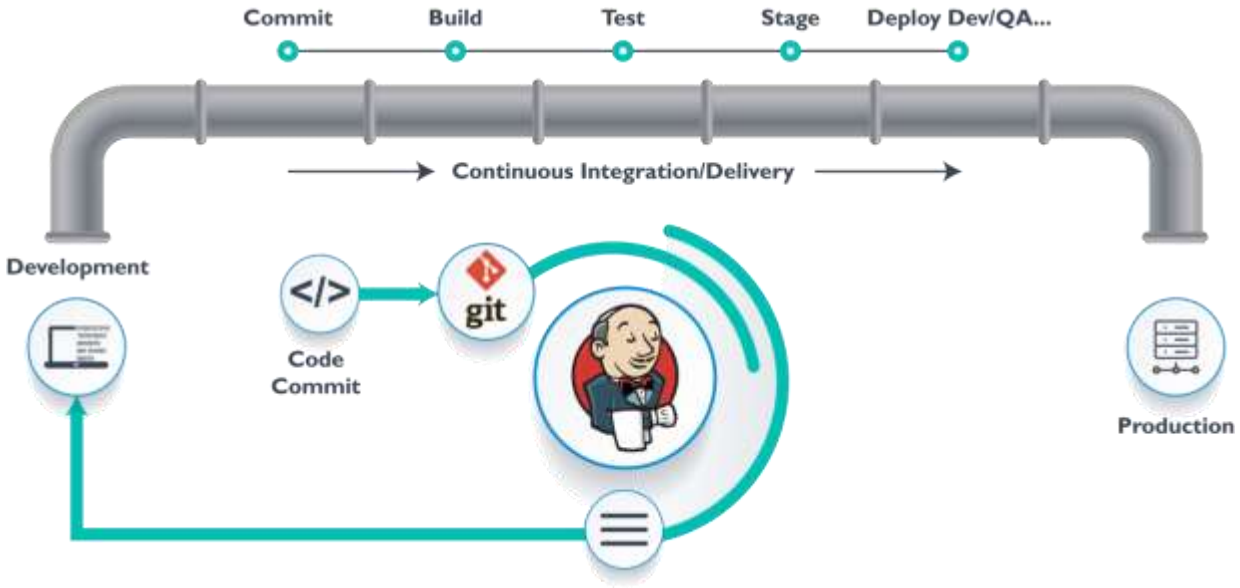
# Jenkins:

Jenkins is open source Continuous Integration tool.




**Continuous delivery** has been most important aspect of the software development life cycle now and with that Jenkins has been always the first choice of tool for the continuous delivery. Why Jenkins has gained so much of popularity and become one of the first row choice tool for the software community, one of the reason is Jenkins Pipeline. Jenkins pipeline is a continuous delivery pipeline that executes the software workflow as code.

## Continuous Integration:

In Continuous Integration after a code commit, the software is built and tested immediately.

In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If deployment is a success, the code is pushed to production. This commit, build, test,

and deploy is a continuous process and hence the name continuous integration/deployment.

A **Continuous Integration Pipeline** is a powerful instrument that consists of a set of tools designed
to **host**, **monitor**, **compile** and **test** code, or code changes, like:

- **Continuous Integration Server** (Jenkins)
- **Source Control Tool** (SVN/GIT)
- **Build tool** (ANT/Maven/Gradle, and others)
- **Automation testing framework** (Selenium)

# Jenkins:

**Jenkins** is an open source Continuous Integration server capable of orchestrating a chain of actions that help to achieve the Continuous Integration process (and not only) in an automated fashion.

Jenkins is entirely written in Java.

It is a server-based application and requires a web server like Apache Tomcat. The reason Jenkins became so popular is that of its monitoring of repeated tasks which arise during the development of a project. For example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.

By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate build and test at a rapid rate.

Jenkins supports the complete development lifecycle of software from building, testing, documenting the software, deploying and other stages of a software development lifecycle.

**After Jenkins:**

The code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day

If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team.

If the build fails, then Jenkins will notify the errors to the developer team.

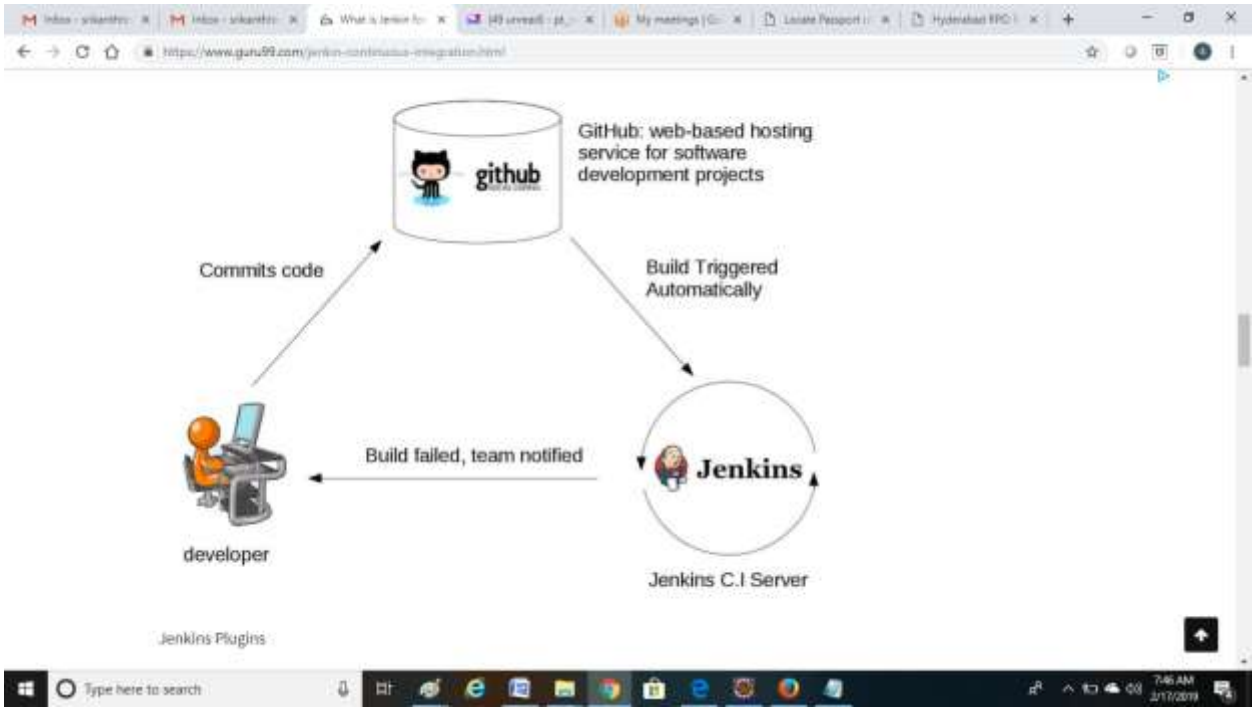
The code is built immediately after any of the Developer commits.

Since the code is built after each commit of a single developer, it's easy to detect whose code caused the built to fail

Automated build and test process saving timing and reducing defects.

The code is deployed after every successful build and test.

The development cycle is fast. New features are more readily available to users. Increases profits.




**Jenkins Plugins**

By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git. For integration with tools like Maven you need to install respective plugins in your Jenkins.

# Important Points:

- In Continuous Integration, after a code commit, the software is built and tested immediately
- Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions
- Before Jenkins when all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.

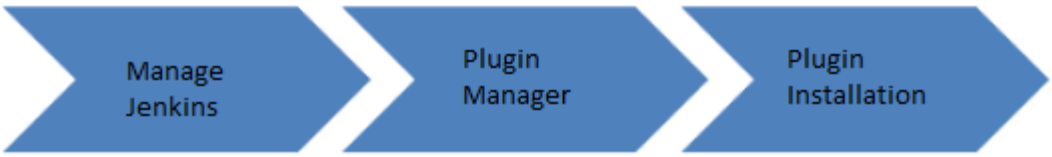
- After Jenkins the code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day
- By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git
- The biggest pros of Jenkins is that it is managed by the community which holds public meetings and take inputs from the public for the development of Jenkins projects
- The biggest con of Jenkin is that Its interface is out dated and not user friendly compared to current UI trends.

## Software Requirements for Jenkins:

- Since Jenkins runs on Java, you need either Java Development Kit (JDK) or Java Runtime Environment (JRE).

