

# Open Source Examples of Unit Tests fora Jenkins Shared LibraryLisa Ranjbar, Emilio Reyes,Ernesto Ojeda



# Introduction

DEVOPS DEVOPS

## About EdgeX Foundry

- An open source, vendor neutral project (and ecosystem)
- A microservice, loosely coupled software framework for IoT edge computing
- Hardware and OS agnostic
- Started in 2017

# $E D G E \bigotimes F O U N D R Y^{\mathsf{M}}$

**ILFEDGE** 

https://www.edgexfoundry.org/



## About edgex-global-pipelines

- Opinionated Jenkins Shared Library for EdgeXFoundry
- Backbone of the Jenkins pipelines for EdgeXFoundry
- Open Source
- Active Development

https://github.com/edgexfoundry/edgex-global-pipelines



## Why Unit Test Your Jenkins Shared Library?

- Jenkins Shared Libraries can have global scope
- Functions should be reused in a large number of Jenkins jobs
- Without automated tests you still have to regression test!
- Having automated tests will help you find regressions faster.
- Enables faster development of features for your Jenkins Shared Library



## Starting point for edgex-global-pipelines

- Initial code base had limited test automation
- Continuously building new features
- Delivering a simple interface for developers to use in Jenkinsfiles
- Required extensive regression testing to facilitate future iterations
- We needed something to help us build Continuous Confidence

Example Jenkinsfile:

edgeXBuildGoApp ( project: 'device-modbus-go', goVersion: '1.13', buildSnap: true



## Jenkins-Spock Framework

- Extension of the Spock Framework for Jenkins Pipelines
- Supports common testing flow: setup, expect, when, then, etc
- Facilitates automatic mocking of core Jenkins steps and Jenkins Plugins steps
- Supports Jenkins Shared Libraries
- Groovy based

https://github.com/ExpediaGroup/jenkins-spock



## Jenkins-Spock Framework cont.

- loadPipelineScriptForTest()
- getPipelineMock()
  - Stub out interactions between functions
- getBinding()

DEVOPSWORLD

by CloudBees

- We use this to set environment variables
- explicitlyMockPipelineVariable()
  - Mock another custom pipeline function in your library
- explicitlyMockPipelineStep()
  - Mock Pipeline step from a Jenkins plugin

```
def setup() {
    edgeXDockerLogin = loadPipelineScriptForTest('vars/
    edgeXDockerLogin.groovy')
def "Test call [Should] raise error [When] config has no
settingsFile" () {
    setup:
   when:
        edgeXDockerLogin()
    then:
        1 * getPipelineMock('error').call('Project Settings
        File id (settingsFile) is required for the docker
        login script.')
```

# **Mocking Best Practices**

DEVOPS DEVOPS

## **Mock Pipeline Plugins**

- Allow Jenkins-Spock to automatically mock Pipeline Steps
  - sshagent(credentials: [...])
    - Error: java.lang.lllegalStateException: There is no pipeline step mock for [sshagent]
  - Always add Jenkins Plugins as dependencies to Maven/Gradle project
    - Get Plugin ID for Step in Pipeline Steps: <u>https://www.jenkins.io/doc/pipeline/steps/</u>
    - Search for Plugin ID in Maven Repository: <a href="https://mvnrepository.com/">https://mvnrepository.com/</a>
    - Add Plugin's JAR as a project dependency
      - org.jenkins-ci.plugins:ssh-agent:1.17@jar
  - Eliminate the need of having to call explicitlyMockPipelineStep()

https://github.com/edgexfoundry/edgex-global-pipelines/blob/master/build.gradle



## **Mock Environment Variables**

#### Enforce consistent pattern for getting and setting environment variables

- Use env. VARIABLE to get value
- Use env.VARIABLE = VALUE to set value avoid using env.setProperty()
- Always fully qualify using env.
- Facilitate mocking by eliminating unnecessary mocks
- Environment variables are script variables
  - Must be set using getBinding().setVariable()
  - Env variables should be specified as a Groovy map

https://github.com/edgexfoundry/edgex-global-pipelines/blob/master/vars/edgeXSemver.groovy



### **Mock Environment Variables - Example**

#### Source

```
53 def setenv() {
54 env.SHORT_GIT_COMMIT = "${env.GIT_COMMIT}".substring(0, 7)
55 env.SEMVER_DEBUG = 'on'
56 }
```

**DEVOPSWORLD** 

by CloudBees

Unit Test

```
12
         def "Test edgeXPatterns setenv [Should] set expected env vars [When] called" () {
13
             setup:
14
                 def envVars = ['GIT_COMMIT': '0fe050cff581bd7866a842f45ab0666d7601b761']
                 edgeXPatterns.getBinding().setVariable('env', envVars)
15
16
             when:
17
                 edgeXPatterns.setenv()
18
             then:
                 envVars['SEMVER DEBUG'] == 'on'
19
                 envVars['SHORT_GIT_COMMIT'] == '0fe050c'
20
                 envVars['GIT_COMMIT'] == '0fe050cff581bd7866a842f45ab0666d7601b761'
21
22
```

