

How To Create Jenkins Shared Library For Pipelines

by **devopscube** · July 7, 2019



In this tutorial, you will learn how to create a basic shared library and integrate it with Jenkins and a sample pipeline.

Note: If you want to learn the shared library basic concepts and use cases, please [take a look at this article](#).

We will look into the following four things to get your hands dirty with the shared library.

- 1 Create a Shared Library Structure
- 2 Create Custom Shared Library Code
- 3 Configure Shared Library In Jenkins Configuration
- 4 Create Declarative Pipeline as Code With Shared Library.

Let's look at each one in detail.

Create a Shared Library Structure

Note: In this guide, we will be concentrating only on the vars folder for creating your first shared library. src and resources will be covered in the advanced shared library guide.

Jenkins shared library has the following structure. You can get the basic structure and code used in this article from Github -> [Jenkins Shared Library Structure](#)

```
jenkins-shared-library
|__vars
|__src
|__resources
```

All the files under vars are global functions and variables. The file name is the function name. We will be using the filename in our declarative pipeline.

Create Custom Shared Library Code

In this section, we will create the shared library code for Git Checkout functionality.

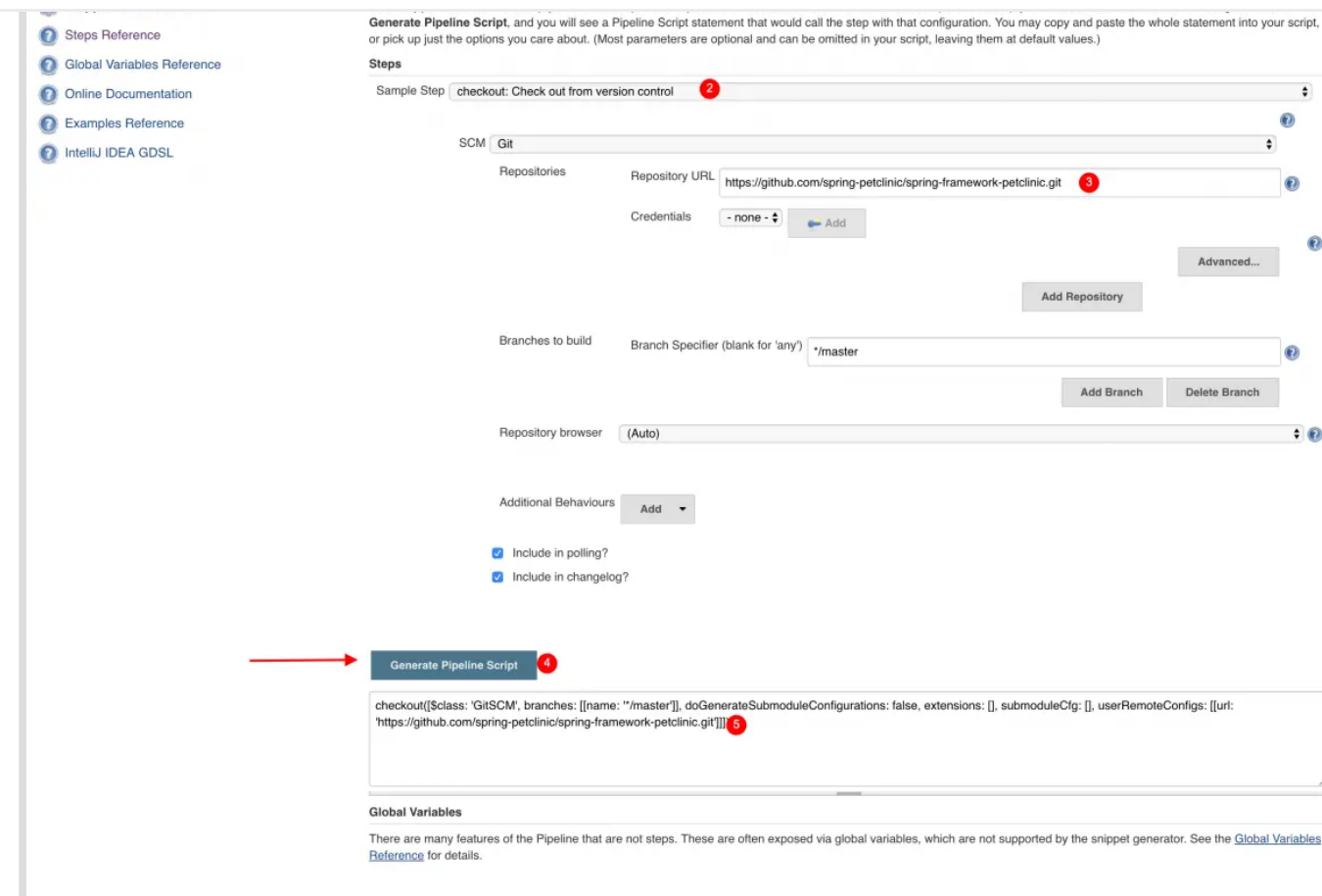
Generate Pipeline Syntax Using Snippet Generator:

You can create the code snippets that can be used in share library function using the Pipeline Syntax Generator available in Jenkins. This will make our life easier for creating custom library DSL. All the supported pipeline functionality can be generated from the snippet generator.