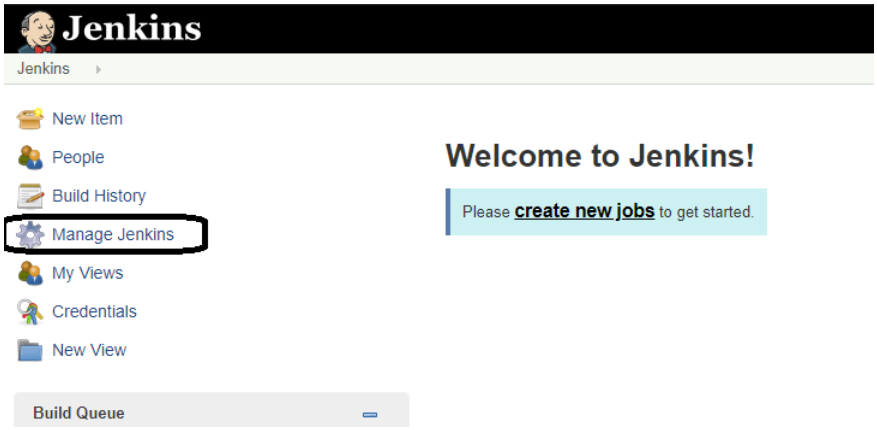
## Jenkins GitHub Integration: Install Git Plugin:

https://www.jenkins.io/download/

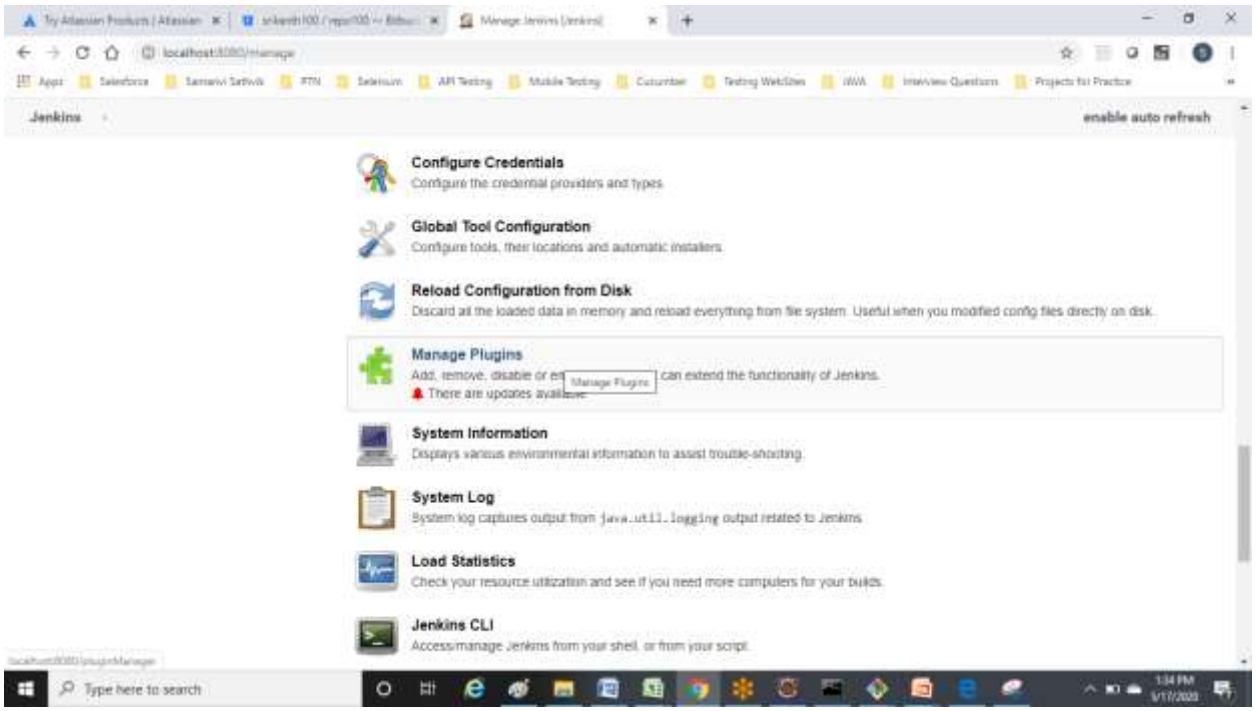


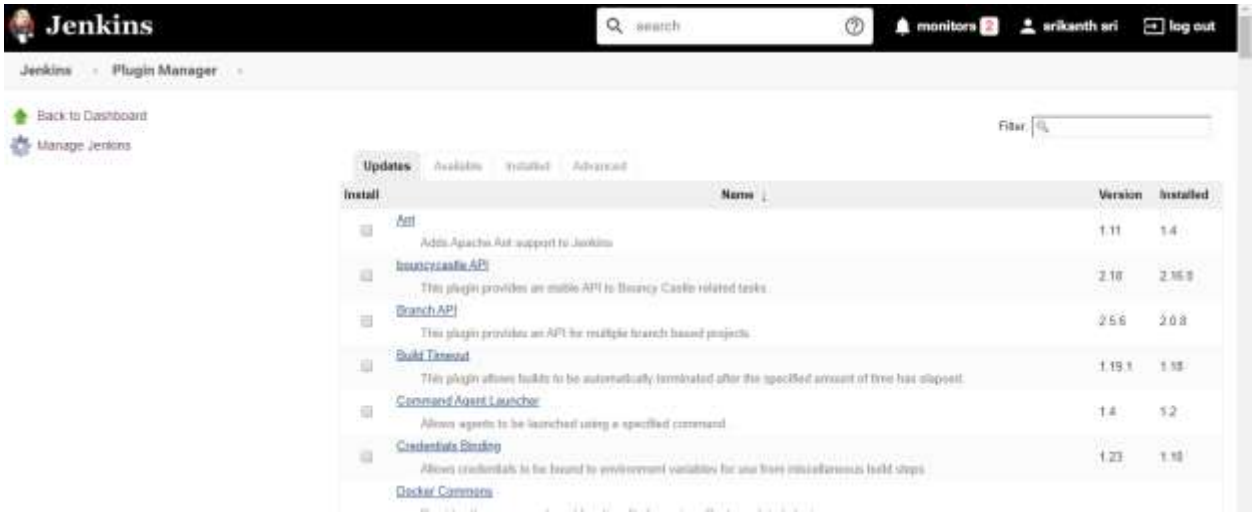

# Install GIT Plugin

**Step 1:** Click on the **Manage Jenkins** button on your Jenkins dashboard:




**Step 2:** Click on **Manage Plugins**:

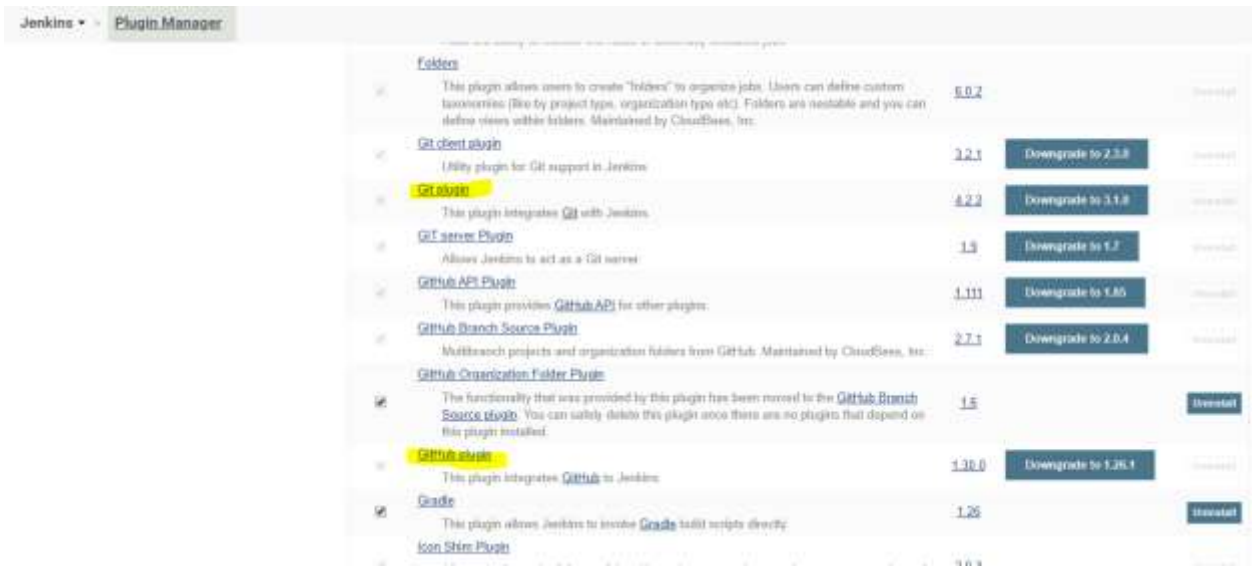




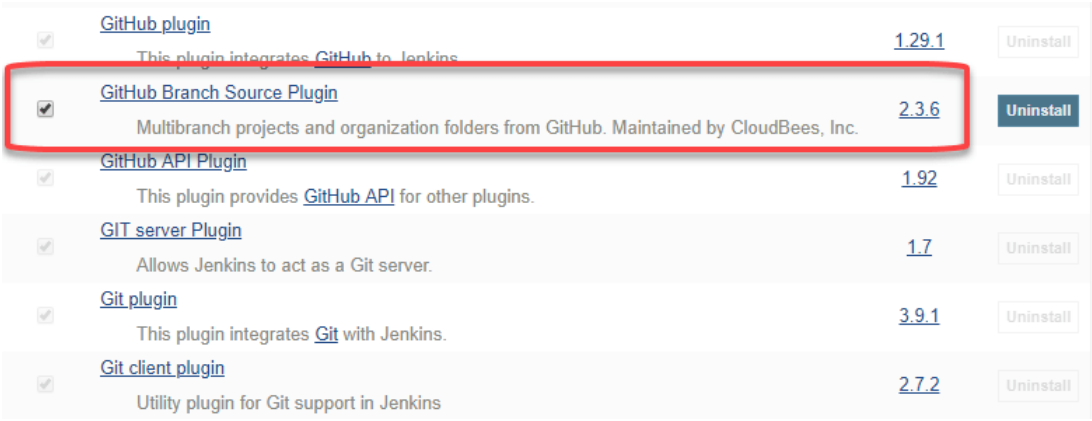

Look  for GIT plugins. Check the check boxes.

**Step 3:** In the Plugins Page

1. Select the GIT Plugin
2. Click on **Install without restart.** The plugin will take a few moments to finish downloading depending on your internet connection, and will be installed automatically.
3. You can also select the option **Download now and Install after restart** button. In which plugin is installed after restart
4. You will be shown a "No updates available" message if you already have the Git plugin installed.



**Step 4:** Once the plugins have been installed, go to **Manage Jenkins** on your Jenkins dashboard. You will see your plugins listed among the rest.