## **Mock Nested Objects**

- Nested objects require explicit mocking
  - docker.image('...').inside()
    - Error: java.lang.NullPointerException: Cannot invoke method inside() on null object
  - Mock using explicitlyMockPipelineVariable()
  - Expect and stub interactions using getPipelineMock()

https://github.com/edgexfoundry/edgex-global-pipelines/blob/master/src/test/groovy/edgeXNexusPublishSpec.groovy



## Mock Nested Objects - Example

**DEVOPSWORLD** 

by CloudBees

#### Source

18	def	gitSemver() {			
19		<pre>def imageName = env.ARCH &amp;&amp; env.ARCH == 'arm64'</pre>			
20		<pre>? "\${env.REPO}/semver-arm64:latest"</pre>			
21		: "\${env.REPO}/semver:latest"			
22		<pre>ker.image(imageName).inside {</pre>			
23		sh 'git semver'			
24		}			
25	}	Unit Test			
37		<pre>def "Test edgeXPatterns gitSemver [Should] call expected [When] called" () {</pre>			
38		setup:			
39		<pre>edgeXPatterns.getBinding().setVariable('env', ['REPO': 'nexus1'])</pre>			
40		when:			
41		<pre>edgeXPatterns.gitSemver()</pre>			
42		then:			
43		1 * getPipelineMock('docker.image').call('nexus1/semver:latest') >>			
44		explicitlyMockPipelineVariable()			
45		1 * getPipelineMock('sh').call('git semver')			
46		}			

## **Mock External Method Calls**

#### • Method in one script calls method in another

- Shared Library may consist of multiple scripts
- Helper methods can be centrally located in a utils groovy script for DRY reasons
  - Error: java.lang.lllegalStateException: There is no pipeline variable mock for [utils].
- Pipeline libraries and scripts are also variables
- Mock using explicitlyMockPipelineVariable()
- Expect and stub interactions using getPipelineMock()







## Mock External Methods - Example

#### Source

```
def extcall() {
66
         docker.image('semver:latest').inside() {
67
             if(utils.isDryRun()) {
68
69
                 echo('sh git semver')
70
             } else {
                  sh 'git semver'
71
72
73
74
                                     Unit Test
         def "Test edgeXPatterns excall [Should] call expected [When] DryRun true" () {
86
87
             setup:
                 explicitlyMockPipelineVariable('utils')
88
                 getPipelineMock('utils.isDryRun').call() >> true
89
                 getPipelineMock('docker.image').call('semver:latest') >> explicitlyMockPipelineVariable()
90
91
             when:
92
                 edgeXPatterns.extcall()
93
             then:
                 1 * getPipelineMock('echo').call('sh git semver')
94
95
```

```
by CloudBees
```

## **Mock Internal Method Calls**

- Method in script calls another method in same script
  - Deliberate effort to develop small functionally cohesive methods
    - Isolate method under test
  - Mocking method call is complicated and doesn't scale
    - Unable to mock method in same script
- Workaround
  - Create call graph for all methods in scriptA
  - Create another script: scriptAU
  - Methods in odd-numbered layers stay in **scriptA**, even-numbered layers move to **scriptAU**
  - Use external method call mocking techniques

https://github.com/edgexfoundry/edgex-global-pipelines/blob/master/src/test/groovy/edgeXReleaseGitTagSpec.groovy





# Unit Testing Setup



## Running the tests... Getting Started

- Initial implementation was based off the jenkins-spock shared-library example
- Tests originally run with Maven
- pom.xml...so much xml
- Issues with dockerized builds
- Lacking features for fast local development



## Running the tests... Migrating to Gradle

- Overall speed improvements
- Better developer experience
- Gradle offers a lot of flexibility / extensibility
- Plugin support to get code coverage (potentially build our own)
- Unit test summary report

A good Maven vs Gradle comparison can be found here: <u>https://gradle.org/maven-vs-gradle/</u>



## Running the tests... Speed improvements

#### • Before and After Migration

- Maven: mvn clean test
  - 103 tests, 5 minutes on average to run
- Gradle: gradle clean test
  - 103 tests, 2 minutes on average to run
- Today
  - 188 total tests, 3 minutes on average to run
  - Use Gradle's --parallel option with forking on Test tasks



# CI/CD for the Pipeline Library



## Use of Docker for running unit tests

- Docker is used for running Gradle tasks
- Optimizations
  - Vanilla Gradle docker images can be used i.e.
     gradle:6.5
  - Recommended approach: Bundle dependencies in custom base docker image
  - Can save 1.5 2m downloading individual dependencies
  - docker pull takes about 16-20s for 602MB image