You can access the syntax generator from your Jenkins on /pipeline-syntax/ path. For example,

```
http://devopscube-jenkins.com:8080/pipeline-syntax/
```

Here is the screenshot which shows creating a git checkout pipeline snippet using the pipeline syntax generator.



Number 5 in the screenshot shows the generated snippet. Here is the properly formatted checkout snippet.

```
checkout([
    $class: 'GitSCM',
    branches: [[name: '*/master']],
    doGenerateSubmoduleConfigurations: false,
    extensions: [],
    submoduleCfg: [],
    userRemoteConfigs: [[url: 'https://github.com/spring-projects/spring-
petclinic.git']]
])
```

Create a Shared Library For Git Checkout

Lets convert the checkout snippet we generated in the above step to a shared library.

Create a file named `gitCheckout.groovy` under vars folder.

Here is our Git Checkout shared library code. We have removed all the empty checkout parameters which got generated by default.

```
def call(Map stageParams) {

    checkout([
        $class: 'GitSCM',
        branches: [[name: stageParams.branch ]],
        userRemoteConfigs: [[ url: stageParams.url ]]
    ])
}
```

Here is the code explanation,

Map as an argument. From the pipeline stage, we will pass multiple arguments which get passed as a map to the shared library.

2 `stageParams.branch` – its the branch parameter which comes from the pipeline stage and we use `stageParams` to access that variable in the shared library.

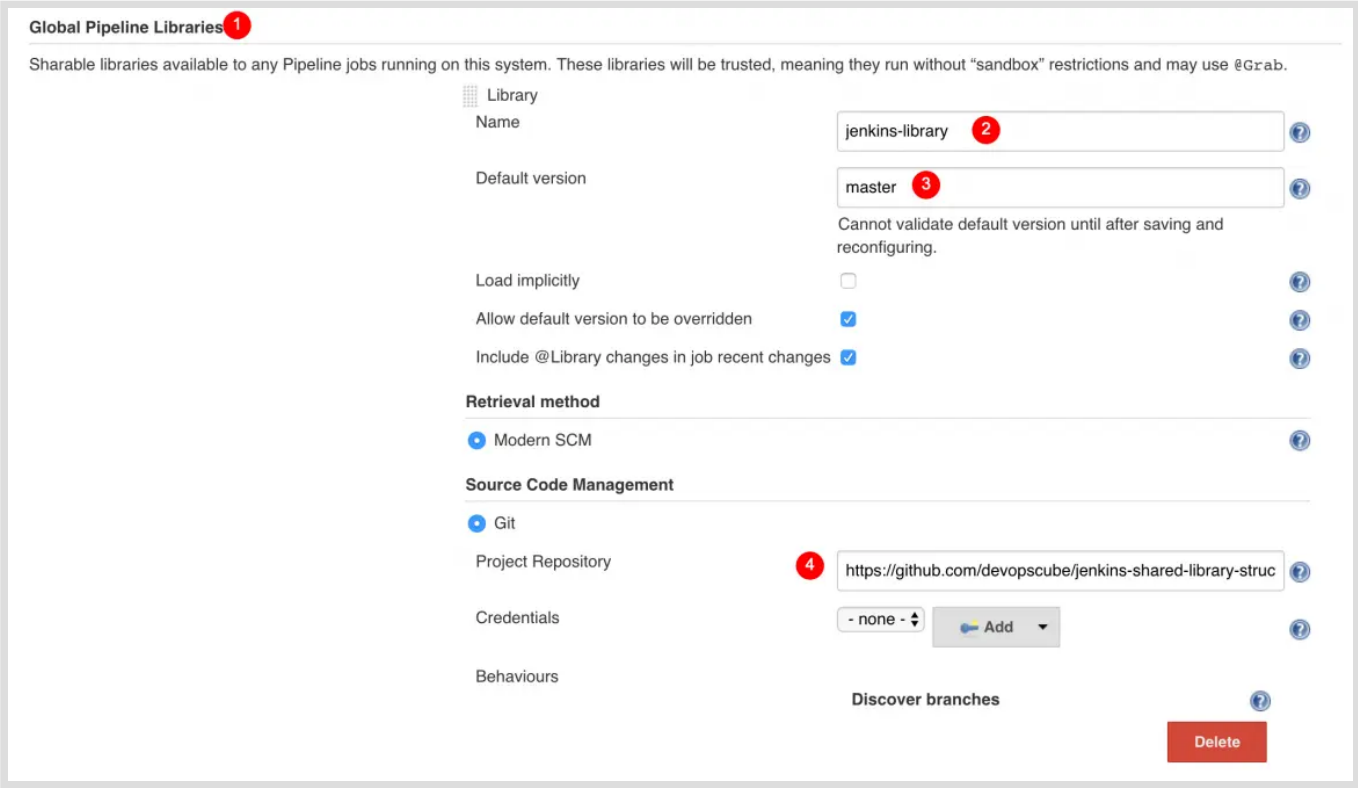
Commit the changes and push it to your repository.