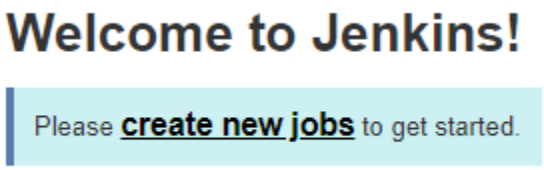



# Integrating Jenkins with GitHub

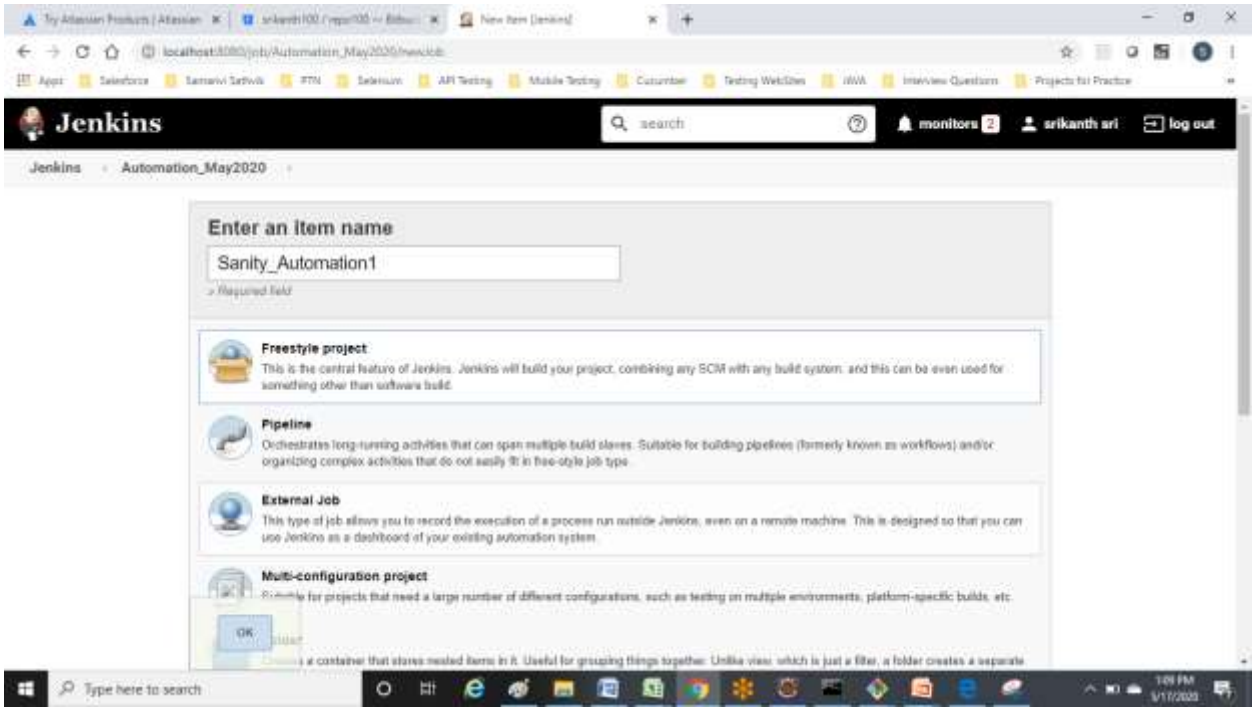
We shall now discuss the process of integrating GitHub into Jenkins in a Windows system.

**Step 1)** Create a new job in Jenkins, open the Jenkins dashboard with your Jenkins URL. For example, http://localhost:8080/

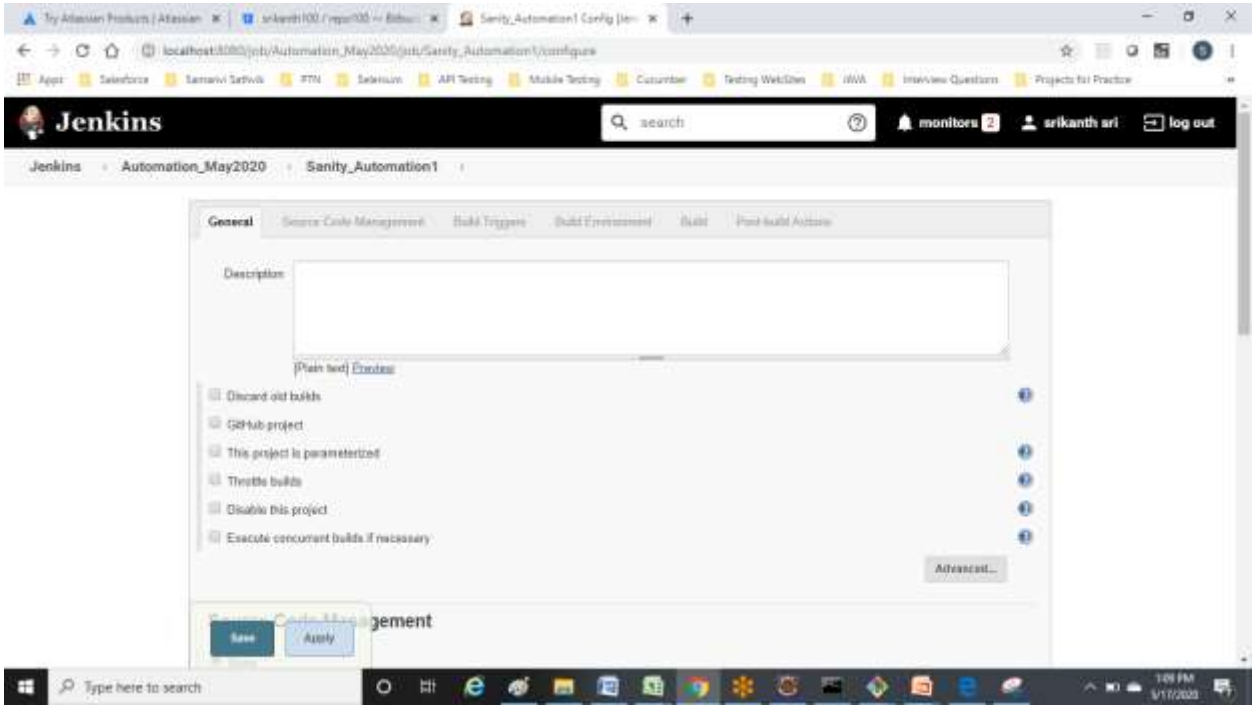
Click on **create new jobs**:



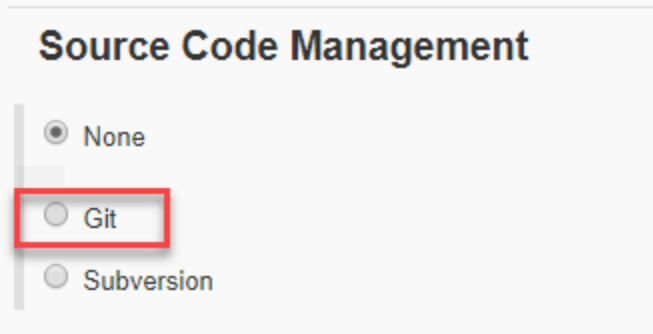

**Step 2)** Enter the item name, select job type and click **OK**. We shall create a Freestyle project as an example.




**Step 3)** Once you click **OK,** the page will be redirected to its project form. Here you will need to enter the project information:



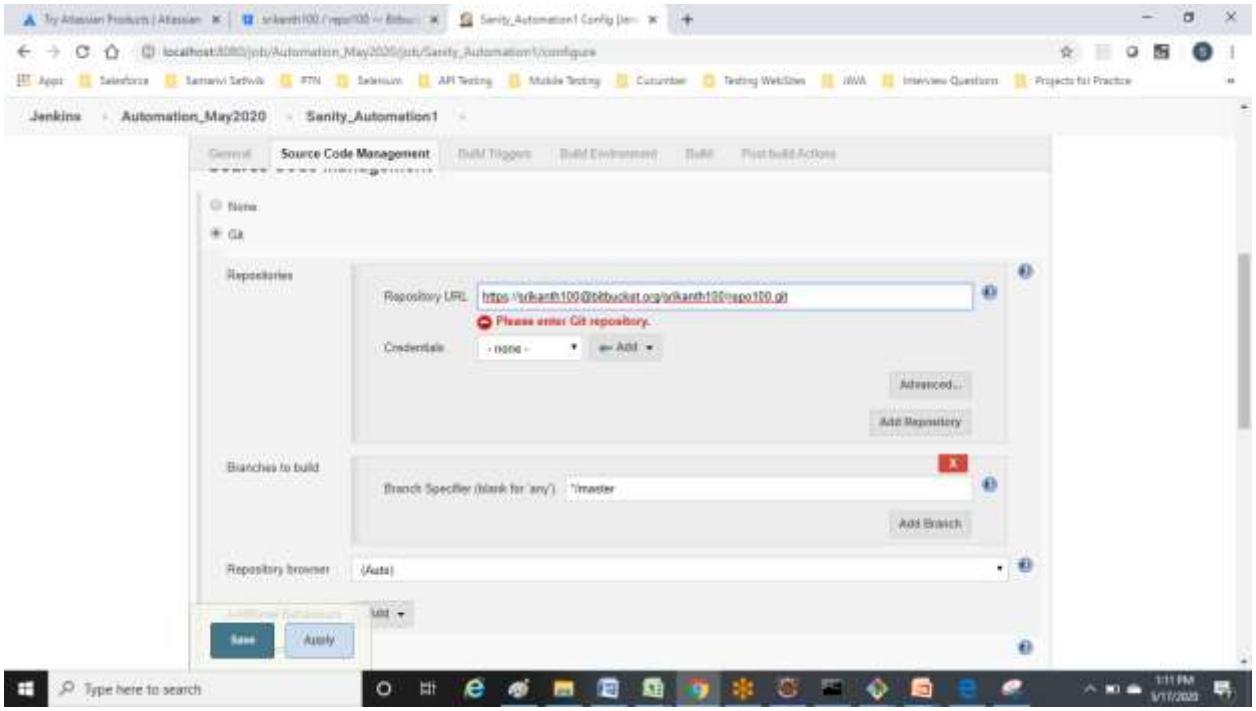

**Step 4)** You will see a **Git** option under **Source Code Management** if your Git plugin has been installed in Jenkins:




NOTE: If the **Git** option does not appear, try re-installing the plugins, followed by a restart and a re-login into your Jenkins dashboard. You will now be able to see the **Git** option as mentioned above.

