Jenkins - Shared Libraries

Introduction
Configure Shared Library
Run a pipeline using Shared Library

Introduction

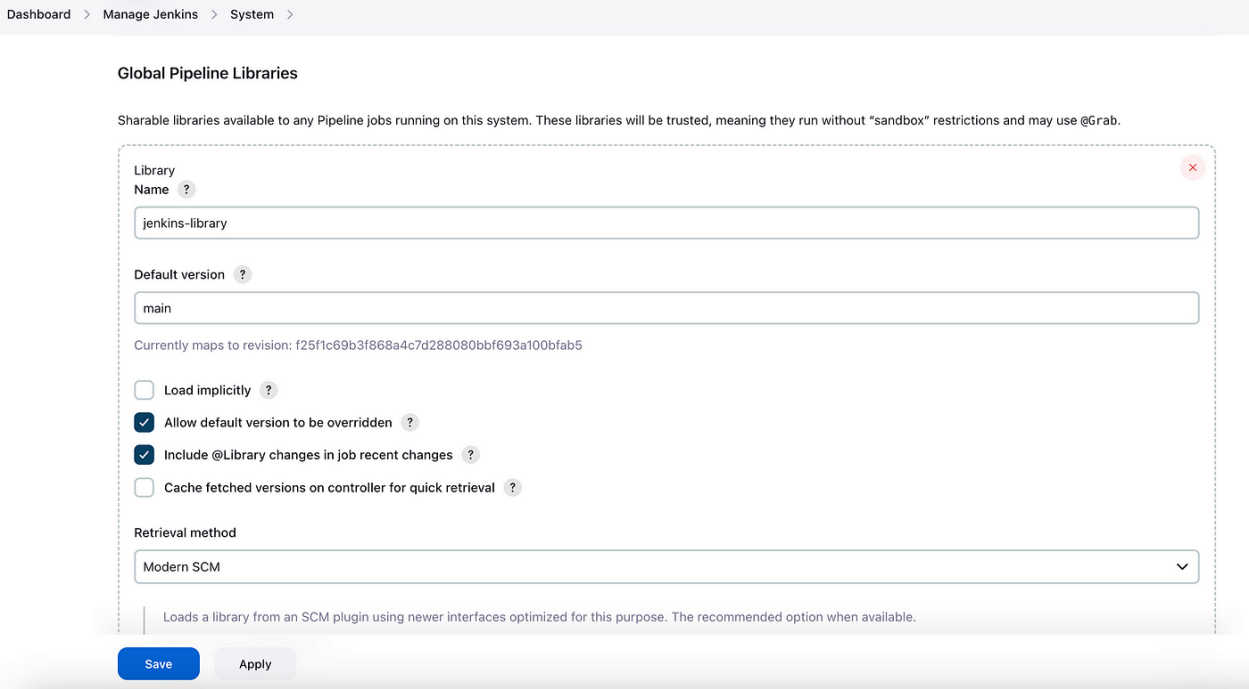
A shared library in the context of Jenkins, also known as a Jenkins shared library, is a powerful and flexible feature that allows you to centralize and reuse code, scripts, and common functionality across multiple Jenkins pipelines or jobs. Shared libraries are written in Groovy, and they can include a collection of reusable functions, classes, and pipeline steps. They provide several key benefits: code reusability, centralized management, team collaboration, versioning and so on.

