

# Create Builds with Jenkins Freestyle Project

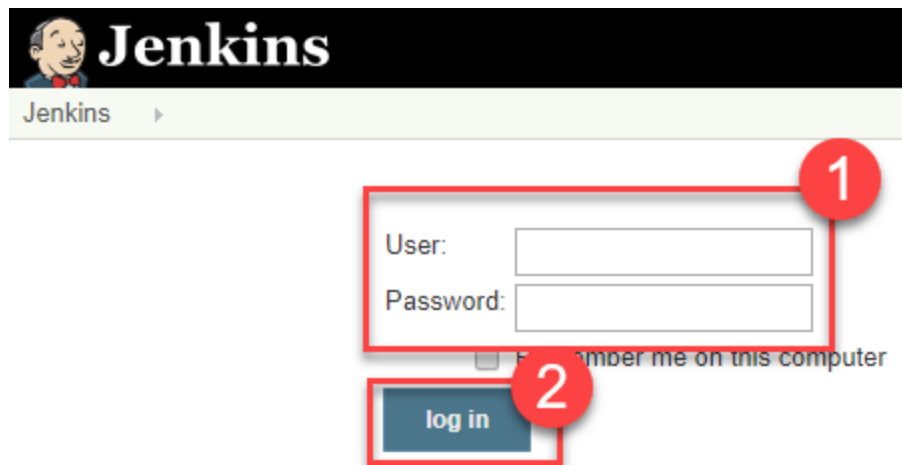
## Jenkins freestyle project

A Jenkins project is a repeatable build job which contains steps and post-build actions. The types of actions you can perform in a build step or post-build action are quite limited. There are many standard plugins available within a Jenkins freestyle project to help you overcome this problem. They allow you to configure build triggers and offers project-based security for your Jenkins project.

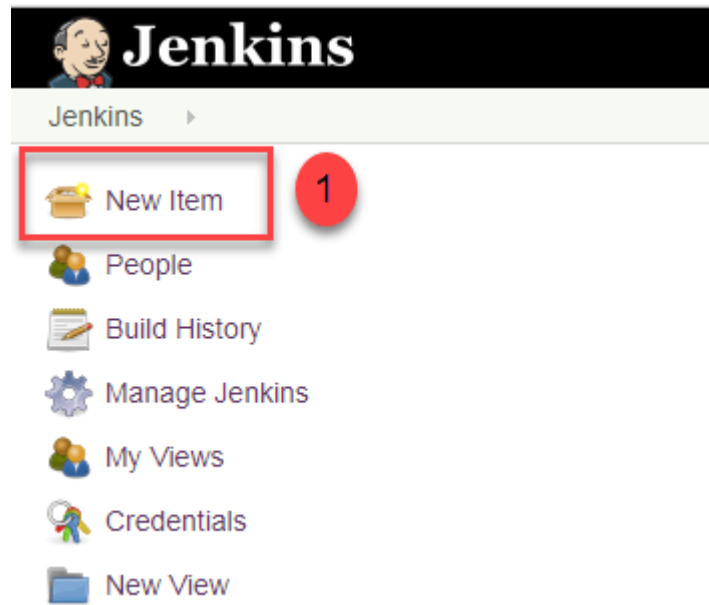
## Creating a Freestyle Build Job

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.



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


1. Enter the name of the item you want to create. We shall use the "Hello world" for this demo.
2. Select Freestyle project
3. Click Okay

**Enter an item name**


Hello World **1**

Required field


**Freestyle project** **2**

 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build tool used for something other than software build.


**Pipeline**

 Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines and/or organizing complex activities that do not easily fit in free-style job type.


**Multi-configuration project**

 Suitable for projects that need a large number of different configurations, such as testing on multiple builds, etc.


**Folder**

 Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, separate namespace, so you can have multiple things of the same name as long as they are in different namespaces.

**GitHub Organization**

 Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**

 Creates a set of Pipeline projects according to detected branches in one SCM repository.

**OK** **3**

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

**Step 5)** Under Source Code Management, Enter your repository URL. Assume We have a test repository located at <https://github.com/lrepo/firstJava.git>

The screenshot shows the Jenkins 'Source Code Management' configuration page. At the top, there are tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Source Code Management' tab is active. Below the tabs, there are two radio buttons: 'None' and 'Git'. The 'Git' radio button is selected and highlighted with a red box. Below the radio buttons, there is a section for 'Repositories'. In this section, the 'Repository URL' field is highlighted with a red box and contains the text 'https://github.com/repo/firstJava.git'. Below the 'Repository URL' field, there is a 'Credentials' dropdown menu set to '- none -' with an 'Add' button next to it. To the right of the 'Add' button is an 'Advanced...' button. Below the 'Repositories' section, there is a section for 'Branches to build'. In this section, the 'Branch Specifier (blank for \'any\')' field contains the text '\*/master'. To the right of this field is a red 'X' button and a help icon. Below the 'Branches to build' section, there is an 'Add Branch' button.

