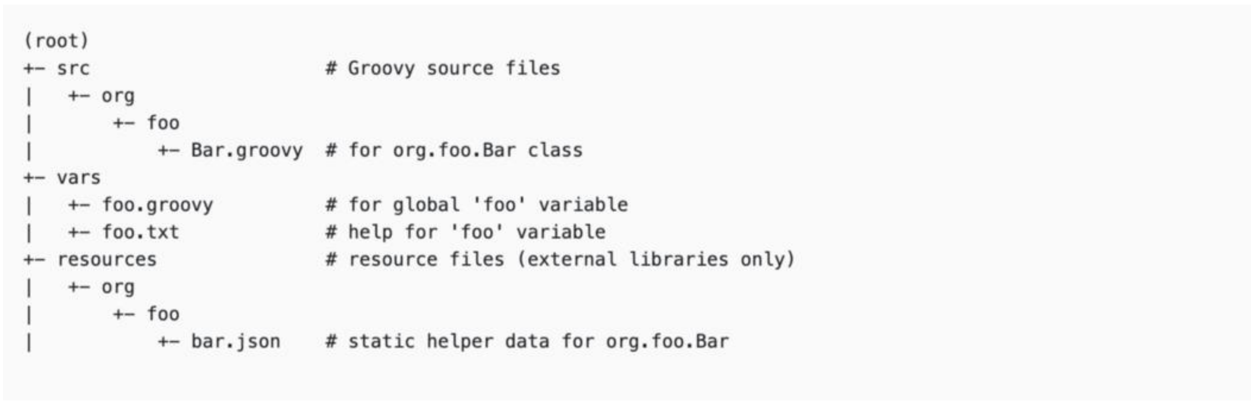
Concept of shared Libraries in Jenkins

What is shared libraries in Jenkins ? and why do we need it?

Jenkins Shared Libraries are written in Groovy and allow you to create common sets of logic, and share that among teams/projects/organizations. Instead of "copy and pasting" the code from some other Jenkinsfile, you can simply load a library in to Jenkins and every pipeline job on that Jenkins master has access to that shared library.

When writing a shared library, the code must be structured in a specific way.



```
(root)
+- src                     # Groovy source files
|   +- org
|       +- foo
|           +- Bar.groovy  # for org.foo.Bar class
+- vars
|   +- foo.groovy          # for global 'foo' variable
|   +- foo.txt             # help for 'foo' variable
+- resources               # resource files (external libraries only)
|   +- org
|       +- foo
|           +- bar.json    # static helper data for org.foo.Bar
```

The "vars" directory contains groovy scripts that, in Jenkins parlance, are called "global vars". These are the steps that you want to expose to be used in the Jenkinsfile Let’s say you want to create a shared step that handles the deployment of some code, you would add a file called “deploy.groovy” to the “vars” directory.

```
(root)
src/
    ...
vars/
    deploy.groovy
    deploy.txt
```

This would allow the deploy step to be called like

```
node() {
    stage(‘deploy’){
        deploy()  // deploy var from shared library
    }
}
```

Why you should be using Jenkins Shared Libraries in Your Pipelines

Imagine that within your team you have three projects that all deploy code in the same way. You wouldn’t want to have to write that logic three different times. This is a violation of the DRY (Don’t repeat youreslf) principal. This means that, if the process changes, then each team has to go and update their Jenkinsfile to accommodate the changes. This usually includes copying what someone else has done, and then pasting the code into their project and then making small tweaks to accommodate their project.

With a shared library, the code to deploy a project is written once and then made available to all other teams as a simple update to the version of the library. This separation of concerns will allow teams to put all of their focus on writing code, instead of worrying about how to write the code to deploy code, do automatic releases, etc, which in turn saves time and money.

