

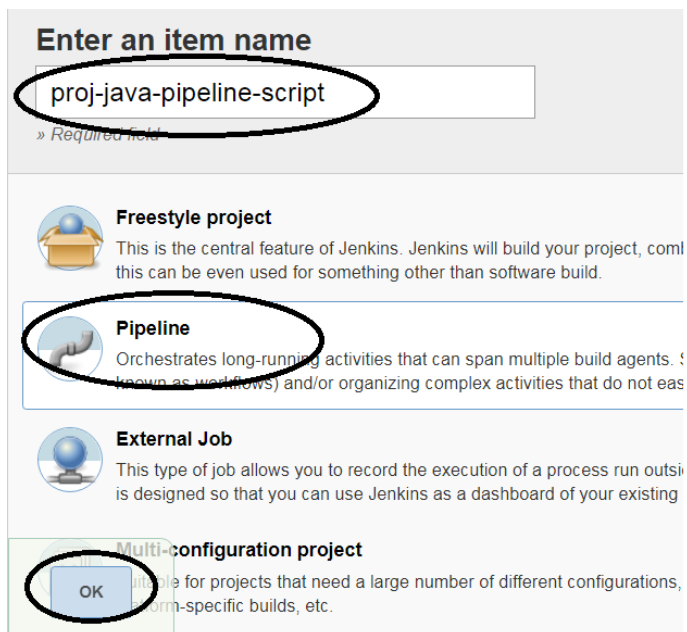
Jenkins-Job-With GITHUB on Pipeline Script

Lab Scenario for Jenkins with GITHUB integration as a pipeline script

1. Create the “Java code” repo as below on the github



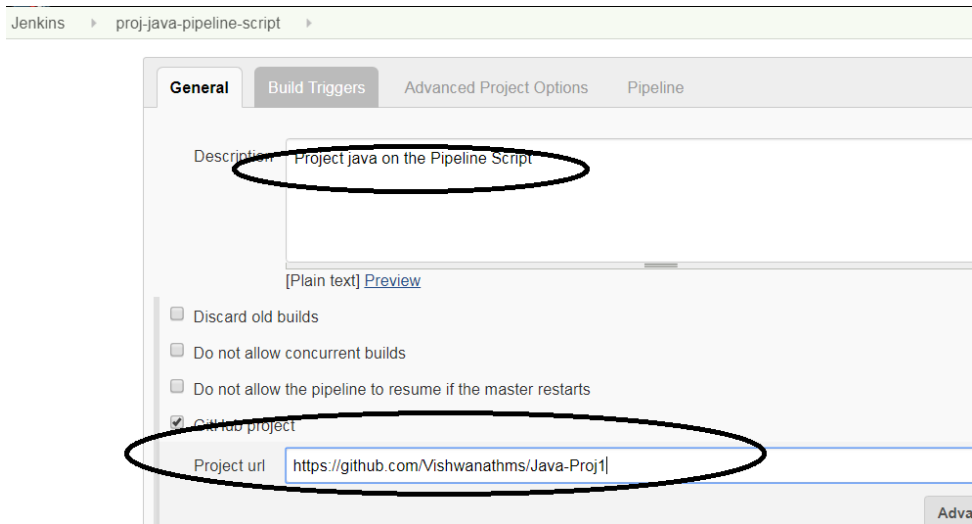
2. Create a Jenkins pipeline script for this java code to run.



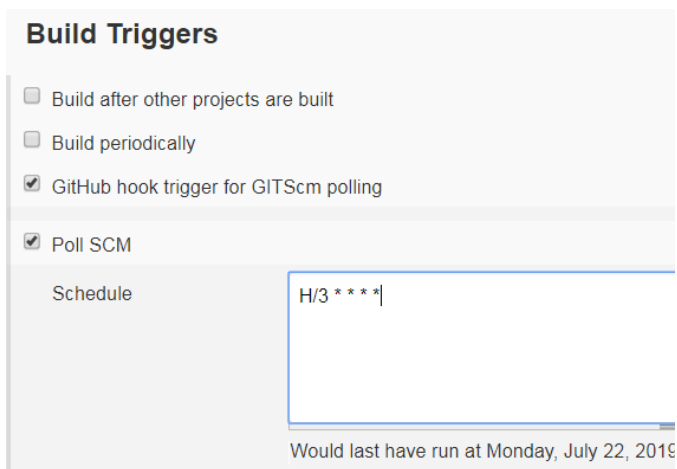
We would need to select “pipeline” instead of “Freestyle project”

