

Jenkins-Job-With GITHUB Integration

Objective

Create an Jenkins job with GITHUB integration

Steps

1. Create a local repo with GIT or github desktop and create the below mention file

class Simple{

```
    public static void main(String args[]) throws InterruptedException{
```

```
        Thread.sleep(10000);
```

```
        System.out.println("Hello Java");
```

```
    }
```

```
}
```

2. Publish the repo to the remote GITHUB.
Example → <https://github.com/Vishwanathms/Java-Proj1>
3. Create an Jenkins job as below.

The screenshot shows the Jenkins job configuration page for a job named "First Java Project". The "General" tab is selected, showing the job description, a "Discard old builds" checkbox, and build strategy settings. The "Strategy" dropdown is set to "Log Rotation". The "Days to keep builds" is set to 3, and the "Max # of builds to keep" is set to 6. A "Preview" link is visible below the description field.

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Description: First Java Project

[Plain text] [Preview](#)

☒ Discard old builds

Strategy: Log Rotation

Days to keep builds: 3
if not empty, build records are only kept up to this number of days

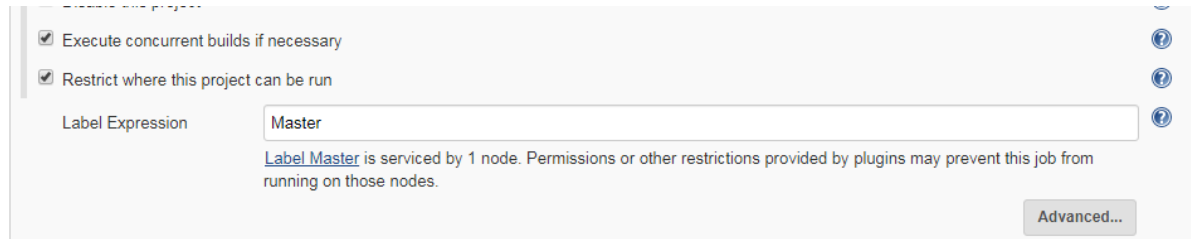
Max # of builds to keep: 6
if not empty, only up to this number of build records are kept

[Advanced...](#)

This would keep the history builds for max no of days of 3 or Max of "6" build.

No of builds takes the precedence over the Max no of days.

