

Jenkins Shared Library



Before Jenkins 2

Manually configure job via jenkins UI
(Freestyle job)

Repository URL: `git@github.com:aleksei-bulgak/jenkins-shared-library-example.git`

Credentials: `jenkins` [Add](#)

Advanced...
[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any'): `*/master`

[Add Branch](#)

Repository browser: `gitweb`

URL: `https://github.com/aleksei-bulgak/jenkins-shared-library-example`

Additional Behaviours: [Add](#)

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GIT/Svn polling
- ☐ Poll SCM

Build Environment

- ☐ Delete workspace before build starts
- ☐ Add timestamps to the Console Output
- ☐ Use secret text(s) or file(s)

Build

Execute shell

Command: `//Compile application from sources`

[See the list of available environment variables](#)

Advanced...

Execute shell

Command: `// Run tests
// Build jar/war file
// Deploy To nexus
// and so on`

[See the list of available environment variables](#)

[Save](#) [Apply](#)





Pluses

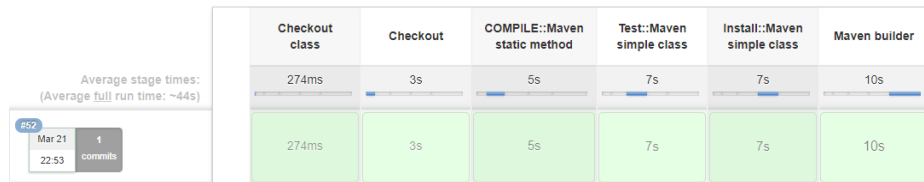
- Easy to create for everybody(dev, qa, manager)
- Easy to add changes any time
- In almost all cases each available component accessible from UI should work as described in docs

Minuses

- Hard to automate job recreation
- Bash scripting



Jenkins 2 Pipelines



```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        echo 'Building..'
      }
    }
    stage('Test') {
      steps {
        echo 'Testing..'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying....'
      }
    }
  }
}
```





Pluses

- Easy to automate job recreation
- Programmatic way of job creation
- In almost all cases each available component accessible from UI should work as described in docs