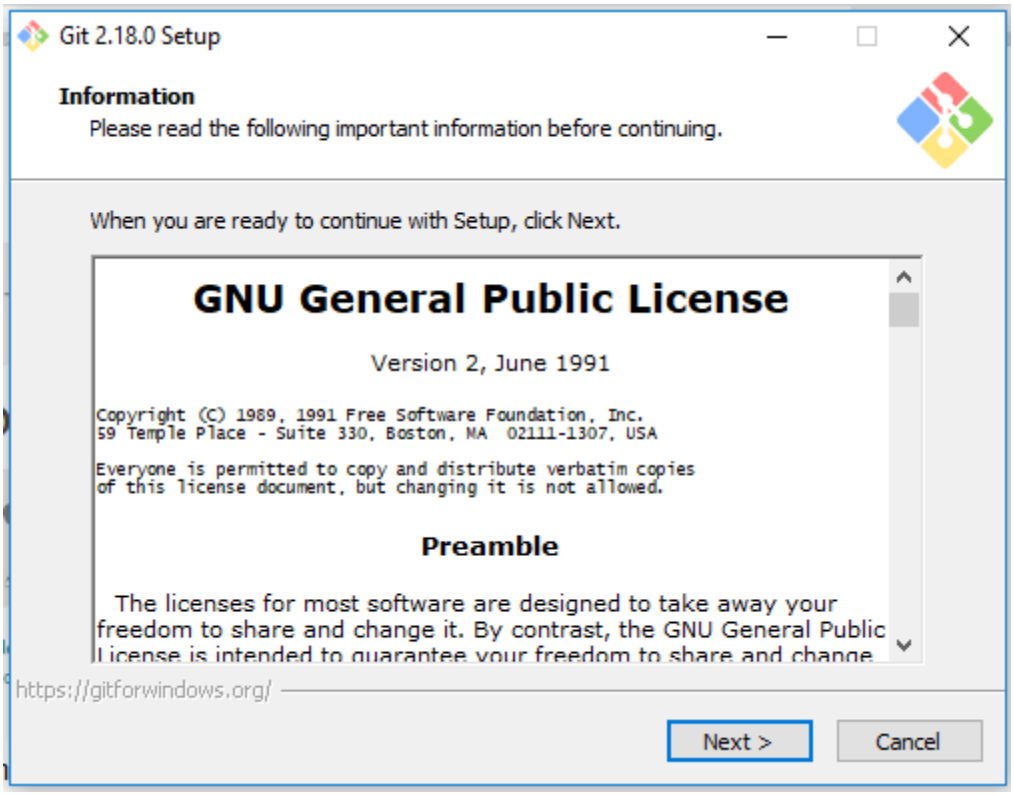
**Step 5)** Enter the Git repository URL to pull the code from GitHub.


**Step 6)** You might get an error message the first time you enter the repository URL. For example:




This happens if you do not have Git installed in your local machine. To install Git in your local machine, go to [https://git-scm.com/downloads](https://git-scm.com/downloads)

Download the appropriate Git file for your Operating System, in this case, Windows, and install it onto your local machine running Jenkins. Complete the onscreen instructions to install GIT.




**Step 6)** You can execute Git repositories in your Jenkins once Git has been installed on your machine. To check if it has been successfully installed onto your system, open your **command prompt,** type "Git" and press enter. You should see different options come up for Git:

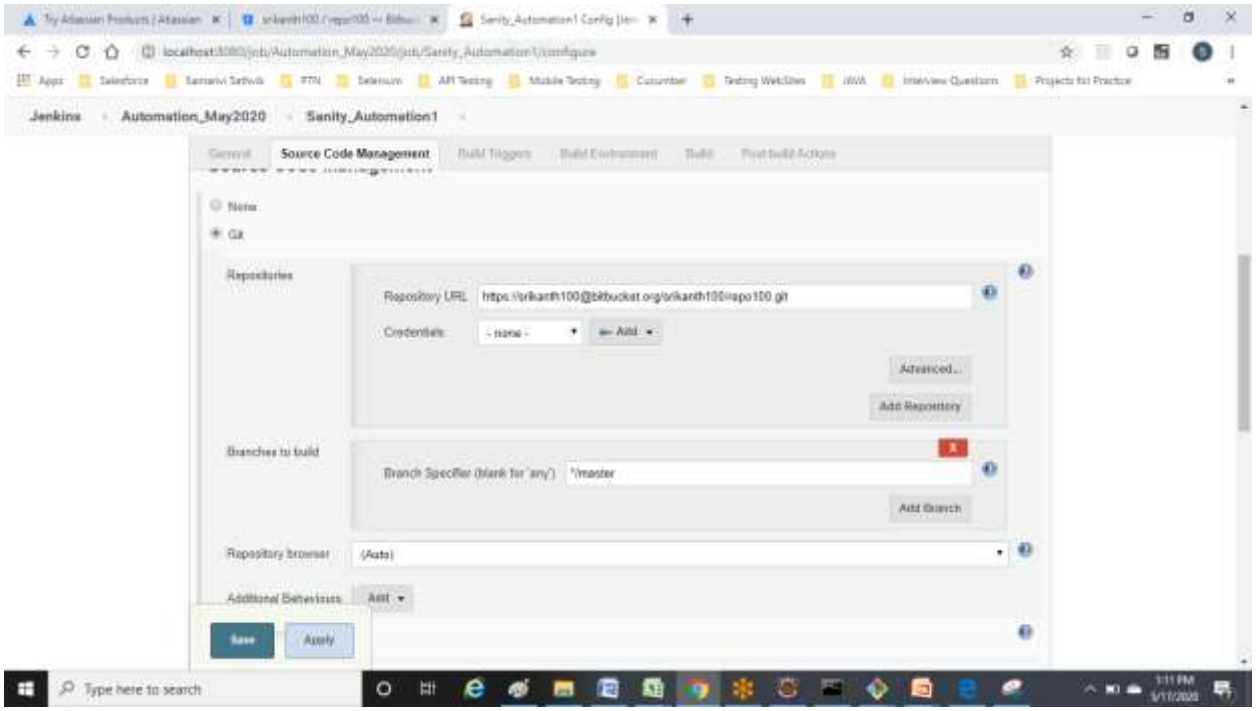
```
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\alex>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:
```

This means that Git has been installed in your system.

**Step 7)** Once you have everything in place, try adding the Git URL into Jenkins. You will not see any error messages:

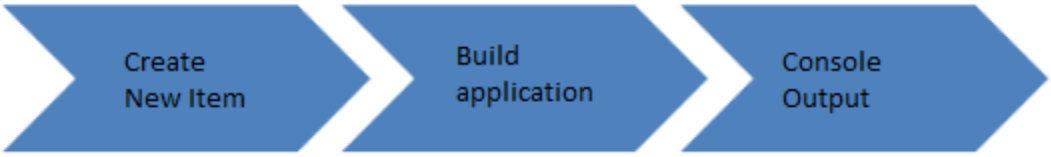



Git is now properly configured on your system.

**How to Create Builds with the Jenkins Freestyle Project:**

# What is a Jenkins freestyle project?

A Jenkins project is a repeatable build job which contains steps and post-build actions.