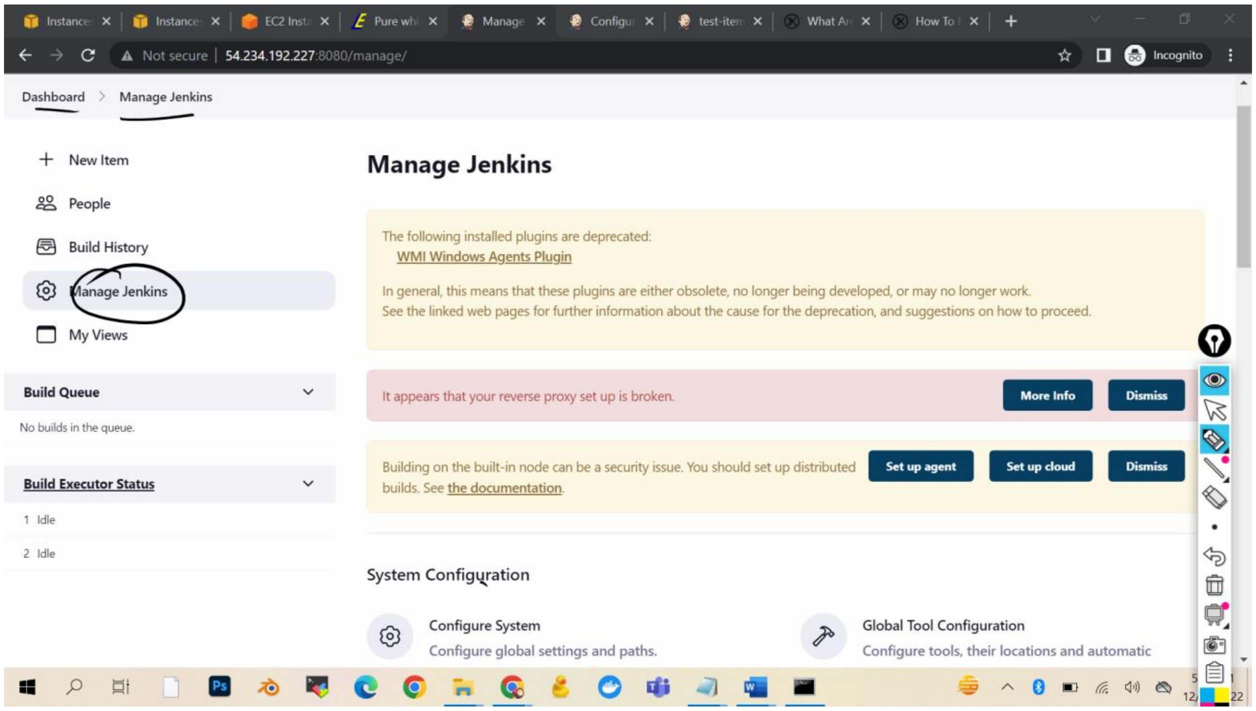
Another advantage of shared libraries is that they promote collaboration between teams. Sometimes teams can have their heads down and not be aware of what others are doing. This often leads to the same code being written multiple times. Shared libraries can bridge the gap between teams that do similar things and allow those teams to work together on a shared piece of code that can benefit them as well as any other team.

Let’s take a look at a Jenkinsfile to deploy a javascript application.

LAB :

Go to Jenkins

Manage Jenkins ->configure system



Dashboard > Manage Jenkins

No builds in the queue.

1 Idle

2 Idle

Manage Jenkins

The following installed plugins are deprecated:
WMI Windows Agents Plugin

In general, this means that these plugins are either obsolete, no longer being developed, or may no longer work.
See the linked web pages for further information about the cause for the deprecation, and suggestions on how to proceed.

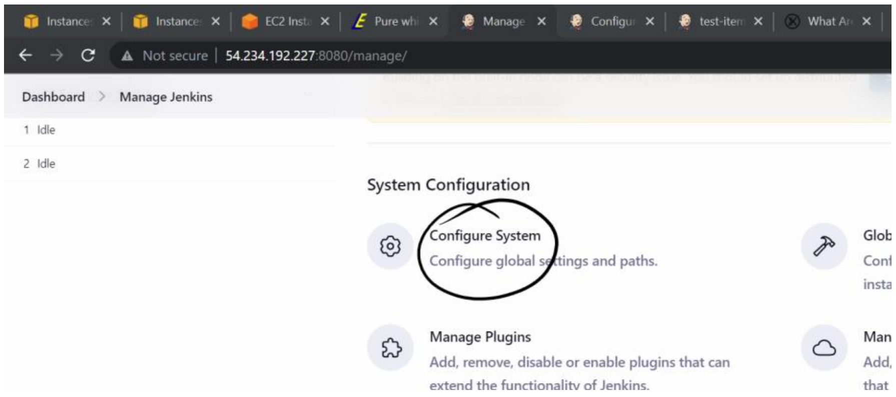
It appears that your reverse proxy set up is broken.