Add Github Shared Library Repo to Jenkins

Now that we have a basic git checkout library ready lets add it to Jenkins configurations.

Step 1: Go to Manage Jenkins → Configure System

Step 2: Find the **Global Pipeline Libraries** section and add your repo details and configurations as shown below.



The screenshot shows the 'Global Pipeline Libraries' configuration page in Jenkins. The page title is 'Global Pipeline Libraries' with a red '1' next to it. Below the title, it says 'Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Gzab.' There is a table with one row for a library named 'jenkins-library' (marked with a red '2'). The 'Default version' is set to 'master' (marked with a red '3'). Below the table, there are checkboxes for 'Load implicitly' (unchecked), 'Allow default version to be overridden' (checked), and 'Include @Library changes in job recent changes' (checked). Under the 'Retrieval method' section, 'Modern SCM' is selected. Under the 'Source Code Management' section, 'Git' is selected. The 'Project Repository' is set to 'https://github.com/devopscube/jenkins-shared-library-struct' (marked with a red '4'). The 'Credentials' dropdown is set to 'none'. There is a 'Discover branches' button and a 'Delete' button at the bottom right.

Use Checkout Library in Declarative Pipeline

We always call the library using the filename under vars. In this case, gitCheckout is the filename created under vars. Here is how we call `gitCheckout` library from the pipeline or Jenkinsfile

```
stage('Git Checkout') {
    gitCheckout(
        branch: "master",
        url: "https://github.com/spring-projects/spring-petclinic.git"
    )
}
```

3
SHARES



```
@Library('jenkins-library@master') _

pipeline {
    agent any
    stages {
        stage('Git Checkout') {
            steps {
                gitCheckout(
                    branch: "master",
                    url: "https://github.com/spring-projects/spring-petclinic.git"
                )
            }
        }
    }
}
```

Like gitCheckout, you can create all your pipeline steps a shared library and you don't have to repeat your common functionalities in all your pipelines.

TAGS: [Pipeline as code](#)

3 SHARES:

 **SHARE** 3

 **TWEET**







devopscube

Established in 2014, a community for developers and system admins. Our goal is to continue to build a growing DevOps community offering the best in-depth articles, interviews, event listings, whitepapers, infographics and much more on DevOps.

[VIEW COMMENTS \(3\)](#)

YOU MAY ALSO LIKE



T — TUTORIAL

How To Setup an NFS Server and Client For File Sharing

by **devopscube** · February 4, 2016

In this tutorial, I will explain how to set up an NFS server and client configurations for mounting...

Jenkins Tutorial For Beginners: Step by Step Guides

by **devopscube** · July 24, 2021

Jenkins is the widely adopted open source continuous integration tool. A lot has changed in Jenkins 2.x when...

 — DEVOPS

List of DevOps Blogs and Resources for Learning

by **Bibin Wilson** · July 14, 2021

In this blog, I have added the best devops blogs that can follow. Also, I have added the...

 — DEVOPS

Setup Jenkins master and Build Slaves as Docker Container

by **Mi Jia** · October 20, 2017

Do you want dockerized Jenkins which includes the configuration for build slaves also as Docker containers? So that you can...

 — DEVOPS

How To Run Docker in Docker Container [3 Easy Methods]

by **Bibin Wilson** · June 25, 2021

In this blog, I will walk you through the steps required to run docker in docker using three...



Setting Up Azure CLI on Ubuntu Linux

by **devopscube** · December 8, 2015

Azure has a great web interface called azure portal for performing all the functions. But if you prefer...