And Click “OK”

Jenkins-Job-With GITHUB on Pipeline Script

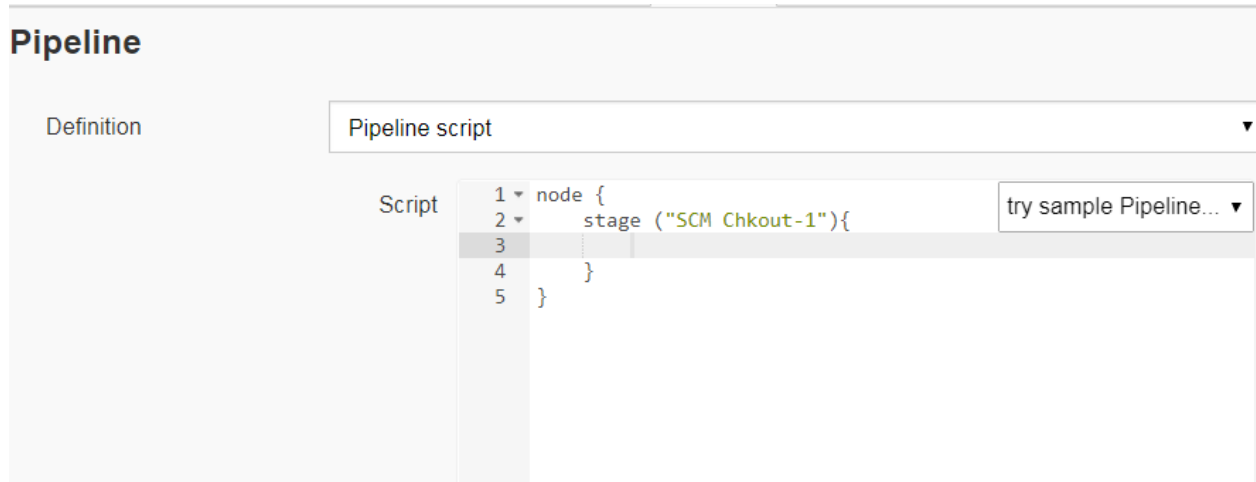


Paste the link of the java project github link for a automatic trigger.



This would trigger the pipeline script, if any changes occur in the Java GITHUB repo.

Now going to the Pipeline script



The below script is the basic skeleton of the pipeline script

```
NODE {
    STAGE ("SCM CHKOUT-1"){

    }
}
```

Now we would need to add the stages we want to execute.

We can start by clicking on **"pipeline syntax"**

We get the below new page.

[Back](#)

- Snippet Generator**
- Declarative Directive Generator
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

archiveArtifacts: Archive the artifacts

Files to archive

Advanced...

Generate Pipeline Script

Here we can generate any kind of pipeline scripts and paste it back in the main pipeline script JOB.

Jenkins-Job-With GITHUB on Pipeline Script

For example.

Lets create an GIT pipeline script.

The screenshot shows the 'Steps' configuration in Jenkins. Under the 'Sample Step' tab, the 'git' step is selected. The 'Repository URL' is set to 'https://github.com/Vishwanathms/Java-Proj1'. The 'Branch' is set to 'master'. The 'Credentials' dropdown is set to '- none -' with an 'Add' button. There are two checked options: 'Include in polling?' and 'Include in changelog?'. A blue button labeled 'Generate Pipeline Script' is visible. Below it, the generated script is shown: 'git \'https://github.com/Vishwanathms/Java-Proj1\''. A callout bubble points to this script with the text 'Pipeline Script generated'.

Now paste this script back in the main pipeline job page.

The screenshot shows the 'Pipeline' configuration page. The 'Definition' dropdown is set to 'Pipeline script'. The 'Script' text area contains the following code:

```
1 node {  
2   stage ("SCM Checkout-1") {  
3     git 'https://github.com/Vishwanathms/Java-Proj1'  
4   }  
5 }
```

The line 'git 'https://github.com/Vishwanathms/Java-Proj1'' is highlighted with a red box. Below the script area, there is a checkbox for 'Use Groovy Sandbox' which is checked. At the bottom, there are 'Save' and 'Apply' buttons.

Now lets click “**Save**” and lets build this project to chk if the repo is downloaded to the Jenkins server into the project workspace or not.