Jenkins-Job-With GITHUB Integration



Execute concurrent builds if necessary ☒

Restrict where this project can be run ☒

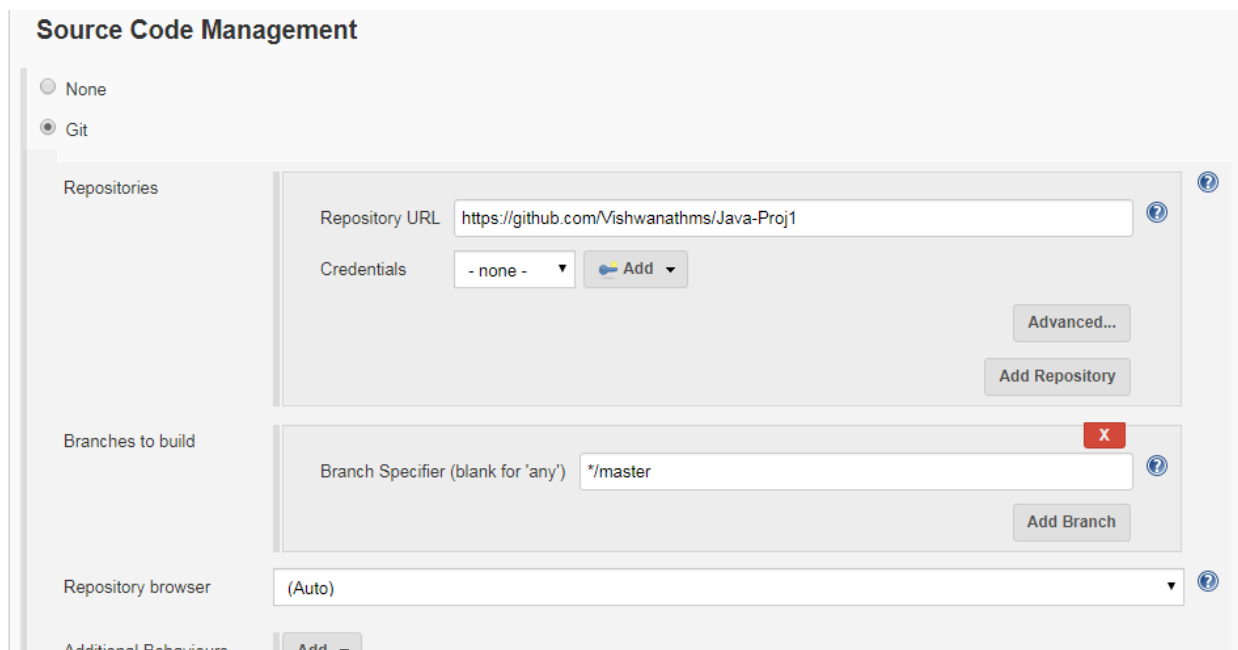
Label Expression

[Label Master](#) is serviced by 1 node. Permissions or other restrictions provided by plugins may prevent this job from running on those nodes.

Advanced...

First option is to execute multiple builds together.

Next is to run this job only on Master , as the label is mentioned with it.



Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials [Add](#)

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

Repository browser

Additional Behaviours [Add](#)

Here you configure the github repo link which the Jenkins should pull.

Note→ for this to work successfully, we should hve installed “git” on the master Jenkins

yum install git

```
[root@localhost ~]# yum install git
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * epel: download.nus.edu.sg
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
Package git-1.8.3.1-20.el7.x86_64 already installed and latest version
Nothing to do
[root@localhost ~]#
```

And configure the same in the “Global Tool Configuration” under the “Manage Jenkins”

Jenkins-Job-With GITHUB Integration



Global Tool Configuration

Maven Configuration

Default settings provider

Default global settings provider

JDK

JDK installations

List of JDK installations on this system

Git

Git installations

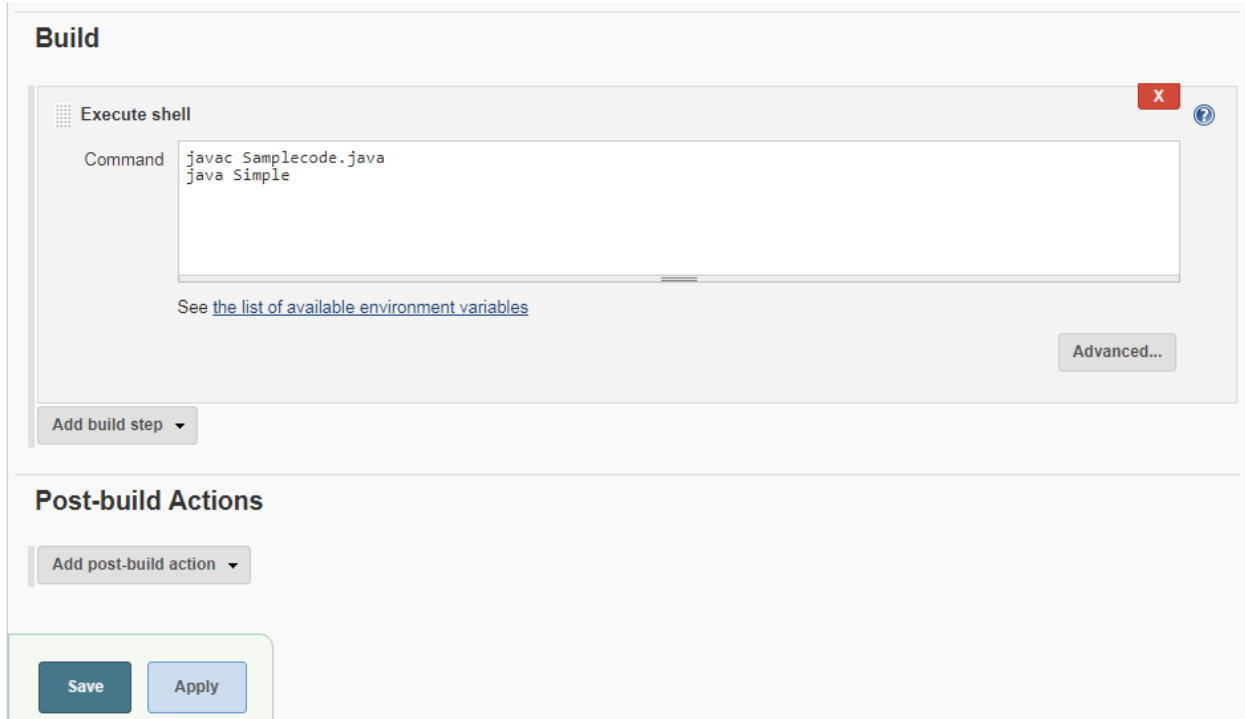
Git	
Name	<input type="text" value="Default"/>
Path to Git executable	<input type="text" value="/usr/bin/git"/>
<input type="checkbox"/> Install automatically	<input type="checkbox"/>

