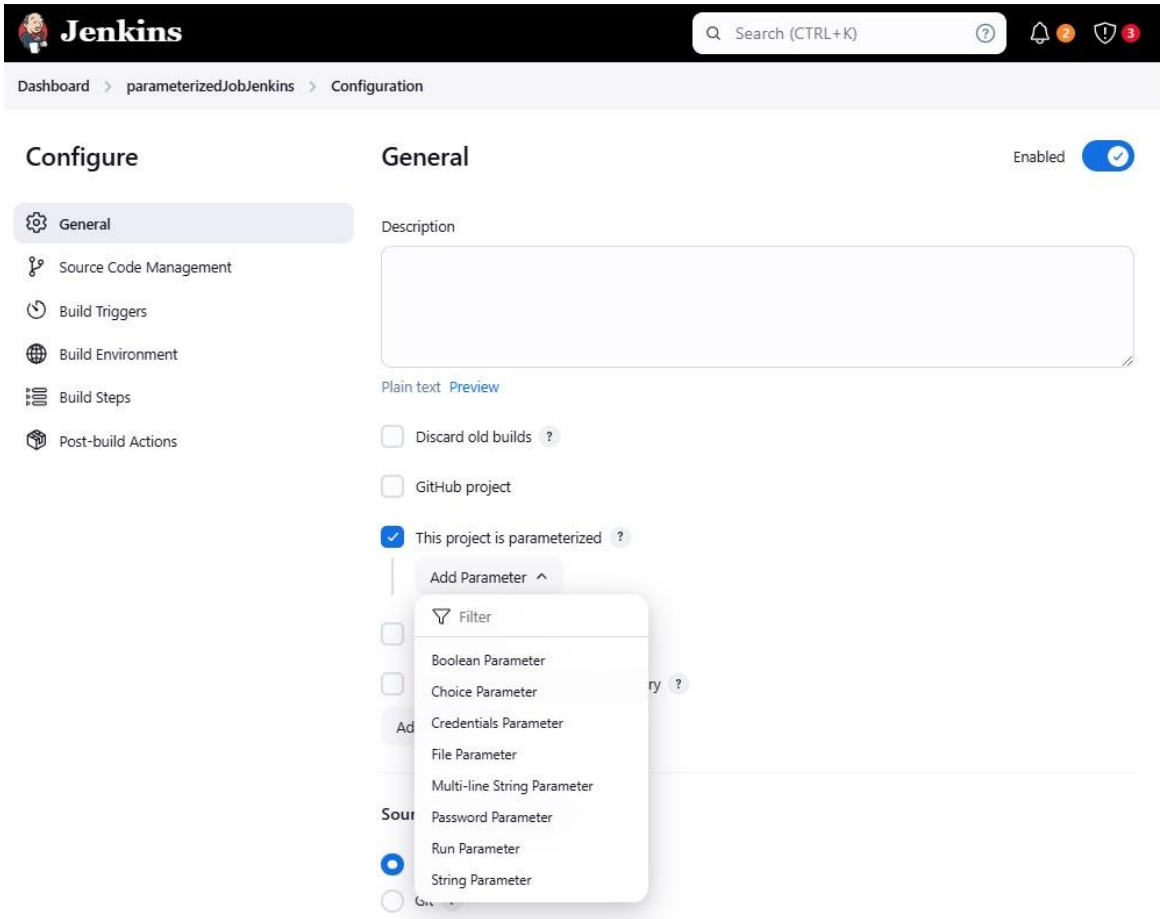
Configure job left Side Section

Step 3: Enable Parameterized Builds

On the configuration page, you will find a section which is General. In that section, there is an option called This project is parameterized that will be Tick marked already like shown in bellow.