Configure Shared Library

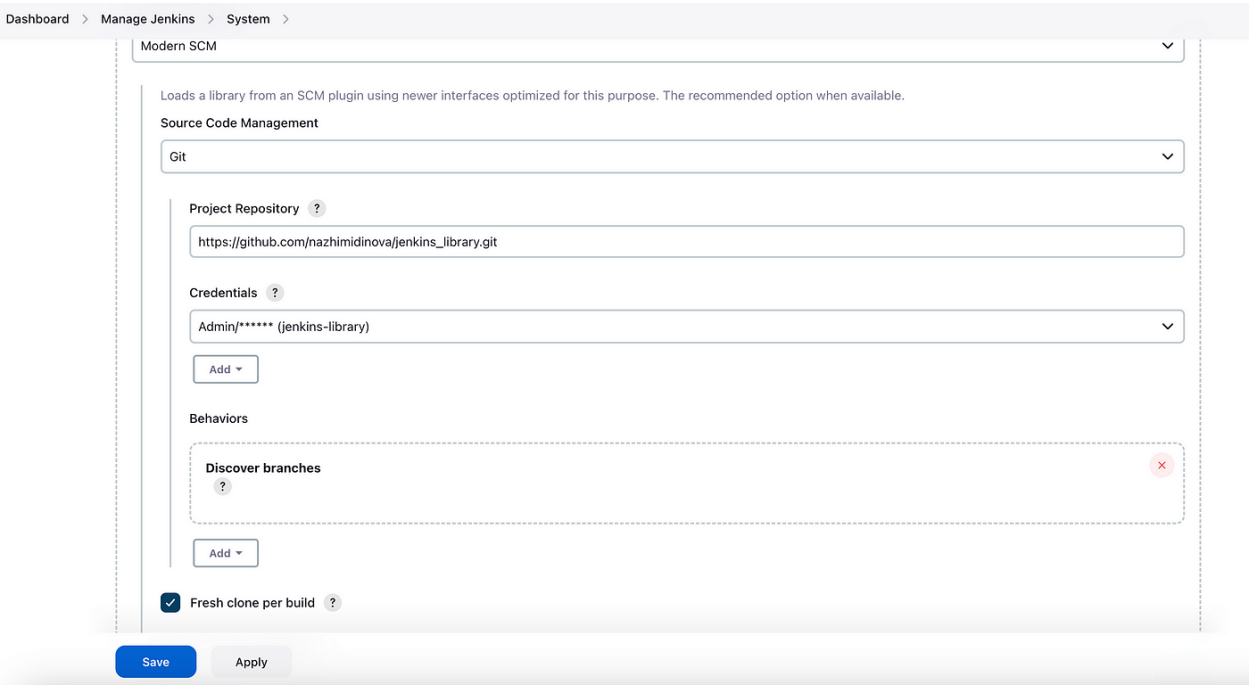
Prerequisites:

1. Have your folder and files ready in you repository
2. If your repositroy is private, configure new jenkins credentials with you github api token. Choose username with password option. Name it “jenkins-library”.

Step1. Go to Manage jenkins→System configuration→System→Global pipeline libraries. Give your library a name, you will referencing that name later on. Select your default version.



Step 2. Get HTTPS URL from your git repository and paste it under Project Repository. Select the credetials that you configured under prerequisites. Click "Save". This way we configured our Jenkins to be able to load shared libraries from SCM.