Jenkins-Job-With GITHUB on Pipeline Script

Jenkins > proj-java-pipeline-script > #1

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#1'

Git Build Data

No Tags

Replay

Console Output

Started by user [user1](#)
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on [Jenkins](#) in /var/lib/jenkins/workspace/proj-ja
[Pipeline] {
[Pipeline] stage
[Pipeline] { (SCM Chkout-1)
[Pipeline] git
No credentials specified
Cloning the remote Git repository
Cloning repository <https://github.com/Vishwanathms/Java->
> /usr/bin/git init /var/lib/jenkins/workspace/proj-ja

Build is triggered manually.

And it show "Success".

```
> /usr/bin/git checkout -b master 6c6bacac58ad89594d4bd250cae62f1e2ea4db41
Commit message: "Update Samplecode.java"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Lets check on the Jenkins server now.

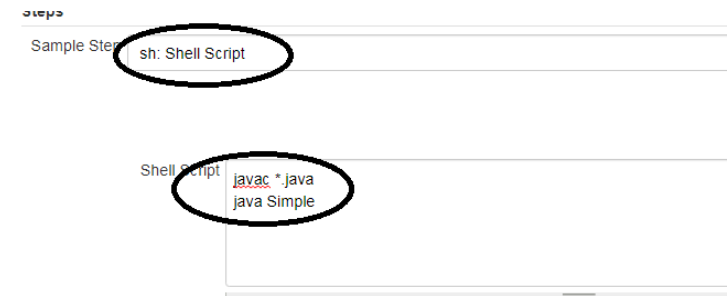
```
[root@localhost workspace]# cd proj-java-pipeline-script/
[root@localhost proj-java-pipeline-script]# pwd
/var/lib/jenkins/workspace/proj-java-pipeline-script
[root@localhost proj-java-pipeline-script]# ls -l
total 4
-rw-r--r--. 1 root root 166 Jul 22 21:58 Samplecode.java
[root@localhost proj-java-pipeline-script]#
```

This cleared indicates, the ".java" file was download from the github repo.

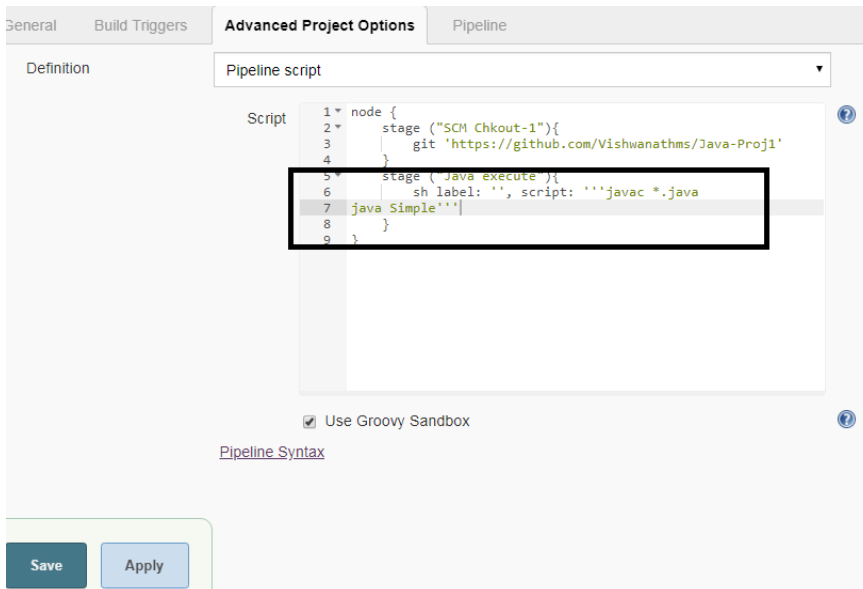
Next lets run the java code for the output.

Going back to the pipeline syntax.

Jenkins-Job-With GITHUB on Pipeline Script



Lets paste this generated pipeline script back to the main Script page.



I hope this is understood.

And then click "Save"

Now its time to BUILD the project again.

The output of the build is as below, the java code was run successfully.

Jenkins-Job-With GITHUB on Pipeline Script