Minuses

- Scripts syntax depends on version of pipeline library
- Not a Groovy





Backoffice



Proxy service



RMS



Jobseeker jobs



One big app



Jobs



ESS

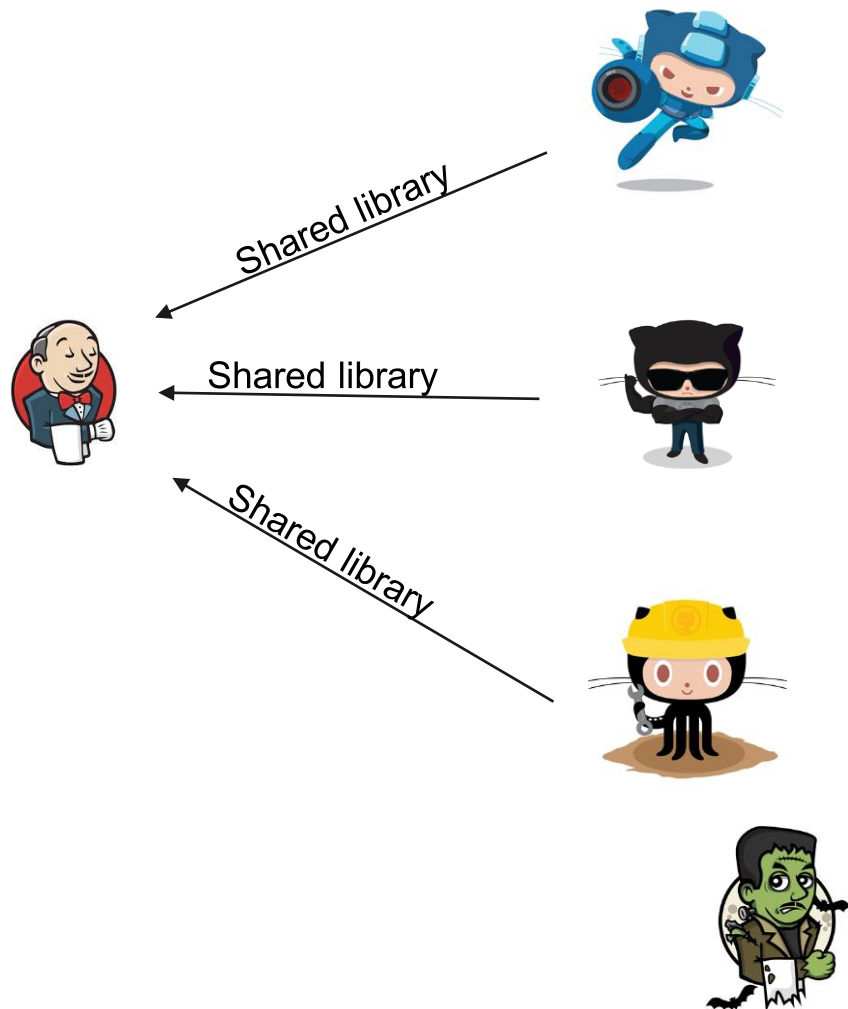


Document storage





Shared library



Structure

```
(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
```



Import library

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

my-shared-library

?

Default version

master

?

Load implicitly

☐

?

Allow default version to be overridden

☒

?

Retrieval method

Modern SCM

?

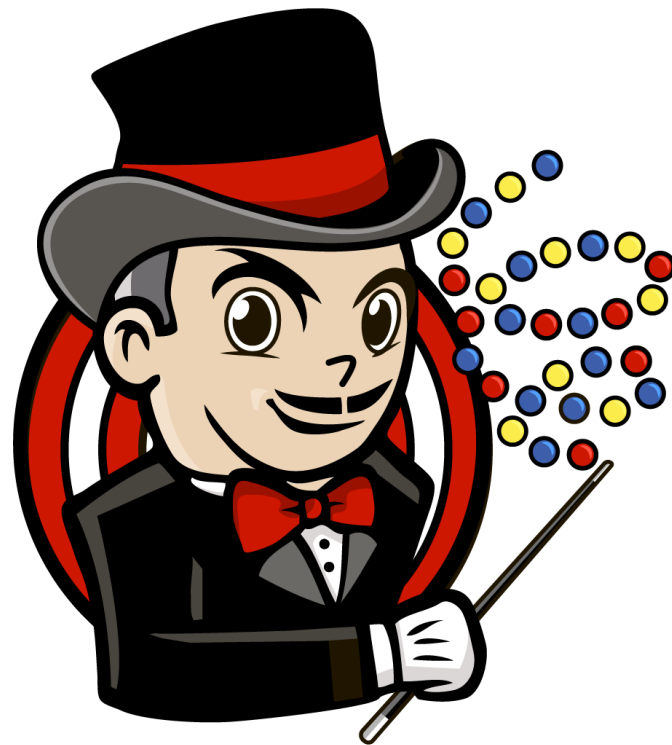
```
vars/  
@Library('my-shared-library') _  
/* Using a version specifier, such as  
branch, tag, etc */  
@Library('my-shared-library@1.0') _  
/* Accessing multiple libraries with one  
statement */  
@Library(['my-shared-library',  
'otherlib@abc1234']) _  
  
library 'my-shared-library'
```

```
src/  
До сих пор не понял как это работает  
@Library('somelib')  
import  
com.mycorp.pipeline.somelib.UsefulClass  
  
library('my-shared-  
library').com.mycorp.pipeline.Utils.someSt  
aticMethod()
```



DEMO 01/02

jenkins





@Library

- Libraries are resolved and loaded during *compilation* of the script, before it starts executing
- Use `@Library('lib-name@version')` _ if your lib contains only var functions

library

- Dynamically loaded
- You cannot `import` or otherwise “statically” refer to types from the library



Versions

- `@Library('my-shared-library')` `_` -> default version
- `@Library('my-shared-library@$BRANCH_NAME')` `_` -> \$BRANCH_NAME version
- `library "my-shared-library"` -> default version
- `library "my-shared-library@$BRANCH_NAME"` -> \$BRANCH_NAME version