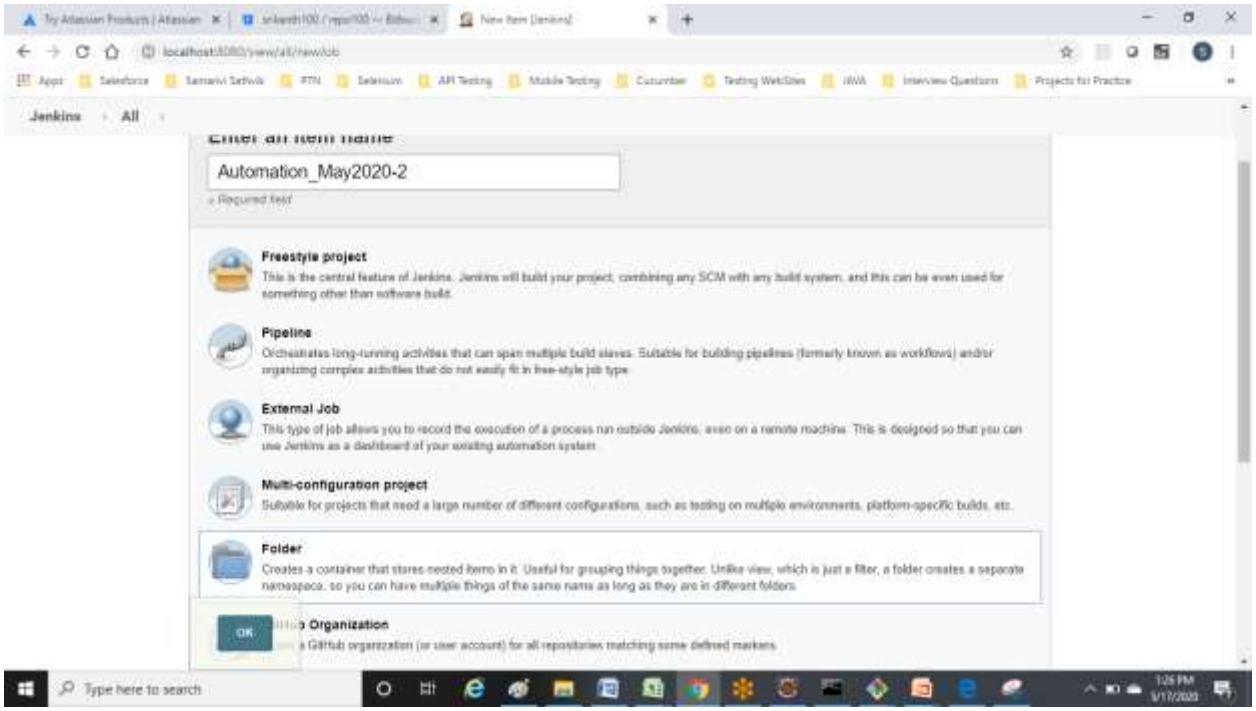

# Creating a Freestyle Build Job

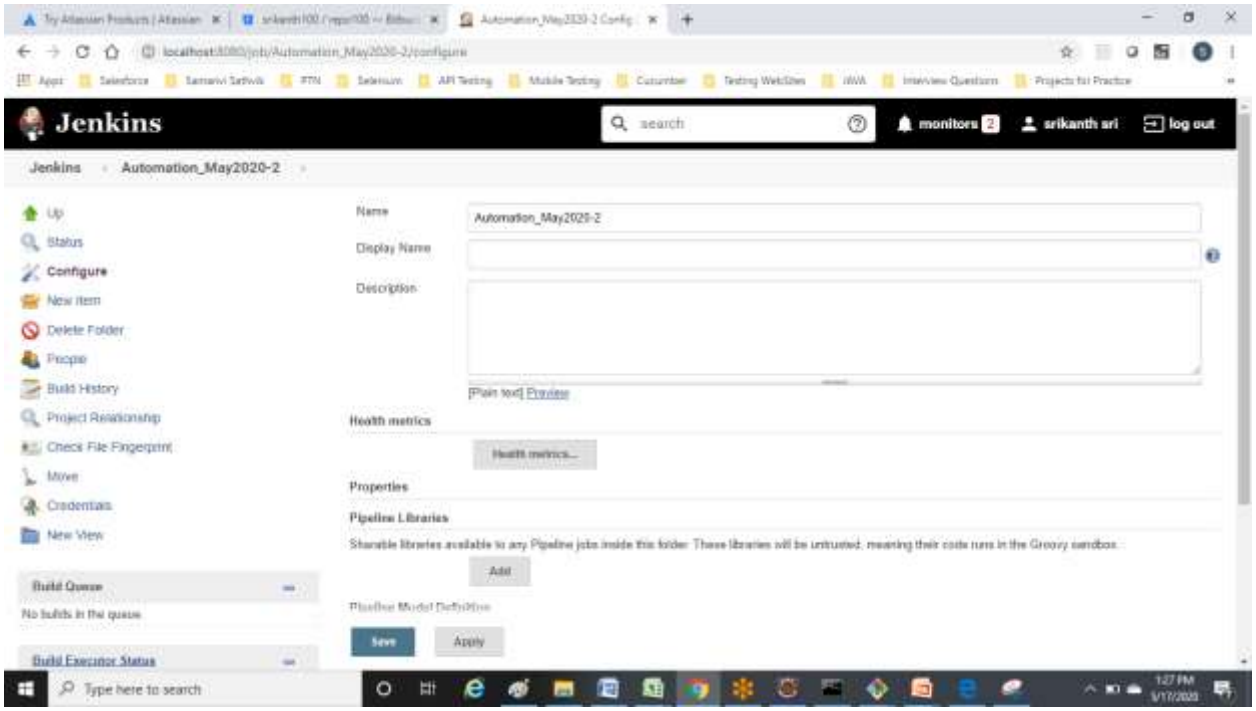
The freestyle build job is a highly flexible and easy-to-use option. You can use it for any type of project; it is easy to set up, and many of its options appear in other build jobs.

**Step 1)** To create a Jenkins freestyle job, log on to your Jenkins dashboard by visiting your Jenkins installation path. Usually, it will be hosted on localhost at http://localhost:8080 If you have installed Jenkins in another path, use the appropriate URL to access your dashboard.

