**Jenkins shared library**

**What is jenkins shared library?**

In Jenkins, a **Shared Library** is a powerful feature that allows you to define and reuse common code across multiple Jenkins pipelines. This helps in avoiding code duplication, promoting consistency, and making pipeline scripts more manageable and maintainable, especially in large projects or organizations with multiple Jenkins jobs.

**What is the use of jenkins shared library?**

The use of a Jenkins Shared Library is to create a centralized repository of reusable code and scripts that can be shared across multiple Jenkins pipelines. This offers several significant benefits:

**1. Code Reusability**

* **Avoid Duplication**: By storing common logic, functions, and classes in a Shared Library, you can avoid duplicating code across multiple pipelines. This makes your pipeline scripts cleaner and more concise.
* **Example**: A common build or deployment function that is used by multiple projects can be placed in a Shared Library, allowing all projects to use the same code.

**2. Consistency**

* **Standardization**: Shared Libraries allow you to enforce consistent practices across different pipelines. Whether it's a standardized way of building, testing, or deploying applications, the shared code ensures that all pipelines follow the same process.
* **Example**: A standardized way of checking out code, running tests, or sending notifications can be maintained in a Shared Library.

**3. Maintainability**

* **Easier Updates**: When you need to make changes to common code, you only need to update the Shared Library. All pipelines that use this library will automatically pick up the change, simplifying maintenance.
* **Example**: If there’s a need to update the deployment strategy (e.g., switching from a canary release to a blue-green deployment), you can make this change once in the Shared Library, and it will be applied to all relevant pipelines.

**4. Modularization**

* **Organized Code**: Shared Libraries allow you to modularize your Jenkins pipeline code. You can separate different concerns (e.g., build, test, deploy) into distinct functions or classes, making the codebase more organized and easier to navigate.
* **Example**: Separate scripts for handling Docker builds, Kubernetes deployments, or artifact storage can be maintained in the library.

**5. Collaboration**

* **Shared Development**: Teams can collaborate on the Shared Library, contributing new functions or improving existing ones. This encourages knowledge sharing and collective ownership of the pipeline codebase.
* **Example**: Different teams might contribute their specific expertise, such as security checks or performance optimizations, to the Shared Library.

**6. Versioning and Flexibility**

* **Version Control**: Shared Libraries can be versioned, allowing pipelines to use specific versions of the library. This is useful for managing changes and ensuring that pipelines continue to function correctly even when the library is updated.
* **Example**: A pipeline can specify to use version 1.2.3 of a library, ensuring stability even if a new version of the library introduces breaking changes.

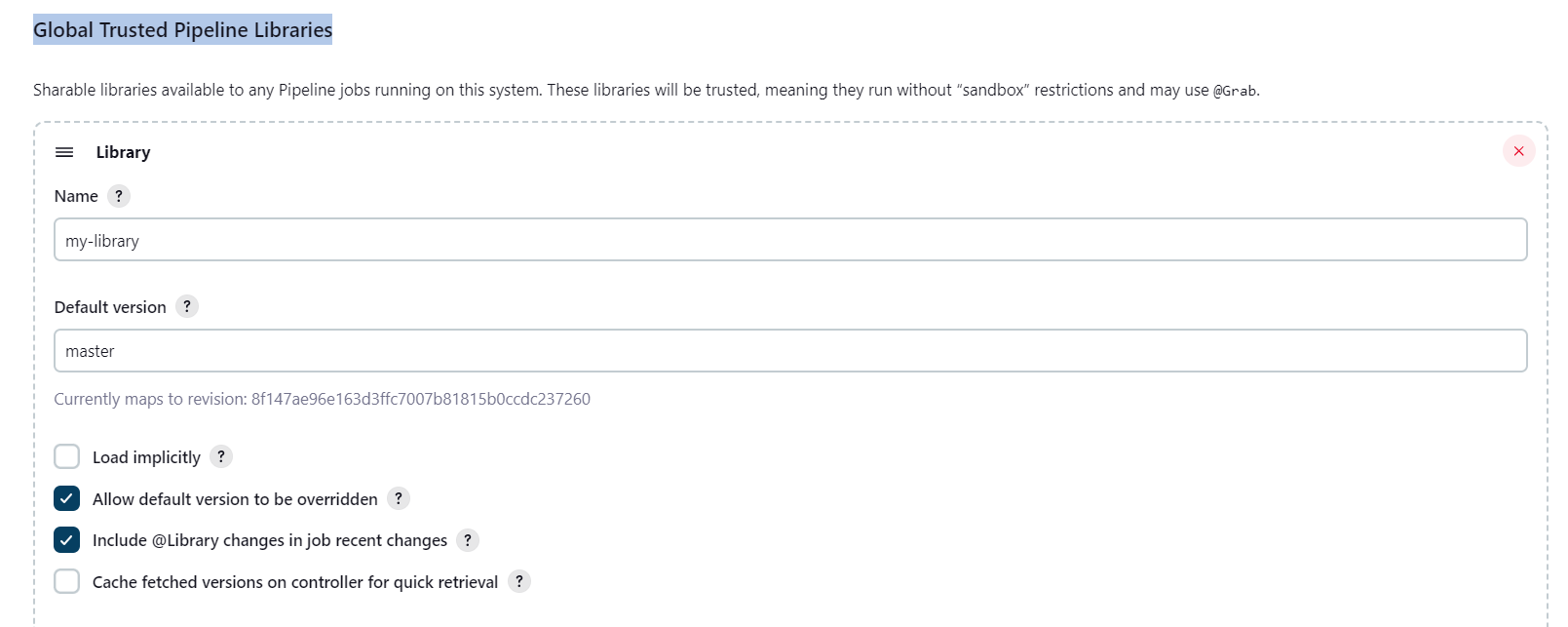
**7. Simplification of Pipeline Scripts**