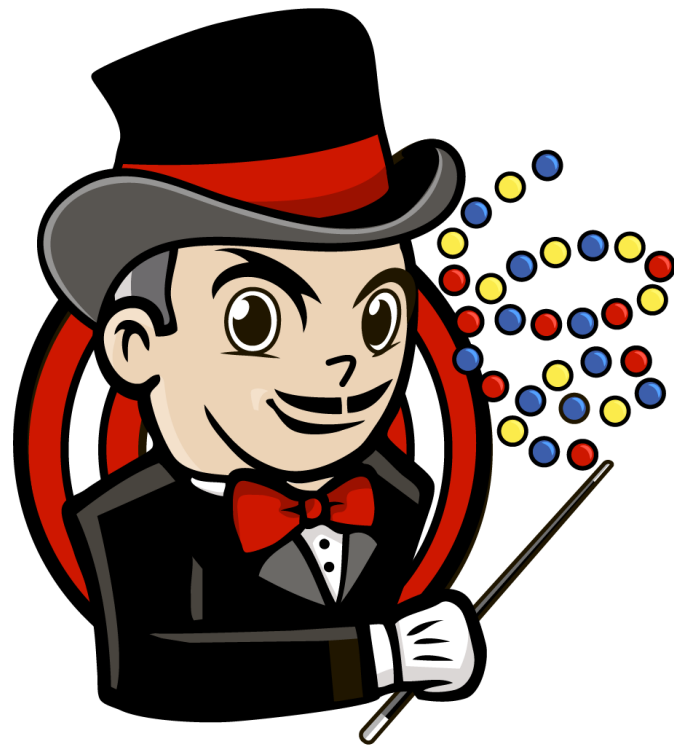
pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.

Library	
Name	<input type="text" value="static-shared-lib"/>
<u>Default version</u>	<input type="text" value="lib"/> ←
	Currently maps to revision: 5a21c737c883f2088e6c44420e45dec8f06fa231
Load implicitly	<input type="checkbox"/>
Allow default version to be overridden	<input checked="" type="checkbox"/>
Include @Library changes in job recent changes	<input checked="" type="checkbox"/>
Retrieval method	
<input checked="" type="radio"/> Modern SCM	
Source Code Management	
<input checked="" type="radio"/> Git	
Project Repository	<input type="text" value="git@github.com:aleksei-bulgak/jenkins-shared-library-example.git"/>
Credentials	<input type="text" value="jenkins"/> <input type="button" value="Add"/>



DEMO 03

jenkins



Src/

```
package org.foo
class Utilities implements Serializable {
  def steps
  Utilities(steps) {this.steps = steps}
  def mvn(args) {
    steps.sh "${steps.tool 'Maven'}/bin/mvn -o ${args}"
  }
}
```

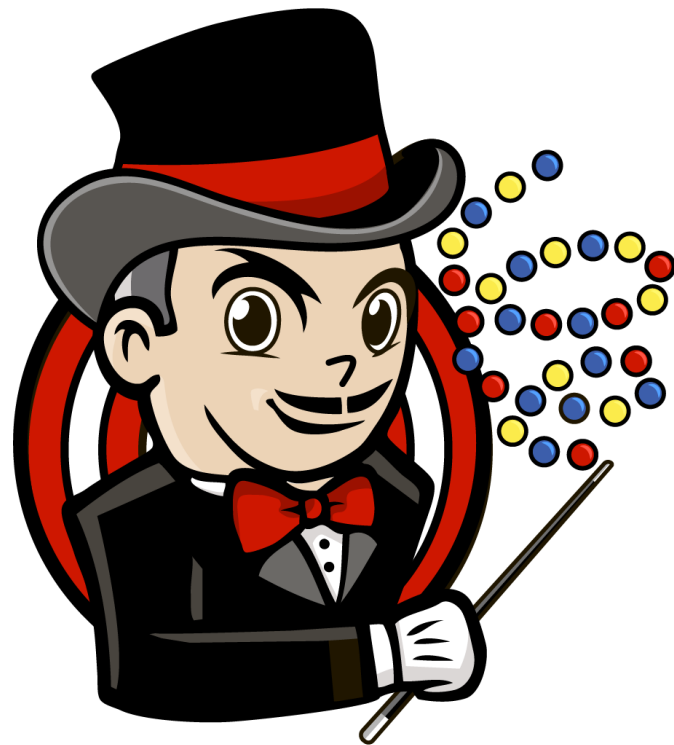
```
def utils = library('utils')org.foo.Utilities
node {
  utils.new(this).mvn('clean package')
}
```

```
package org.foo
def mvn(args) {
  sh "${tool 'Maven'}/bin/mvn -o ${args}"
}
```

```
def utils = library('utils')org.foo.Utilities
node {
  utils.new().mvn('clean package')
}
```