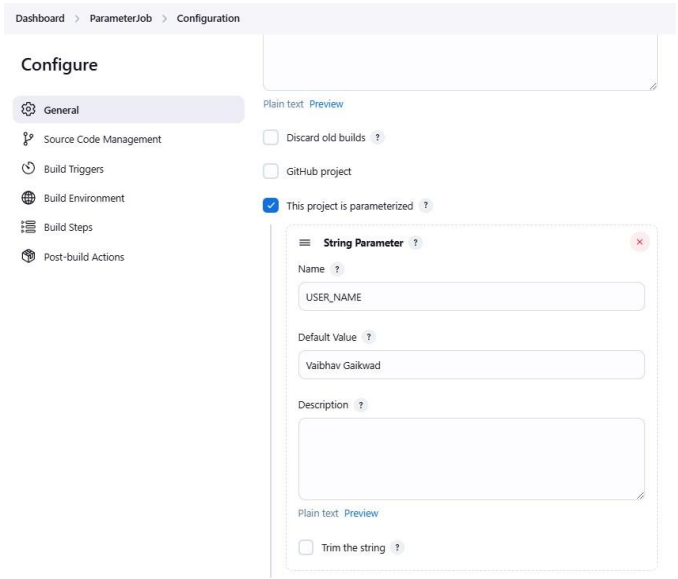
Step 4: Click on the add parameters.



Click on the add parameters in jenkins job

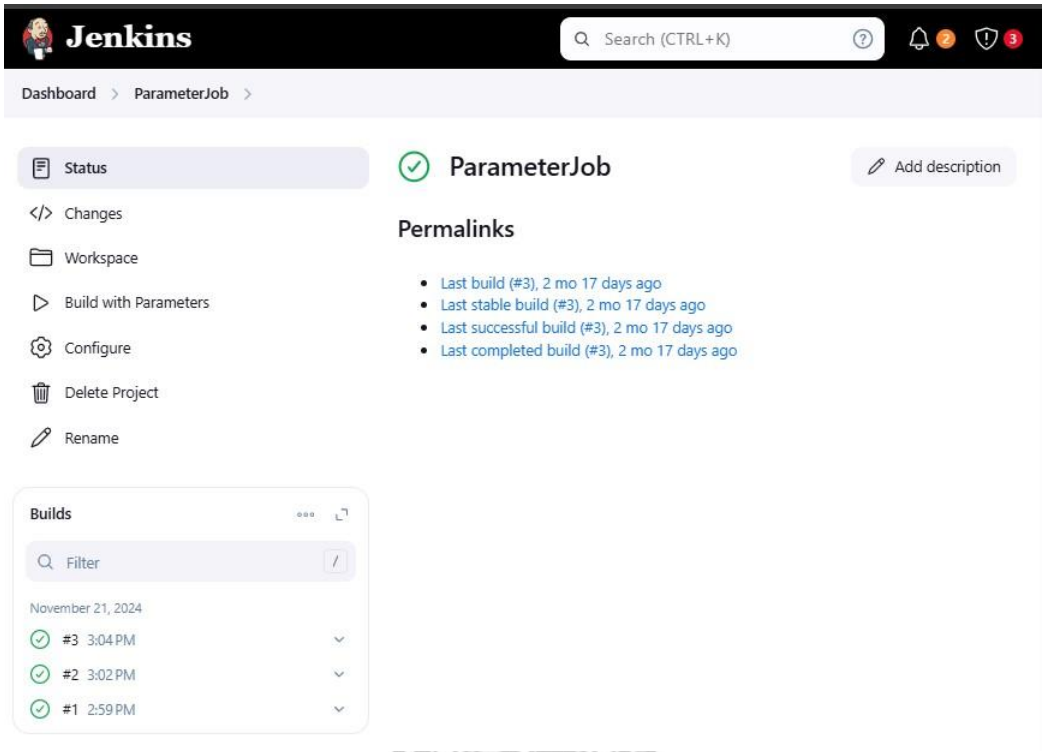
Step 5: Define the Parameter

We can Select the String Parameters and In the Name field, enter the name of the parameters(USER_NAME), and the Default value Select as per your need.



