Jenkins Shared Library Lab For Beginners

We are in an era of microservices where modern applications are split into individually deployable components. Jenkins with no doubt is one of the best open-source CI/CD tool for deploying microservices. As compared to a monolithic application, you will have many pipelines to deploy individual microservices.

With pipeline as code, we can code our entire CI/CD process. It is treated the same way we develop applications. You can version your pipeline code and perform all levels of testing before using it for any application deployments.

What is Jenkins Shared Library

When we say CI/CD as code, it should have modularity and reusability. And mainly it should follow the DRY principles. He is where Jenkins Shared Library comes in to play.

Jenkins Shared library is the concept of having a common pipeline code in the version control system that can be used by any number of pipelines just by referencing it. In fact, multiple teams can use the same library for their pipelines.

You can compare it with the common programming Library. In programming, we create individual libraries that can be used by anyone just by importing it into their code.

For example, if you have a ten java microservices pipelines, the maven build step will be duplicated in all the 10 pipelines. And whenever a new service gets added, you will copy and paste the pipeline code again. Also, let say you want to change some parameter in the maven build step, you will have to do it all the pipelines manually.

For the same scenario, we can write a shared library for maven build, and in all the pipelines we just have to refer the maven build library code. In the future for any maven changes, all you have to update it he shared library code. It will be applied to all the pipelines.

Getting Started With Shared Library

A shared library is a collection of groovy files (DSLs + Groovy). All the groovy files should be present it a git repo. In this example, we will be using Github as our git repo. github.com/fenago/jenkins-shared-library-framework. You can clone this repo to get the basic structure of the shared library.

Shared library repo has the following folder structure.

```
jenkins-shared-library
|____vars
|____src
|____resources
```

1. **vars:** This directory holds all the global shared library code that can be called from a pipeline. It has all the library files with a .groovy extension. It also supports .txt files for the documentation of shared library code. For example, if you have a file named maven-build.groovy, you can have a help file named maven-groovy.txt. In this file, you can write the respective shared library function help documentation in markdown format. The help file can be viewed from <your-jenkins-url>/pipeline-syntax/globals page.
2. **src:** It is a regular java source directory. It is added to the classpath during every script compilation. Here you can add custom groovy code to extend yous shared library code. Also, you can import existing Jenkins and plugins classes using an import statement. There will be scenarios where your groovy DSL's will not be flexible enough to achieve some functionality. In this case, you can write custom groovy functions in src and call it in your shared library code.
3. **resources:** All the non-groovy files required for your ur pipelines can be managed in this folder. One such example is, you might need a common JSON template to make an API call during the build. This JSON

template can be stored in the resources folder and called can be called in the shared library using the `libraryResource` function.