```
> /usr/bin/git fetch --tags --progress https://github.com/Vis
+refs/heads/*:refs/remotes/origin/*
> /usr/bin/git rev-parse refs/remotes/origin/master^{commit}
> /usr/bin/git rev-parse refs/remotes/origin/origin/master^{c
Checking out Revision 6c6bacac58ad89594d4bd250cae62f1e2ea4db41
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f 6c6bacac58ad89594d4bd250cae62f1e2e
> /usr/bin/git branch -a -v --no-abbrev # timeout=10
> /usr/bin/git branch -D master # timeout=10
> /usr/bin/git checkout -b master 6c6bacac58ad89594d4bd250cae
Commit message: "Update Samplecode.java"
> /usr/bin/git rev-list --no-walk 6c6bacac58ad89594d4bd250cae
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Java execute)
[Pipeline] sh
+ javac Samplecode.java
+ java Simple
Hello Java
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Let's cross verify on the Jenkins server as well.


```
[root@localhost proj-java-pipeline-script]# ls -l
total 8
-rw-r--r--. 1 root root 166 Jul 22 21:58 Samplecode.java
-rw-r--r--. 1 root root 545 Jul 22 22:06 Simple.class
[root@localhost proj-java-pipeline-script]# pwd
/var/lib/jenkins/workspace/proj-java-pipeline-script
[root@localhost proj-java-pipeline-script]#
```

The "Simple" class file is created as the result of the java compilation.

Troubleshooting:

Incase the build fails → saying no agent available and if the cursor is hung and waiting, means there are no Available agent to process the request.

As per the previous lab, we have set the below configuration under , Manage Jenkins → Configuration.

Labels	Master
Usage	Only build jobs with label expressions matching this node 

For now, to execute the project, lets change the Usage to "Use this node as much as possible"

3. Create a Jenkins advance pipeline script for the same java code to run.

```
pipeline {
  agent { label 'Master' }
  stages {
    stage ("SCM Chkout-1"){
      steps {
        git 'https://github.com/Vishwanathms/Java-Proj1'
      }
    }
    stage ("Java execute"){
      steps {
        sh label: "", script: "'javac *.java java Simple'"
      }
    }
  }
}
```

And then do a build , even this would work fine.

4. Put the Jenkins pipeline in the SCM and manage the changes of the pipeline code.

Either you can create a new repository in github for this, or we can use the same repository that has the java code in it.

In our case, we would be using the same repo of java code.



```
1 pipeline {
2   agent { label 'Master' }
3   stages {
4     stage ("SCM Chkout-1"){
5       steps {
6         git 'https://github.com/Vishwanathms/Java-Proj1'
7       }
8     }
9     stage ("Java execute"){
10      steps {
11        sh 'javac *.java'
12        sh 'java Simple'
13      }
14    }
15  }
16 }
```

we have create a file in the repo called **"jenkinsfile"**.

With the same pipeline script.

Now lets, change the option the Jenkins job to point it to this jenkinsfile

Jenkins-Job-With GITHUB on Pipeline Script

The screenshot shows the Jenkins Pipeline configuration page. The 'Pipeline' tab is selected. The 'Definition' section has 'Pipeline script from SCM' selected. The 'SCM' dropdown is set to 'Git'. The 'Repository' field is set to 'https://github.com/Vishwanatl'. The 'Credentials' dropdown is set to '- none -'. The 'Branches to build' section has 'Branch Specifier (blank for \'any\')' set to '*/master'. The 'Repository browser' is set to '(Auto)'. The 'Script Path' is set to 'jenkinsfile'. The 'Save' and 'Apply' buttons are visible at the bottom left.

Note→ the file name under – **script path** should match the name of the file in the repo that we just created.

Then execute the build and check if successful.

Also, since we have configured, the SCM poll

The screenshot shows the Jenkins Poll SCM configuration page. The 'Poll SCM' checkbox is checked. The 'Schedule' field is set to 'H/3 * * * *'. The 'Would last have run at Tuesday, July 23, 2019 10:08:02 PM EDT; would next run at EDT.' text is displayed. The 'Ignore post-commit hooks' checkbox is unchecked.

Lets do some changes to the “**jenkinsfile**” in the repo, and commit it, so that the changes triggers the build automatically.

Note→ which is the exact way that is done in the production/Development env.

Jenkins-Job-With GITHUB on Pipeline Script

The output is below.

to Project

JS

nges

sole Output

ew as plain text

Build Information

ng Log

ad Dump

se/resume

ay

line Steps

spaces

ious Build



Console Output

Started by an SCM change

Obtained jenkinsfile from git <https://github.com/Vishwanathms/Java-Proj1>

Running in Durability level: MAX_SURVIVABILITY

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](#) in /var/lib/jenkins/workspace/proj-java-pipeline-script

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

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>

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/remotes/origin/*

> /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

Thanks, -- END of this LAB..