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

Configure System
Configure global settings and paths.

Global Tool Configuration
Configure tools, their locations and automatic



Dashboard > Manage Jenkins

1 Idle

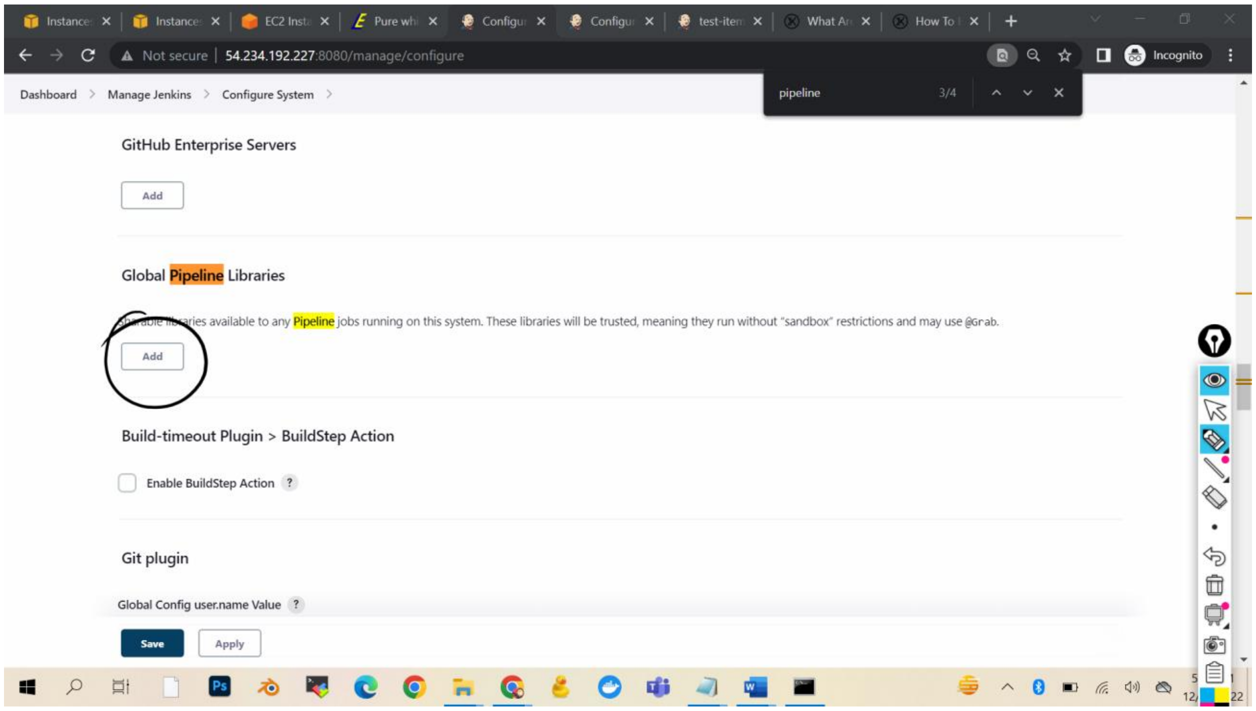
2 Idle

System Configuration

Configure System
Configure global settings and paths.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Akshat Gupta


Not secure | 54.234.192.227:8080/manage/configure
pipeline 3/4
Dashboard > Manage Jenkins > Configure System

GitHub Enterprise Servers

Add

Global Pipeline Libraries

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 @Grab.