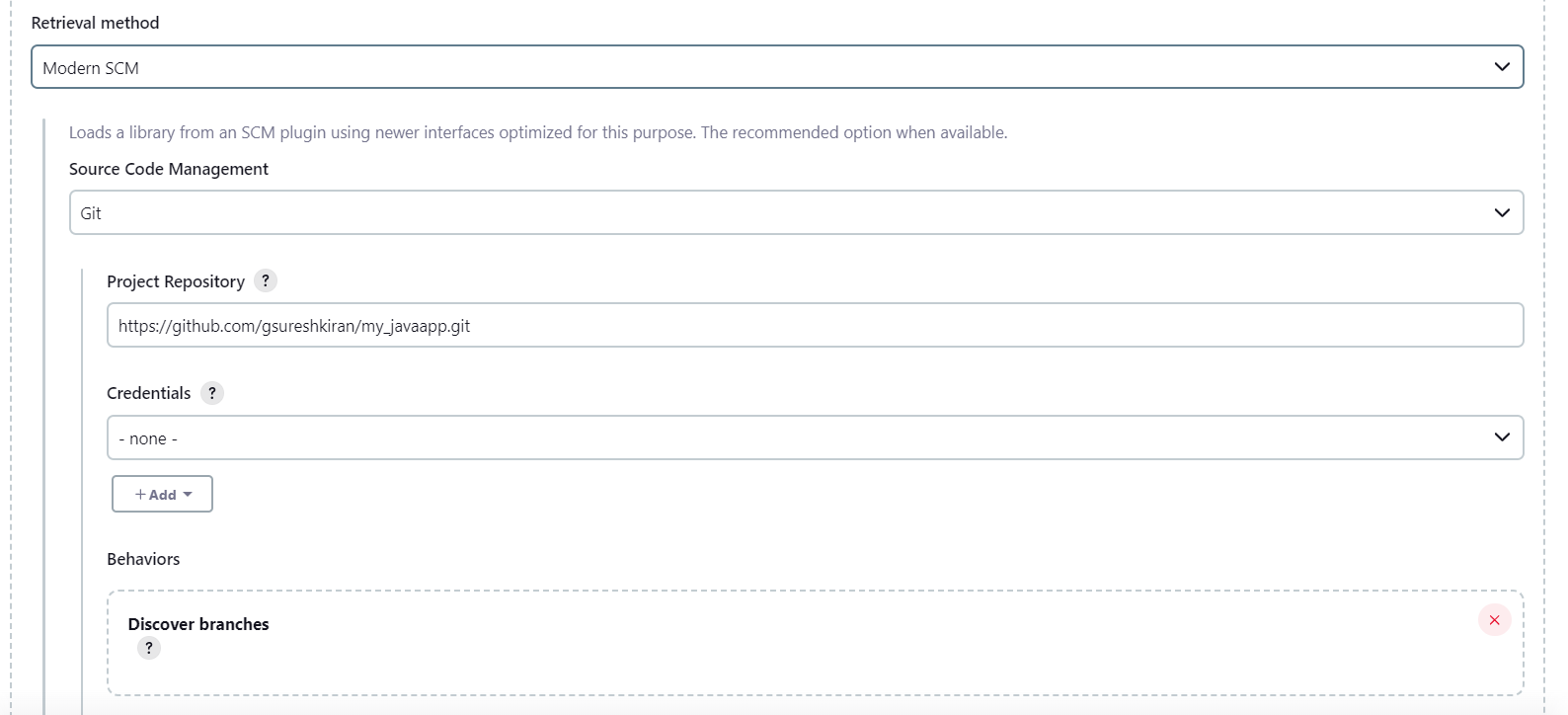
* **Cleaner Pipelines**: With the heavy lifting done in the Shared Library, your pipeline scripts can focus on orchestrating these reusable components rather than implementing detailed logic. This leads to simpler and more readable pipeline code.
* **Example**: Instead of having complex Groovy scripts in each pipeline, you can have a simple sequence of function calls to the shared library.