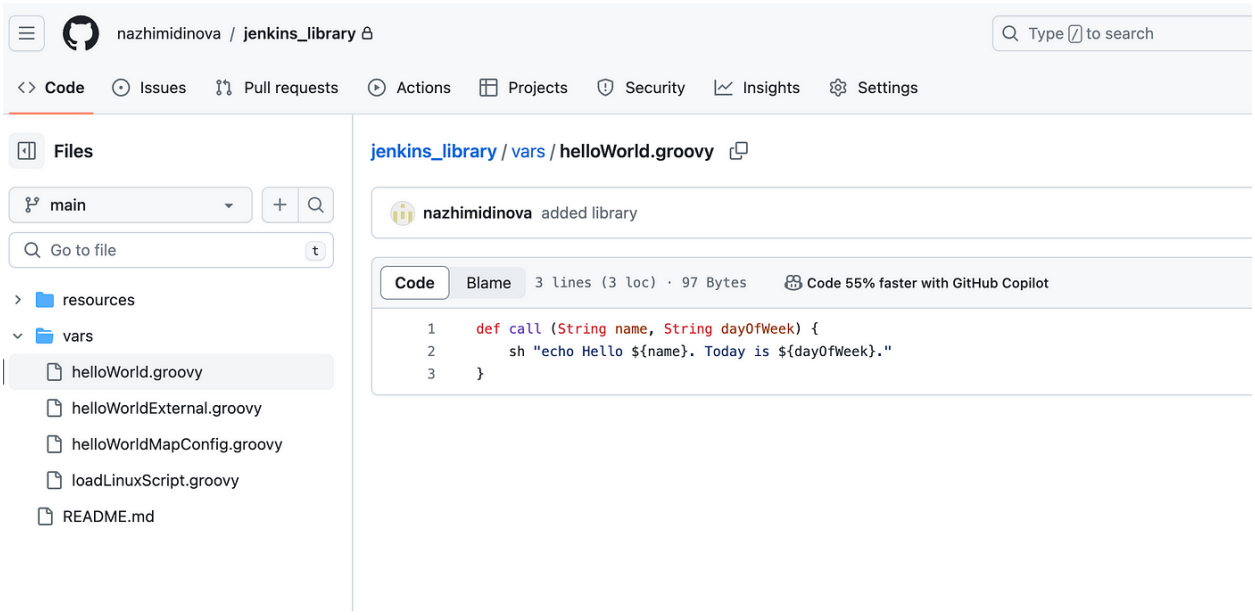
Run a pipeline using Shared Library

Step 1. Create a pipeline and give it a name

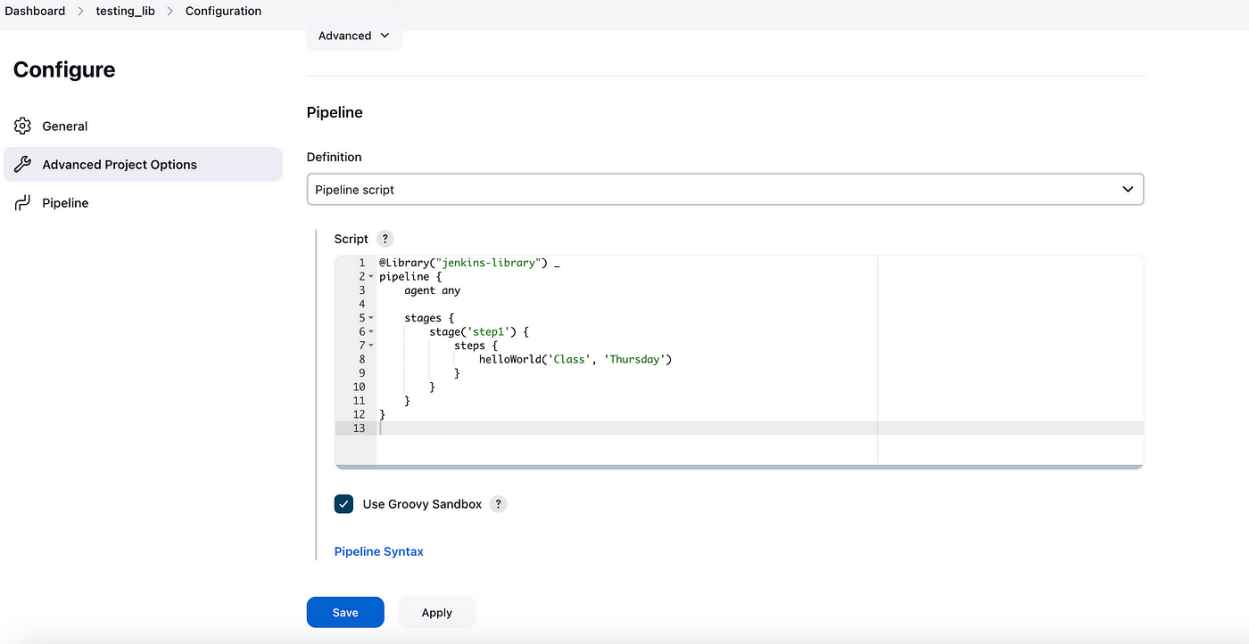
Step 2. This is my helloWorld.groovy file. Use camelCase when naming your files. In my groovy file I am defining a function and passing two variables. As you can see I have resources and vars folders, there is also third one src.

The file must be in vars or src directory directory as shown.

```
def call(String name, String dayOfWeek) {
    sh "echo Hello ${name}, today is ${dayOfWeek}."
}
```



Step 3. Make sure to reference a library and its name as shown below. After giving it a name, leave a space and type "_". Which means you want to load every file in this library. Inside the steps type your files name open the parenthesis and pass in parameters to your variables. Click "Save" and run your pipeline. It should run successfully.



The output of a build:

```
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/testing_lib
[Pipeline] {
[Pipeline] stage
[Pipeline] { (step1)
[Pipeline] sh
+ echo Hello Class. Today is Thursday.
Hello Class. Today is Thursday.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```