Add

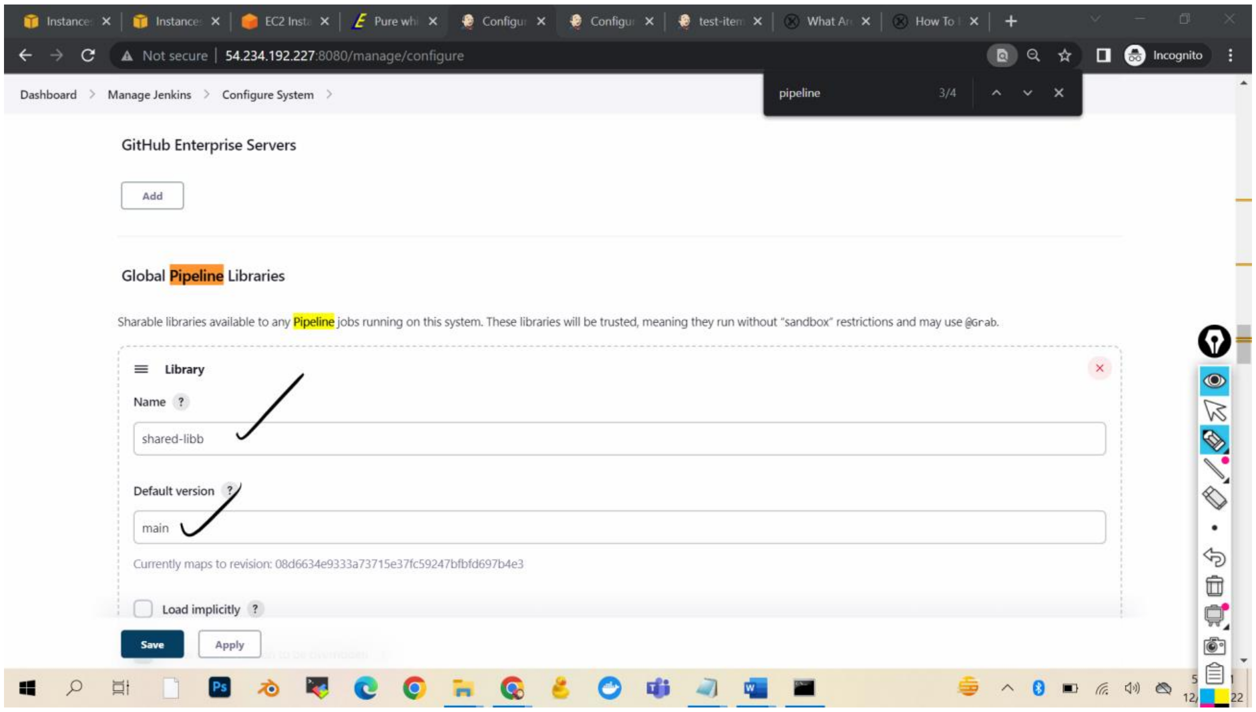
Build-timeout Plugin > BuildStep Action

Enable BuildStep Action

Git plugin

Global Config user.name Value

Save Apply


Not secure | 54.234.192.227:8080/manage/configure
pipeline 3/4
Dashboard > Manage Jenkins > Configure System

GitHub Enterprise Servers

Add

Global Pipeline Libraries

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 @Grab.