DEMO 04





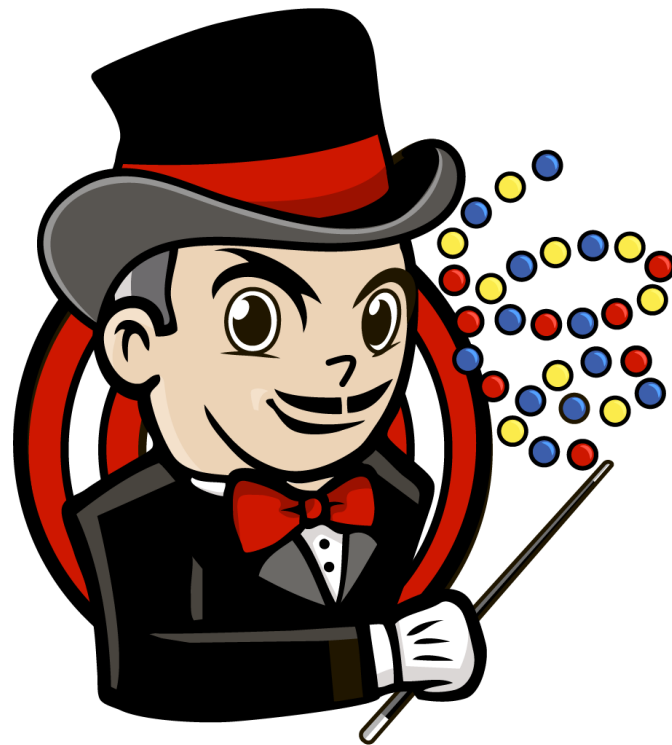
vars/

```
vars/log.groovy  
def info(message) {  
    echo "INFO: ${message}"  
}  
  
def warning(message) {  
    echo "WARNING: ${message}"  
}
```

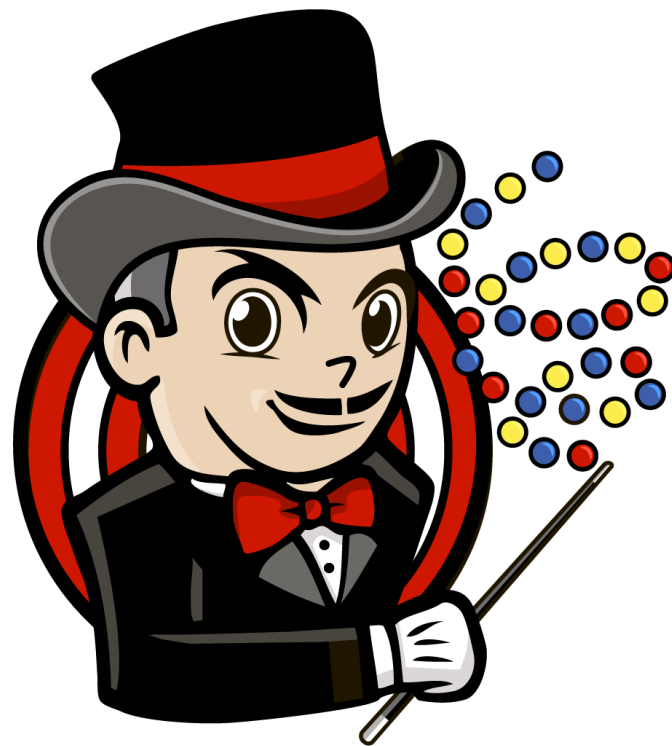
Multiple functions in one file



DEMO 05



DEMO 06





Conclusion

- Not a silver bullet
- Some features works strange
- Hard to debug if you do not have groovy sdk
- You can cover such libs with tests(even with integration tests)
- This is not actually a Groovy and as a result we do not have access to some cool Groovy features
- But you can inject external libs and code via `@Grab('groupId:artifactId:version')`





References

<https://jenkins.io/doc/book/pipeline/shared-libraries/>

<https://github.com/fabric8io/fabric8-jenkinsfile-library>

<https://github.com/fabric8io/fabric8-pipeline-library>