String parameters provided

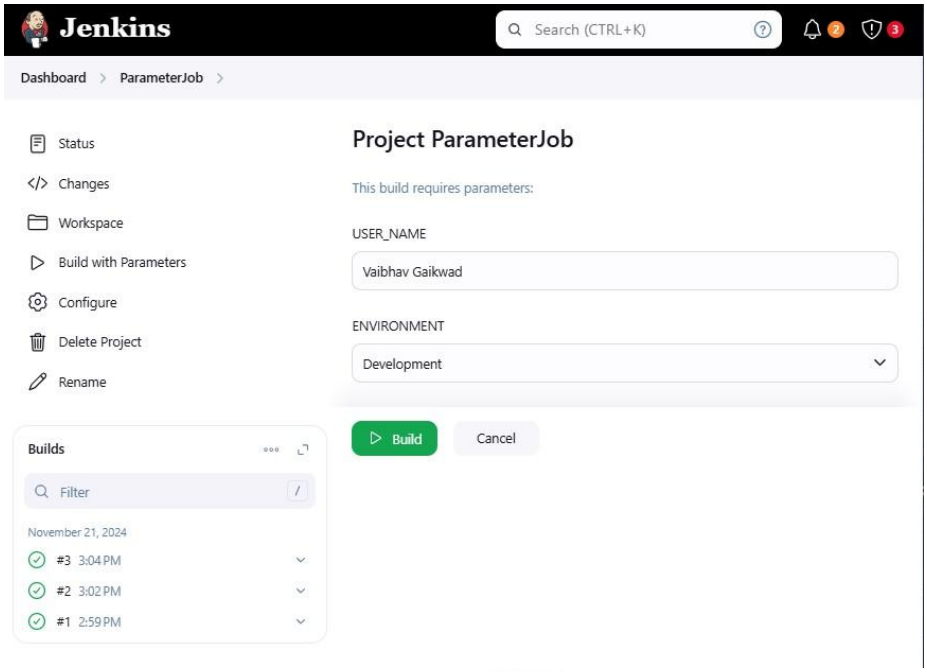
Step 6: Save the Job Configuration



Build With parameters option

Step 7: Then you will see the option which is "Build with Parameters"

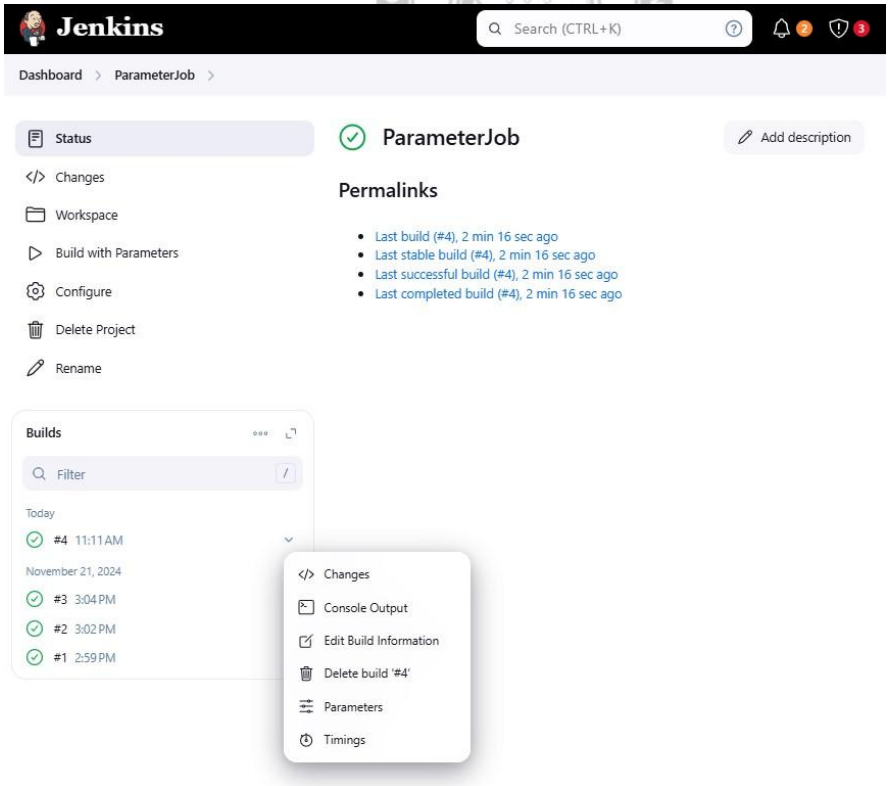
Enter the desired value for each parameter. Once you have done with filling the required values then click on Build to run the job.