Library

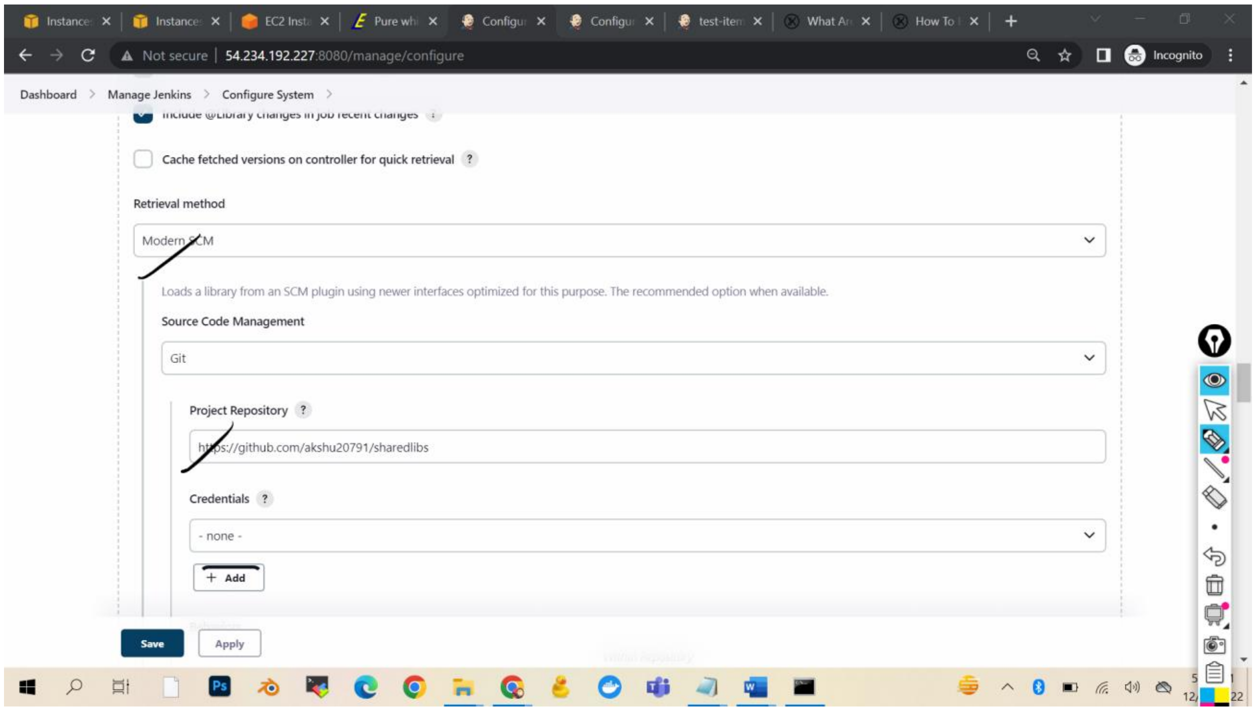
Name: shared-libb

Default version: main

Currently maps to revision: 08d6634e9333a73715e37fc59247bfbfd697b4e3

Load implicitly

Save Apply



Dashboard > Manage Jenkins > Configure System >

Include @Library changes in job recent changes

Cache fetched versions on controller for quick retrieval

Retrieval method

Modern SCM

Loads a library from an SCM plugin using newer interfaces optimized for this purpose. The recommended option when available.

Source Code Management

Git

Project Repository

https://github.com/akshu20791/sharedlibs

Credentials

- none -

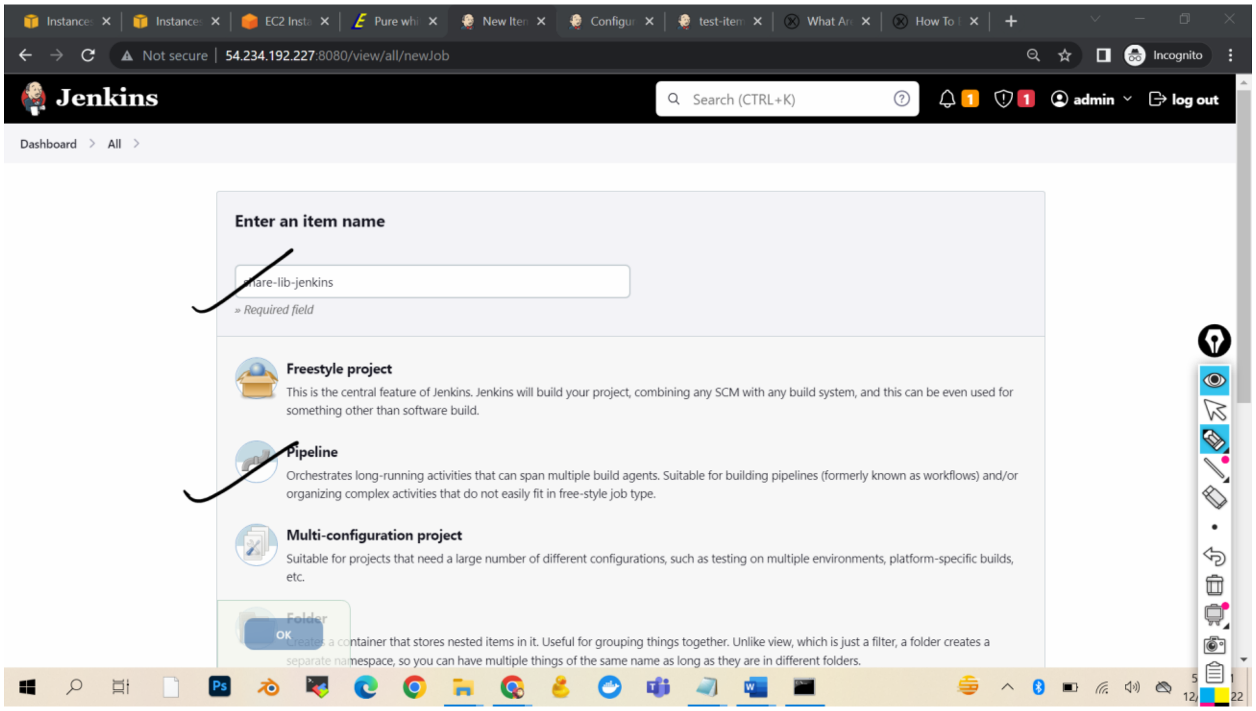
https://github.com/akshu20791/sharedlibs