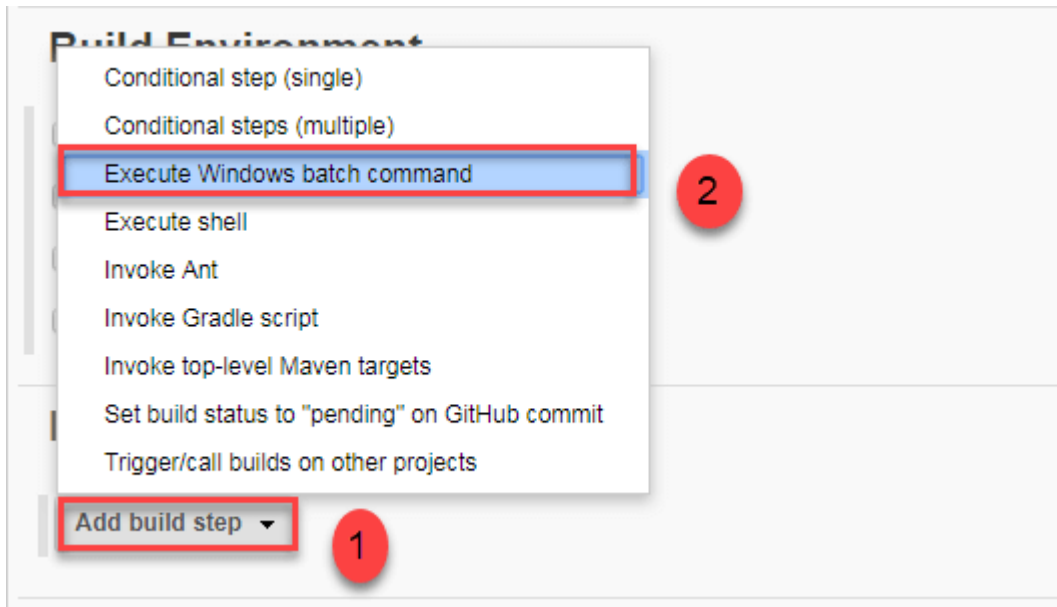
It is also possible for you to use a local repository.

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)** Now that you have provided all the details, it's time to build the code. Tweak the settings under the **build** section to build the code at the time you want. You can even schedule the build to happen periodically, at set times.

Under **build**,

1. Click on "**Add build step**"
2. Click on "**Execute Windows batch command**" and add the commands you want to execute during the build process.

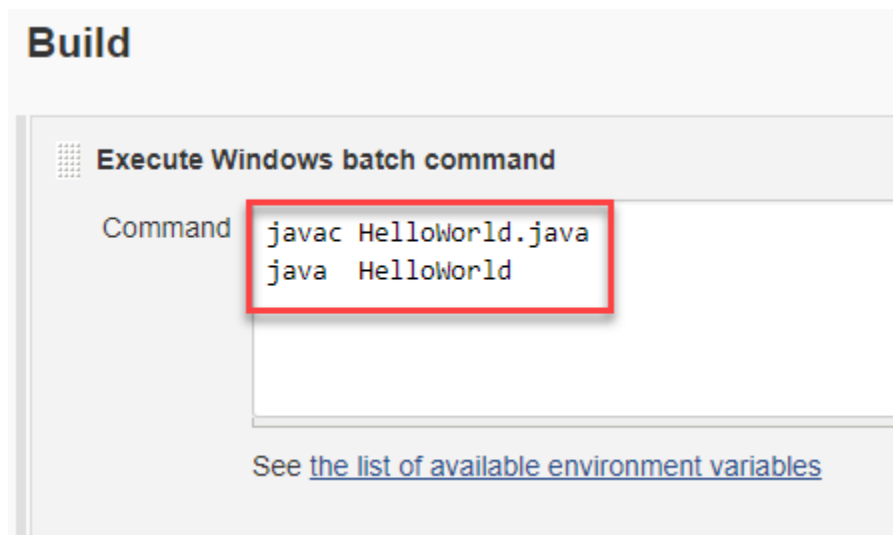


Here, We have added the java commands to compile the java code.