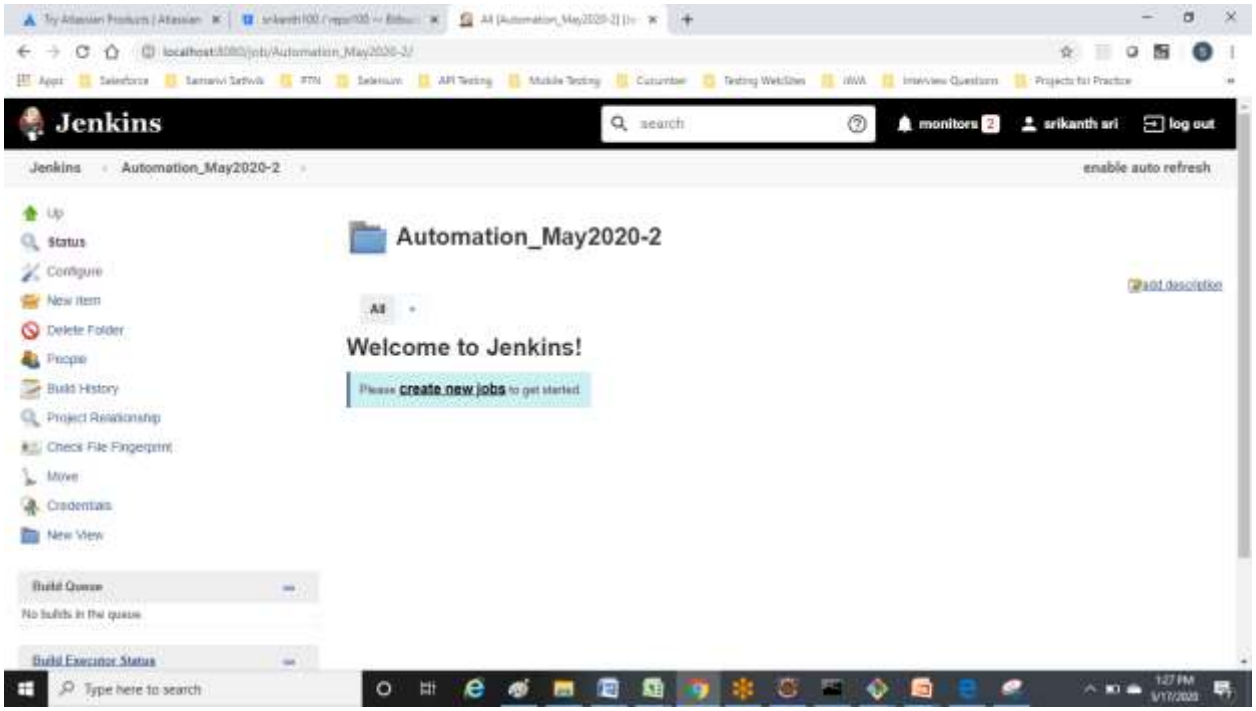

**Step 2)** Click on "**New Item**" at the top left-hand side of your dashboard.


Create one folder



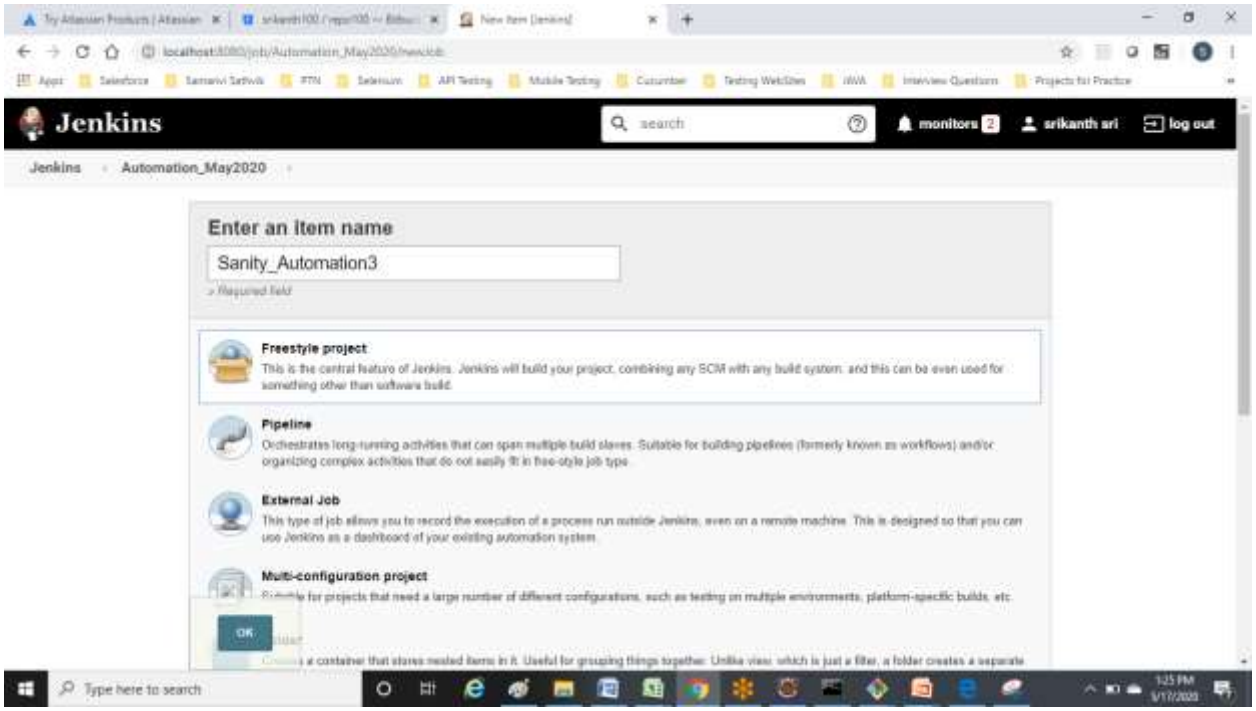

Folder is created



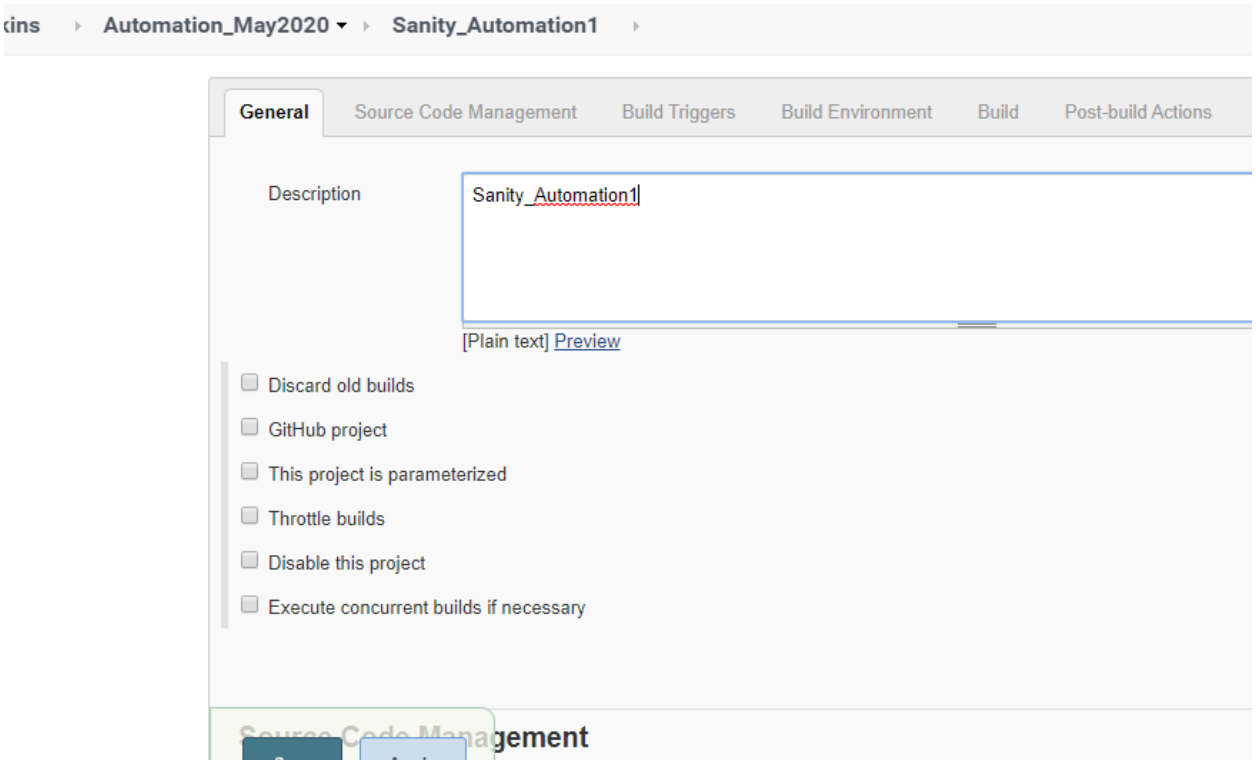

Click on “create new jobs”

**Step 3)** In the next screen,

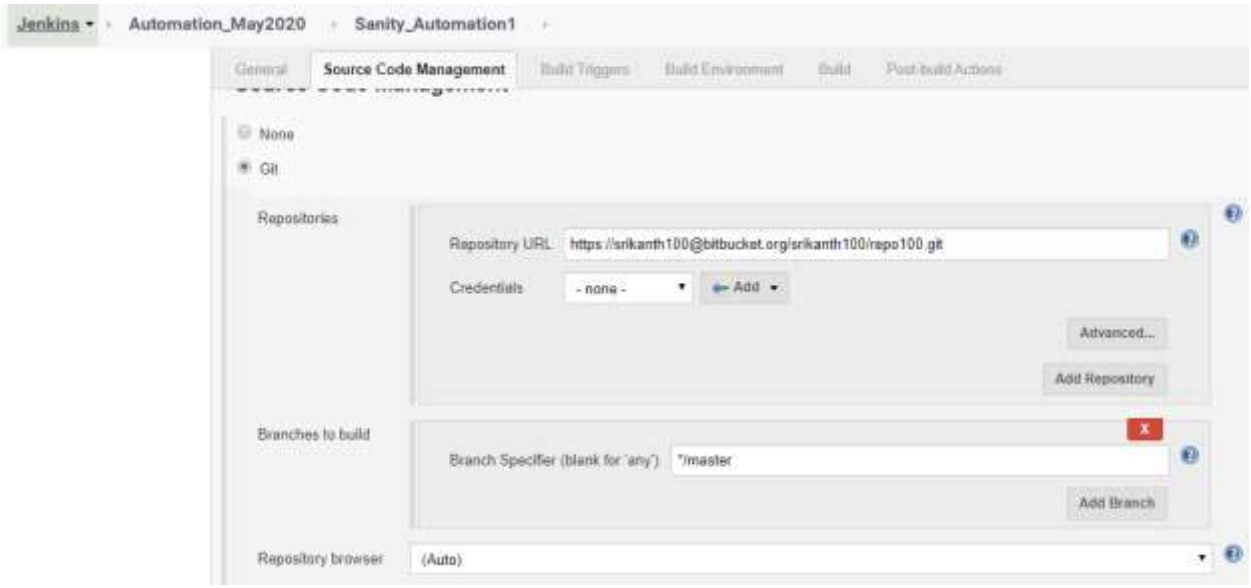
1. Enter the name of the job you want to create. I am using " Sanity_Automation3" for this demo.
2. Select Freestyle project
3. Click Okay




**Step 4)** Enter the details of the project you want to test.



**Step 5)** Under Source Code Management, Enter your repository URL. We have a test repository located at https://srikanth100@bitbucket.org/teamqadec2019/repo2.git




It is also possible for you to use a local repository.

If your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then pull the source code from your GitHub repository.

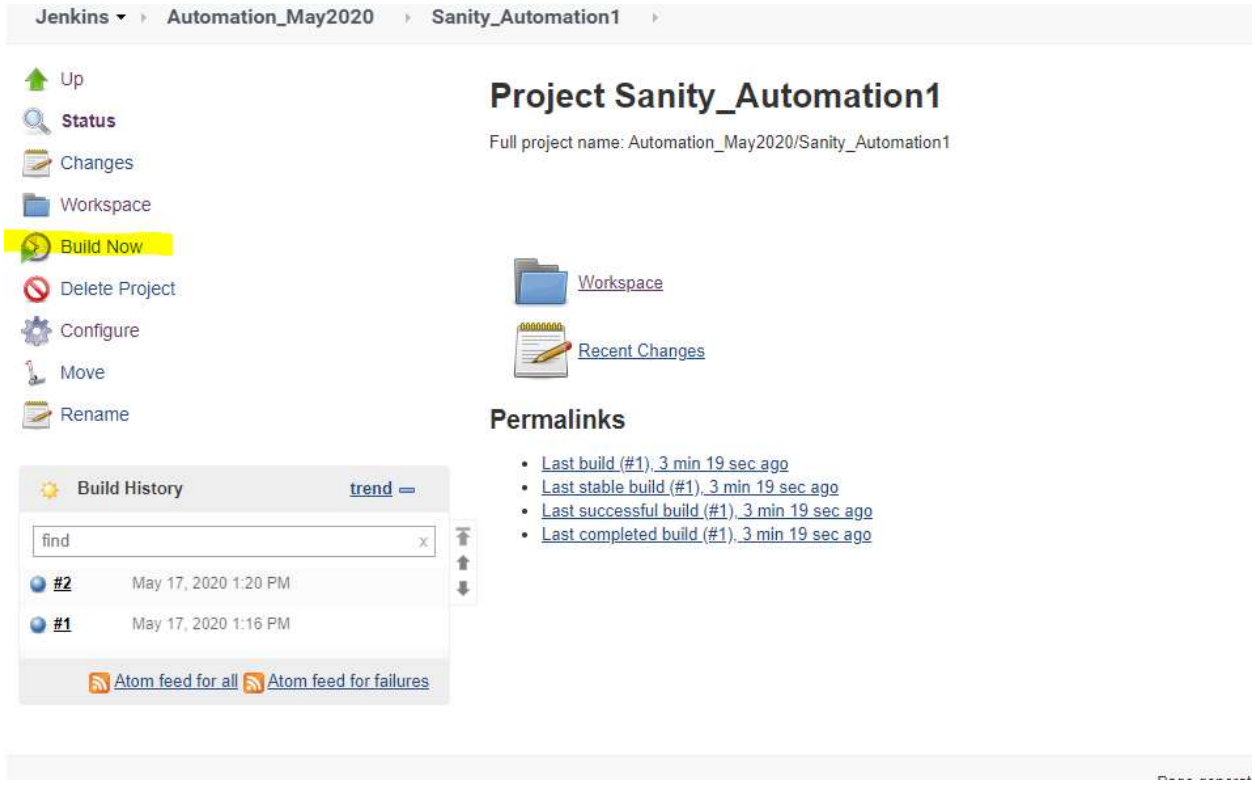
**Step 6)** Below step (Build Triggers) is optional.