save

Now go to Jenkins dashboard

New item

Akshat Gupta


Not secure | 54.234.192.227:8080/view/all/newJob

Dashboard > All >

Enter an item name

share-lib-jenkins

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

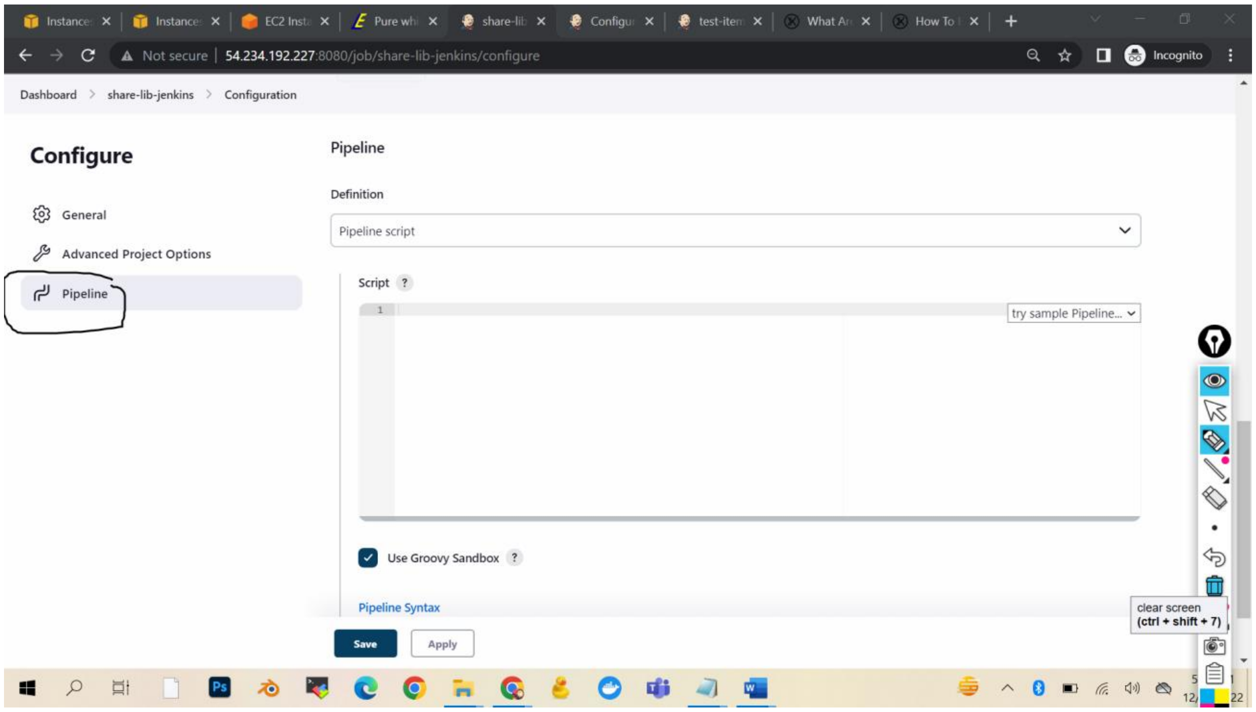
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.