We have added the following windows commands:

```
javac HelloWorld.java
```

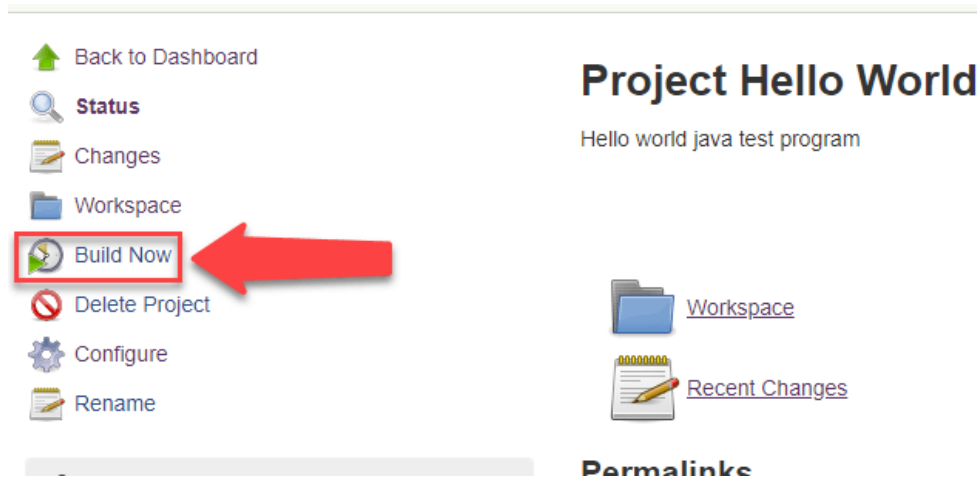
```
java HelloWorld
```



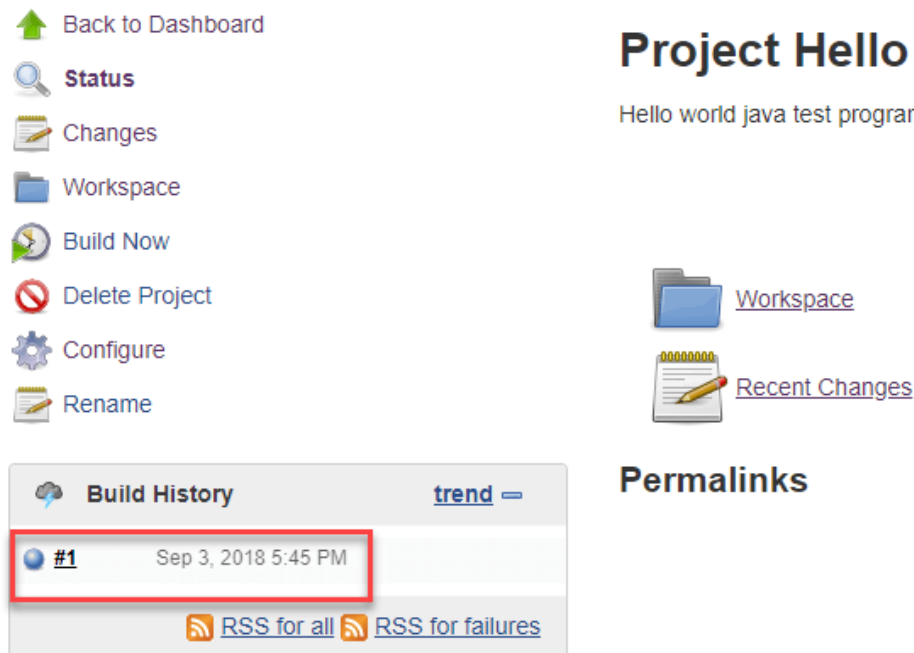
**Step 7)** When you have entered all the data,

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.



**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.

Jenkins > Hello World > #1

Back to Project  
Status  
Changes  
**Console Output**  
View as plain text  
Edit Build Information  
Delete Build  
Next Build

## Console Output

Started by user The

Building in workspace C:\Program Files (x86)\Jenkins\workspace\Hello World

Cloning the remote Git repository

Cloning repository <https://github.com/firstJava.git>

```
> git.exe init C:\Program Files (x86)\Jenkins\workspace\Hello World # timeout=10
Fetching upstream changes from https://github.com/firstJava.git
> git.exe --version # timeout=10
> git.exe fetch --tags --progress https://github.com/firstJava.git +refs/
> git.exe config remote.origin.url https://github.com/firstJava.git # ti
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/
> git.exe config remote.origin.url https://github.com/firstJava.git # ti
Fetching upstream changes from https://github.com/firstJava.git
> git.exe fetch --tags --progress https://github.com/firstJava.git +refs/
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
> git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
> git.exe rev-parse "origin/master^{commit}" # timeout=10
```

C:\Program Files (x86)\Jenkins\workspace\Hello World>javac HelloWorld.java

C:\Program Files (x86)\Jenkins\workspace\Hello World>java HelloWorld  
Hello World

**Finished: SUCCESS**

In sum, we have executed a HelloWorld program hosted on GitHub. Jenkin pulls the code from the remote repository and builds continuously at a frequency you define.