This will make sure the git pull is successfully by Jenkins while executing the jobs.

Note→ Jenkins clones the repo's into its local "workspace" area.

Finally now, lets configure the Build

Jenkins-Job-With GITHUB Integration



The screenshot shows the Jenkins 'Build' configuration page. Under the 'Execute shell' step, the command field contains:
`javac Samplecode.java
java Simple`
Below the command field is a link: [See the list of available environment variables](#). An 'Advanced...' button is located at the bottom right of the step configuration. Below the step configuration is an 'Add build step' dropdown. Under the 'Post-build Actions' section, there is an 'Add post-build action' dropdown. At the bottom left, there are 'Save' and 'Apply' buttons.

For the Build to work successfully.

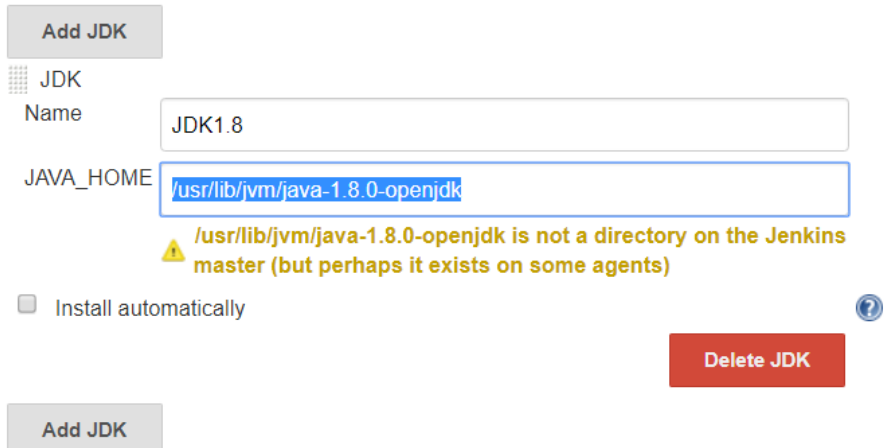
We would need to install the JAVA JDK on the Jenkins master.

```
# sudo yum install java-1.8.0-openjdk-devel.
```


Also in the “Global tool Configuration” configure the below

JDK

JDK installations



The screenshot shows the 'JDK' section of the Jenkins 'Global Tool Configuration' page. It features an 'Add JDK' button at the top. Below it, a table lists the configured JDKs:

JDK
<div>Name: JDK1.8</div> <div>JAVA_HOME: /usr/lib/jvm/java-1.8.0-openjdk</div> <div> /usr/lib/jvm/java-1.8.0-openjdk is not a directory on the Jenkins master (but perhaps it exists on some agents)</div> <div><input type="checkbox"/> Install automatically</div>

At the bottom right of the table is a red 'Delete JDK' button. At the bottom left, there is another 'Add JDK' button.