Not secure | 54.234.192.227:8080/job/share-lib-jenkins/configure

Dashboard > share-lib-jenkins > Configuration

Configure

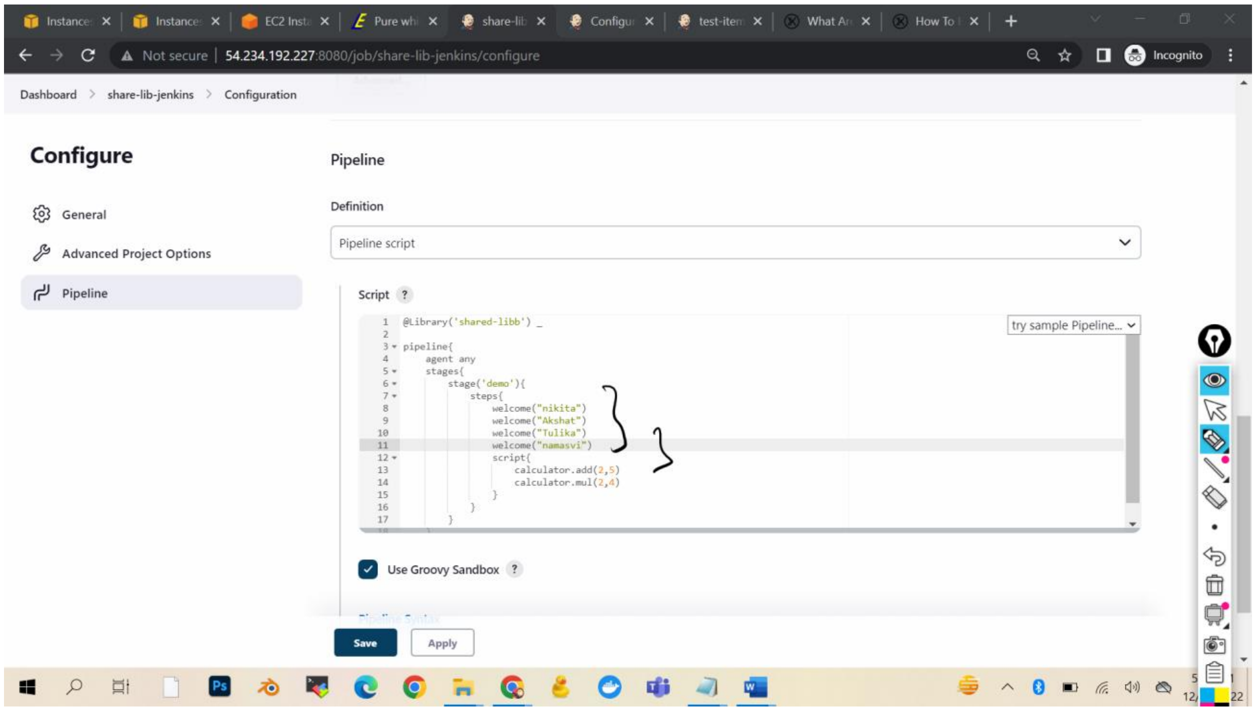
Pipeline

Definition: Pipeline script

Script

Use Groovy Sandbox ✓

Pipeline Syntax


Not secure | 54.234.192.227:8080/job/share-lib-jenkins/configure
Dashboard > share-lib-jenkins > Configuration
Configure
General
Advanced Project Options
Pipeline
try sample Pipeline...
Save
Apply

Pipeline

Definition

Pipeline script

Script

```
@Library('shared-libb') _

pipeline{
    agent any
    stages{
        stage('demo'){
            steps{
                welcome("nikita")
                welcome("Akshat")
                welcome("Tulika")
                welcome("namasvi")
                script{
                    calculator.add(2,5)
                    calculator.mul(2,4)
                }
            }
        }
```