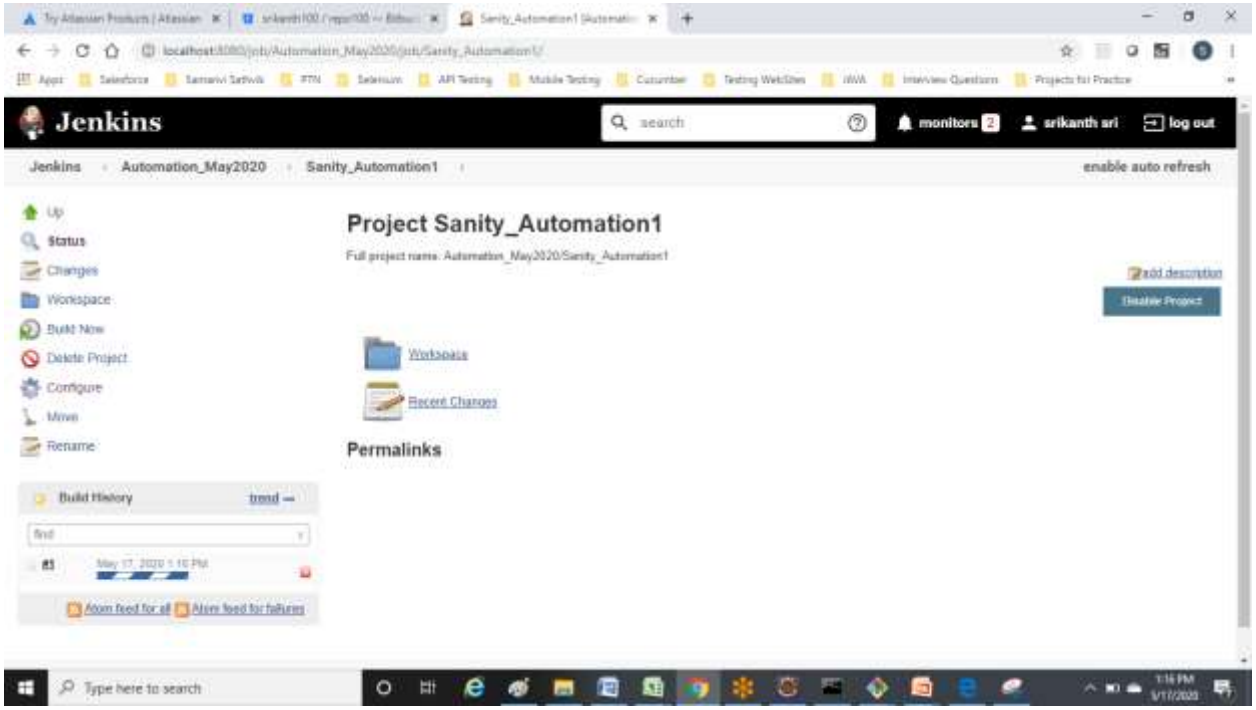

**Step 7) Select Post-build Actions**


When you have entered all the data,

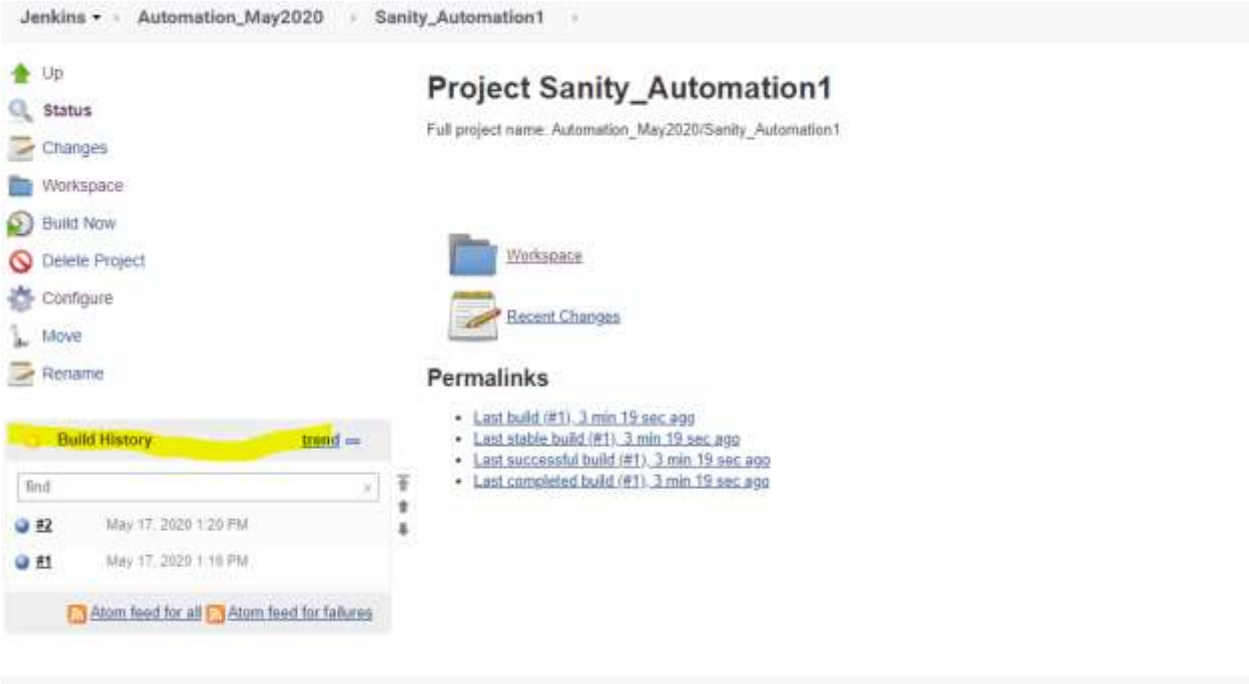
1. Click **Apply**
2. **Save** the project.

**Step 8)** Now, in the main screen, Click the **Build Now** button on the left-hand side to build the source code.


Jenkins
Automation_May2020
Sanity_Automation1
Up
Status
Changes
Workspace
Build Now
Delete Project
Configure
Move
Rename
Build History
trend
find
#2 May 17, 2020 1:20 PM
#1 May 17, 2020 1:16 PM
Atom feed for all
Atom feed for failures
Project Sanity_Automation1
Full project name: Automation_May2020/Sanity_Automation1
Workspace
Recent Changes
Permalinks
Last build (#1), 3 min 19 sec ago
Last stable build (#1), 3 min 19 sec ago
Last successful build (#1), 3 min 19 sec ago
Last completed build (#1), 3 min 19 sec ago

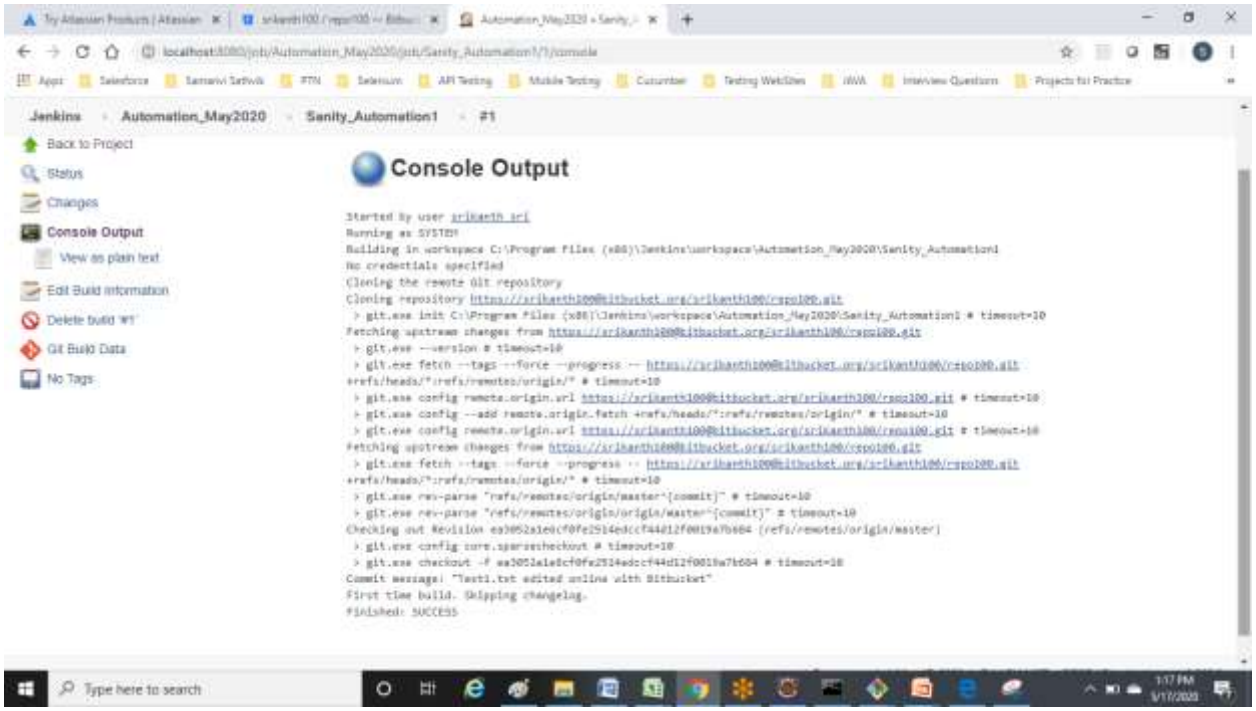

Jenkins
search
monitors
srikanth sri
log out
Jenkins
Automation_May2020
Sanity_Automation1
enable auto refresh
Project Sanity_Automation1
Permalinks
Type here to search