4. Check the build

Now when we execute the build

- Back to Project
- Status
- Changes
- Console Output**
- View as plain text
- Edit Build Information
- Git Build Data
- No Tags
- Previous Build

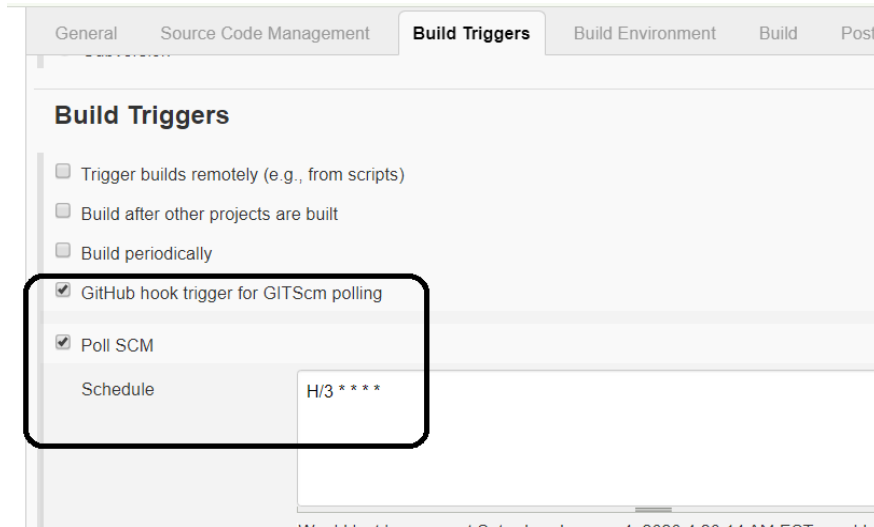
Console Output

```
Started by user user1
Running as SYSTEM
Building on master in workspace /var/lib/jenkins/workspace/Proj-Java1
No credentials specified
> /usr/bin/git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> /usr/bin/git config remote.origin.url https://github.com/Vishwanathms/Java-Proj1 # timeout=10
Fetching upstream changes from https://github.com/Vishwanathms/Java-Proj1
> /usr/bin/git --version # timeout=10
> /usr/bin/git fetch --tags --progress https://github.com/Vishwanathms/Java-Proj1 +refs/heads/*:refs/rem
> /usr/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> /usr/bin/git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 4501f32f013f45c9bb4bd17569b97475364a6e25 (refs/remotes/origin/master)
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f 4501f32f013f45c9bb4bd17569b97475364a6e25
Commit message: "Update Samplecode.java"
> /usr/bin/git rev-list --no-walk 4501f32f013f45c9bb4bd17569b97475364a6e25 # timeout=10
[Proj-Java1] $ /bin/sh -xe /tmp/jenkins8966977132316119950.sh
+ javac Samplecode.java
+ java Simple
Hello Java
Finished: SUCCESS
```

This would be the output.

5. Let's change some content in the repository for Jenkins to trigger the build automatically.

I have edited the repo on the github and committed the changes and in next 3 min as mentioned in the build, the below build is triggered automatically



Now, do the changes in the repo, that is on the .java file and the Jenkins job should trigger automatically as shown below.

- Back to Project
- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#25'
- Polling Log
- Git Build Data
- No Tags
- Previous Build

Console Output

Started by an SCM change

```
Running as SYSTEM
Building on master in workspace /var/lib/jenkins/workspace/Proj-Java1
No credentials specified
> /usr/bin/git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> /usr/bin/git config remote.origin.url https://github.com/Vishwanathms/Java-Proj1 # timeout=10
Fetching upstream changes from https://github.com/Vishwanathms/Java-Proj1
> /usr/bin/git --version # timeout=10
> /usr/bin/git fetch --tags --progress https://github.com/Vishwanathms/Java-Proj1 +refs/heads/*
> /usr/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> /usr/bin/git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 6c6bacac58ad89594d4bd250cae62f1e2ea4db41 (refs/remotes/origin/master)
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f 6c6bacac58ad89594d4bd250cae62f1e2ea4db41
Commit message: "Update Samplecode.java"
> /usr/bin/git rev-list --no-walk 4501f32f013f45c9bb4bd17569b97475364a6e25 # timeout=10
[Proj-Java1] $ /bin/sh -xe /tmp/jenkins3096210502890123652.sh
+ javac Samplecode.java
+ java Simple
Hello Java
Finished: SUCCESS
```