[✓] Use Groovy Sandbox

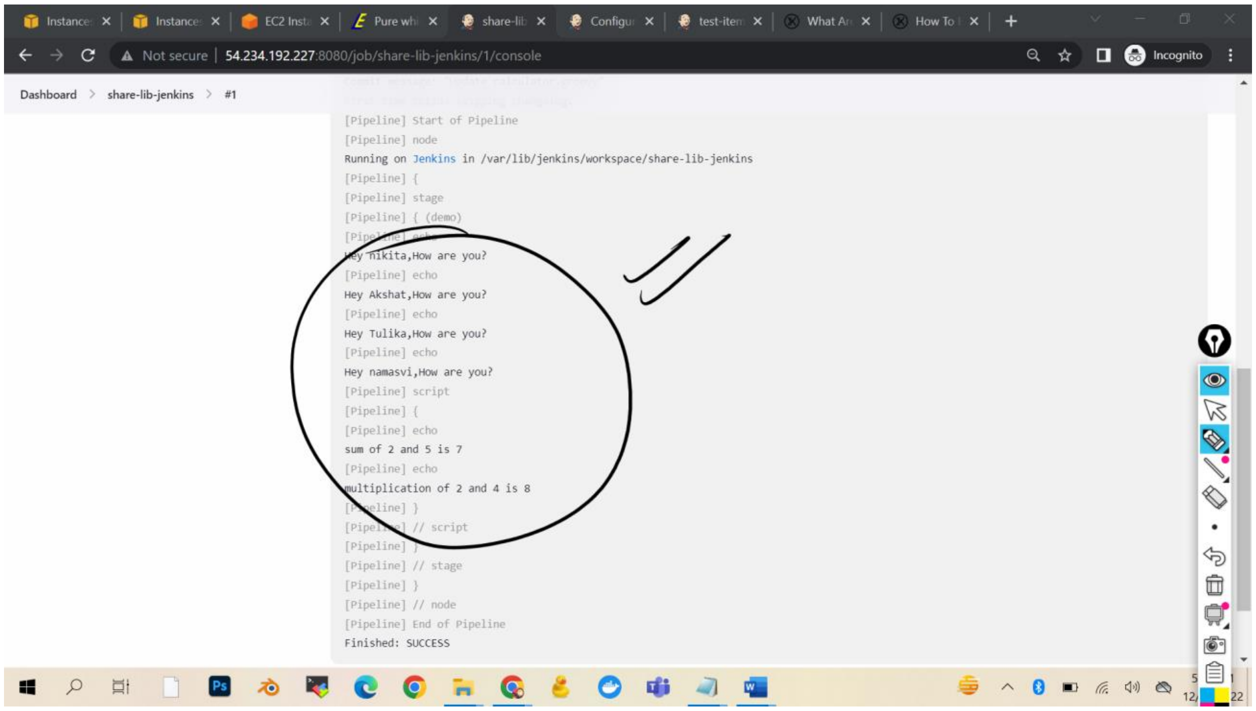
```
@Library('shared-libb') _


pipeline{
    agent any
    stages{
        stage('demo'){
            steps{
                welcome("nikita")
                welcome("Akshat")
                welcome("Tulika")
                welcome("namasvi")
                script{
                    calculator.add(2,5)
                    calculator.mul(2,4)
                }
            }
        }
    }
}
```

Save

Build now

Check console output



```
Commit message: "Update calculator.groovy"
First time build. Skipping changelog.
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/share-lib-jenkins
[Pipeline] {
[Pipeline] stage
[Pipeline] { (demo)
[Pipeline] echo
Hey nikita,How are you?
[Pipeline] echo
Hey Akshat,How are you?
[Pipeline] echo
Hey Tulika,How are you?
[Pipeline] echo
Hey namasvi,How are you?
[Pipeline] script
[Pipeline] {
[Pipeline] echo
sum of 2 and 5 is 7
[Pipeline] echo
multiplication of 2 and 4 is 8
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```