**Step 9)** After clicking on **Build now,** you can see the status of the build you run under **Build History**.

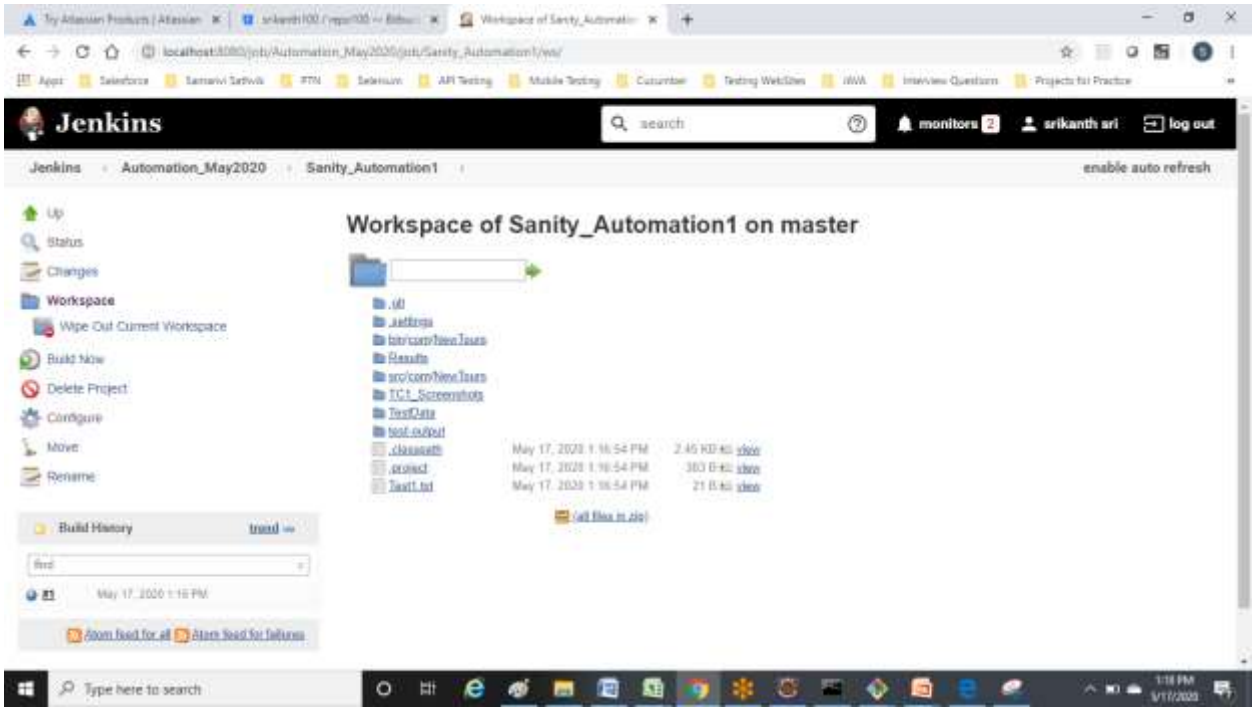



**Step 10)** Click on the **build number** and then Click on **console output** to see the status of the build you run. It should show you a success message, provided you have followed the setup properly.

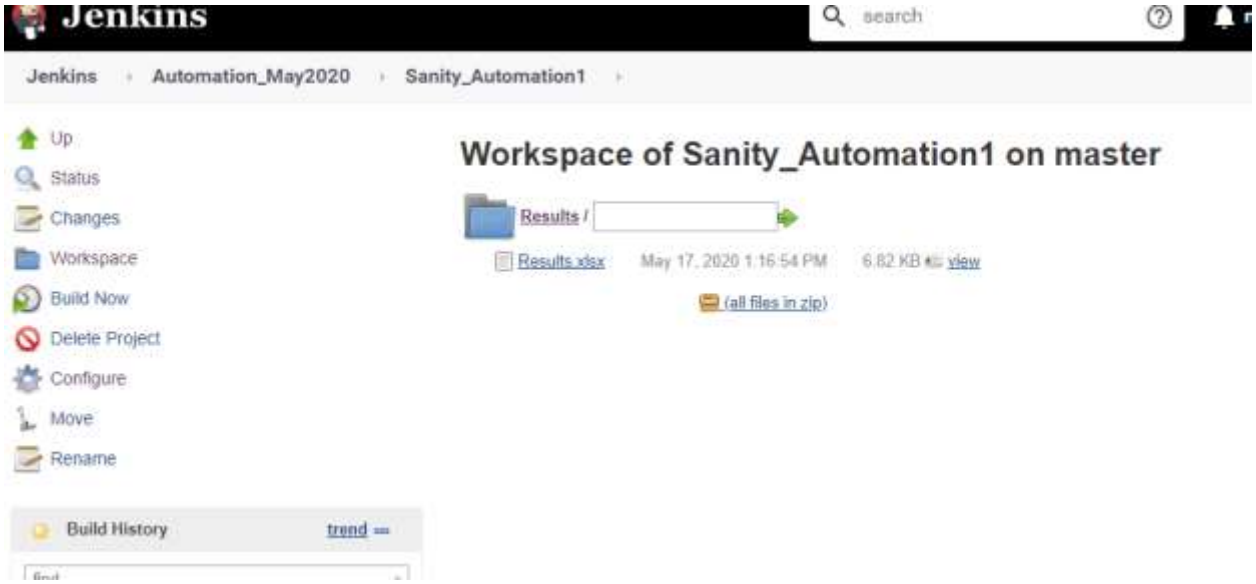
Console log can be seen at,




Workspace can be accessed in Jenkins at




**Result file:**

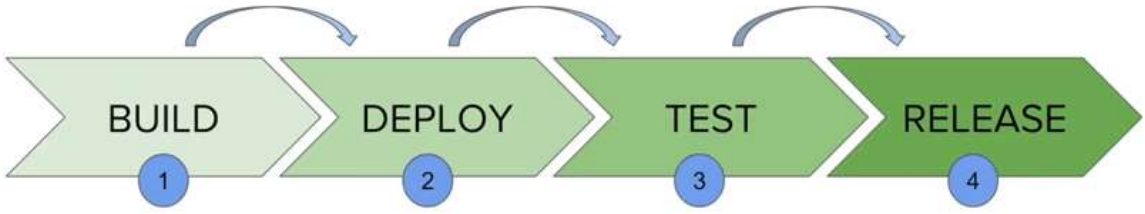



# What is Jenkins Pipeline?

In Jenkins, a pipeline is a group of events or jobs which are interlinked with one another in a sequence.

## What is Continuous Delivery Pipelines? How it Works?

In a Jenkins pipeline, every job or event has some sort of dependency on at least one or more events.




http://localhost:8080/job/Automation_May2020/job/Test/

The picture above represents a continuous delivery pipeline in Jenkins. It contains a group of states called build, deploy, test and release. These events are interlinked with each other. Every state has its events, which work in a sequence called a continuous delivery pipeline.

# What is a JenkinsFile?

Jenkins pipelines can be defined using a text file called **JenkinsFile.** You can implement pipeline as code using JenkinsFile, and this can be defined by using a domain specific language (DSL). With JenkinsFile, you can write the steps needed for running a Jenkins pipeline.

The benefits of using J**enkinsFile are**:

- You can create pipelines automatically for all branches and execute pull requests with just one **JenkinsFile.**
- You can review your code on the pipeline
- You can audit your Jenkins pipeline
- This is the singular source for your pipeline and can be modified by multiple users.

# Jenkin's Pipeline:

Jenkins is an open continuous integration server which has the ability to support the automation of software development processes. You can create multiple automation jobs with the help of use cases, and run them as a Jenkins pipeline.

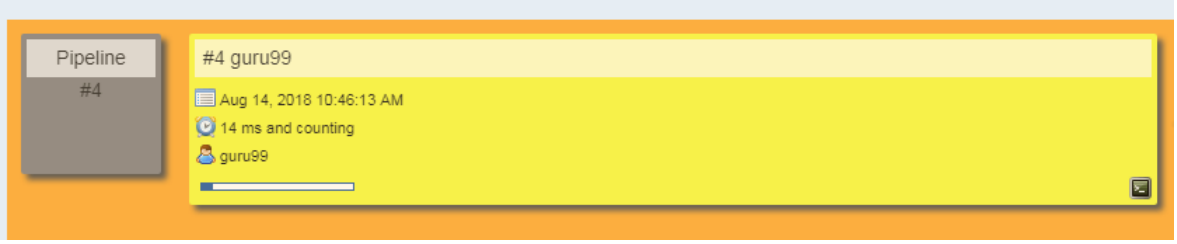
Here are the reasons why you use should use Jenkins pipeline:

- Jenkins pipeline is implemented as a code which allows multiple users to edit and execute the pipeline process.

- Pipelines are robust. So if your server undergoes an unforeseen restart, the pipeline will be automatically resumed.
- You can pause the pipeline process and make it wait to resume until there is an input from the user.
- Jenkins Pipelines support big projects. You can run multiple jobs, and even use pipelines in a loop.

# Running Jenkins pipeline

Click on **Run** to run the Jenkins pipeline. It will look something like this:




In the example above, we are demonstrating a simple "helloworld.java" program. But in real time projects, you will be responsible for creating and building complex pipelines in Jenkins. 0See below for a sample pipeline view.


build
test: integration-&-quality
test: functional
test: load-&-security
approval
deploy: prod
Average stage times:
(Average full run time: ~4s)
836ms
20min 43s
9ms
7ms
89ms
5ms
#17
Sep 22
15.05
No Changes
Retry
Download
538ms
10s
10ms
8ms
72ms
4ms
#16
Sep 22
15.04
No Changes
Retry
Download
479ms
6s
9ms
9ms
74ms
5ms
#15
Sep 22
15.03
No Changes
Retry
Download
922ms
6s
10ms
9ms
failed
#14
Sep 22
15.03
No Changes
Retry
Download
1s
8s
12ms
9ms
80ms
5ms
#13
Sep 22
15.02
No Changes
Download
942ms
9s
13ms
failed
#12
Sep 22
15.02
No Changes
Retry
Download
1s
6s
13ms
11ms
111ms
aborted