**Configure the shared library in jenkins:**

Login to the jenkins console , go to manage jenkins, go to system configurations. In side system configurations we have **Global Trusted Pipeline Libraries.** In thisoption enables the shared library configurations.

We need to give the some options like SCM tool like git and give the github url inside github repo you need to store all the library scripts in vars folder.

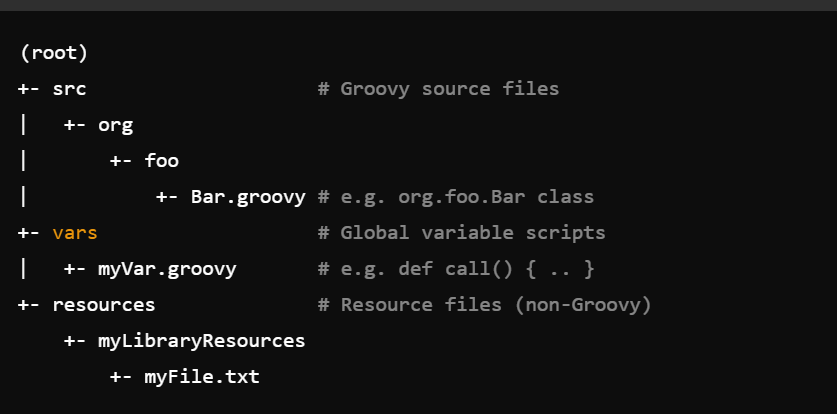




Save the changes.

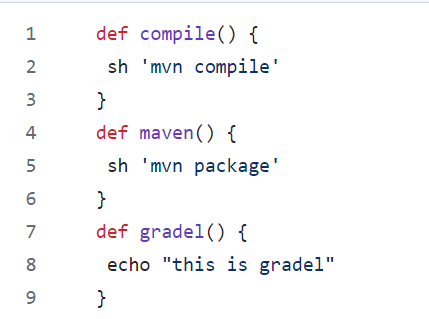
**Shared library strcture in github:**

* A Shared Library is typically stored in a source code repository (like Git) and has a specific directory structure that Jenkins recognizes.



*  **vars/**: Contains Groovy scripts that define global variables and functions. These are accessible directly in Jenkins pipelines.
* **src/**: Contains Groovy classes in a standard Java-like package structure. These are used for more complex logic or when you need to define classes and methods.
* **resources/**: Contains non-Groovy resource files, like configuration files, scripts, or templates, which can be loaded and used within the library.
* **build.gradle** or **pom.xml**: Optional files used if you need to build or package your library.

Suppose you have a function in **vars/build.groovy:**

****

You can use this function in your pipeline script:

****

The above jenkins script call the functions by using **@Library(‘my-library’) \_**

thisname configured in jenkins shared library settings in jenkins

jenkins pipeline compile stage it will calls the compile function in the vars folder. Like that we can implement the all the stages in jenkins pipeline.