Values provided to the job

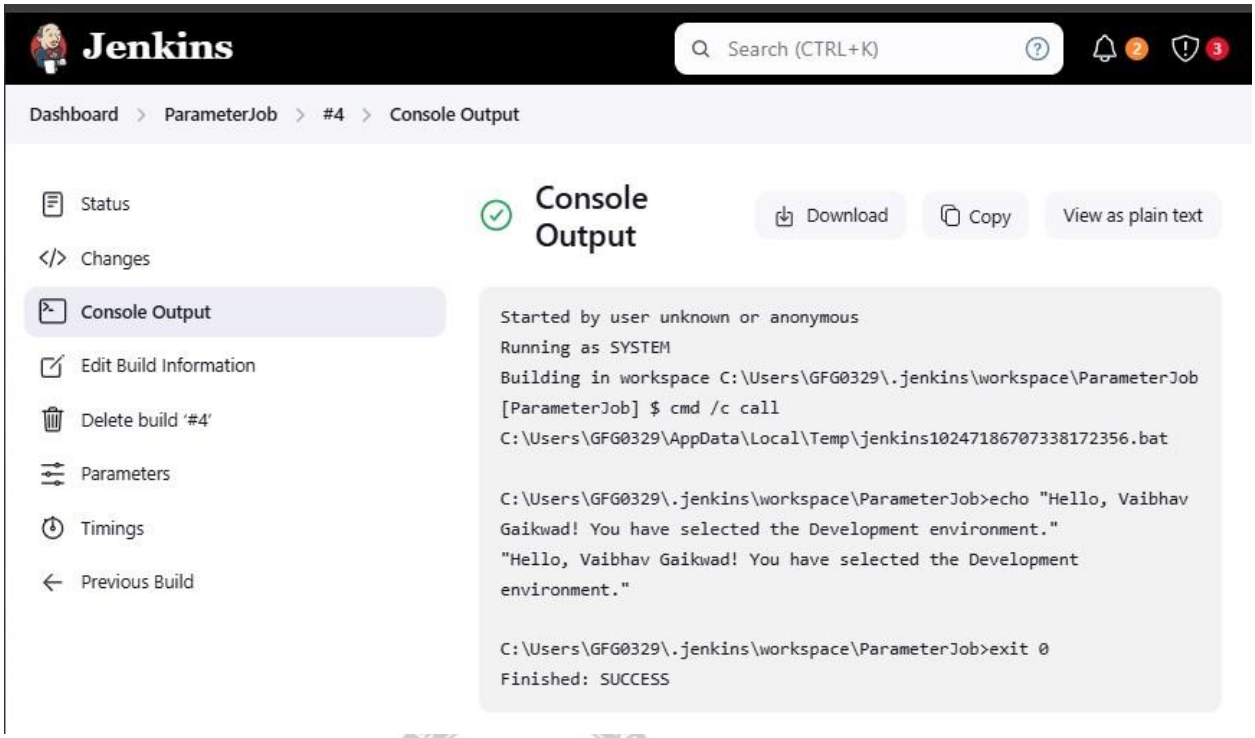
Step 9: Monitor the Job Execution

See the result with clicking the job which are success in the Console output.



job status of parameterized job

Step 10: See the Result in the Console Output tab.


Jenkins
Search (CTRL+K)
Dashboard > ParameterJob > #4 > Console Output
Status
Changes
Console Output
Edit Build Information
Delete build '#4'
Parameters
Timings
Previous Build
Console Output
Download
Copy
View as plain text
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace C:\Users\GFG0329\.jenkins\workspace\ParameterJob
[ParameterJob] $ cmd /c call
C:\Users\GFG0329\AppData\Local\Temp\jenkins10247186707338172356.bat
C:\Users\GFG0329\.jenkins\workspace\ParameterJob>echo "Hello, Vaibhav Gaikwad! You have selected the Development environment."
"Hello, Vaibhav Gaikwad! You have selected the Development environment."
C:\Users\GFG0329\.jenkins\workspace\ParameterJob>exit 0
Finished: SUCCESS