## **Developer Experience**



## **IDE Integration**

- Nice integrations for IDE, Eclipse/IntelliJ, etc.
- Run specific tests
- Run only failed tests
- Debug tests and step through code

https://www.jetbrains.com/idea/





#### © 2020 All Rights Reserved

**DEVOPSWORLD** 

by CloudBees

	• •	edgex-global-	pipelines [~/Intel/code/Edgex/edgex-global-pipelines]/src/test/groovy/edgeXDockerSpec.groovy [edgex-global-pipelines.test]				
📭 edgex-global-pipelines ) 🖿 src ) 📭 test ) 🖿 groovy ) 📴 EdgeXDockerSpec 💌 🕨 🤞 Cit: 🖌 🗸 🔘 🖒 🖿 🖸							
ect	🔲 Project 👻 😌 😤 🗢 -						
Proj	► ■ docs	/1	DUCKER_REGISTRY': 'MyUOCKERKEGISTRY'				
÷,	Image: Second						
- 1	▶ ■ gradle		getPipelineMock(_pipeline_extension: 'docker.image')('MyOockerImageName') >> explicitlyMockPipelineVariable(_variable_na	ame: 'DockerImageMock')			
	resources		when:				
	scripts		edgeXDocker.push('MyDockerImageName')	=			
	▼ 🖿 src		then:	=			
	🔻 📑 test		1 * getPipelineMock( _pipeline_extension: 'Docker[mageMock.push'),cgil('MyGitCommit')	=			
	🔻 🖿 groovy		1 * getripe inemock _ pipeline _extension: 'DockerImageMock, push'), catist ')				
	edgeXBuildCAppSpec.groovy		1 * getFipeLineMock( _pipeline_stemation: 'DockerImageNock.push').call('MyGitCommit=MyVersion')				
	edgeXBuildDockerSpec.groovy		1 * getPipelineMock(pipeline_extension: 'DockerImageMock.push').call('MyVersion')				
	edgeXBuildGoAppSpec.groovy		<pre>2 * getPipelineMock( _pipeline_extension: 'DockerImageMock.push').call('MyDockerCustomTags')</pre>				
	edgeXBuildGoModSpec.groovy		1 * getPipelineMock( _pipeline_extension: 'docker.withRegistry').call(_) >> { _arguments ->				
	edgeXBuildGoParallelSpec.groovy		<pre>assert 'https://MyDockerRegistry:10004' == _arguments10101</pre>	Ξ.			
	edgeXClairSpec.groovy		}	=			
	edgeXCodecovSpec.groovy						
	edgeXDockerLoginSpec.groovy		"Test push [Should] call push to correct registry and port [When] nexusRepo is snapshot" () {				
		90	setup:	_			
		91	def environmentVariables = [				
		92	'GIT_COMMIT': 'MyGitCommit',	=			
	edgeXinfraEr rootsorghopee.groovy		DUCKER_REGISTAT = MyDUCKETREGISTTY	_			
	edgeXinfraBhipLogSpec.groovy						
		EdgeX	) DockerSpec 💚 Test push [Should] call image push with expected arguments [When] VERSION SEMVER_BRANCH DOCKER_CUSTOM_TAGS()				
1	Run: 🥏 EdgeXDockerSpec 🔀						
	▶ ▲ ◎ 15 12 Ξ 포 ↓ ↑ ♣ ◎ ┖	₽ \$	8 Tests failed: 2, passed: 27, ignored: 1 of 30 tests – 17 s 198 ms				
1	👂 🔻 😢 Test Results						
-	EdgeXDockerSpec	17 s 198 ms	Too few invocations for:				
	Test build [Should] call docker build	with exp2s142ms	2 +	1 lait cho-th/Cit/Connit! labol labol-th/A.			
	Test build [Should] call docker build	with expec 933 ms	2 * getripetinemock( docker.ouild ).catt([ myuockerimagevame , myuockeriteratinouild-arg akth=myarthtabe				
	Units and the second	with expect 703 ms	Unmatched invocations (ordered by similarity):	<u></u>			
	Test push [Should] call push to correct registry 490 ms						
	✓ Test push [Should] call push to corr	ect registry 602 ms	1 * getPipelineMock("sh").call('sudo chown -R jenkins:jenkins.')				
an '	Iest push [Should] call push to corr at Test push All [Should] call push to corr	ect registry 778 ms	instance == target				
truct	Test pushAll [Should] call push to c	orrect regis 694 ms					
Z: S	Test pushAll [Should] call push to c	orrect regist 517 ms	false				
•	<ul> <li>Test finalimageName [Should] return</li> </ul>	n expected 542 ms					
s	Test finalImageName [Should] return	n expecte1 s 107 ms	Mock for type 'Closure' named '()getPipelineMock("(sh)")')'				
orite	✓ Test finalImageName [Should] return	n expected 759 ms	Mock for type 'Closure' named '((implicit-expected) )getPipelineMock("(docker.build)")'				
Favo	✓ Test cleanImageUrl [Should] return	no http:// oi 645 ms	Mock for type 'Closure' named 'getPipelineMock("sh")'				
10	✓ Test parse [Should] return expected	<b>[When] ca</b> l 560 ms	One or more arguments(s) didn't match:				
*							
	▶ 4: Run III 🗄 5: TODO 🗜 9: Version Control 🖾 Terminal						
	Tests failed: 2, passed: 27, ignored: 1 (moments ago)			83:14 LF UTF-8 4 spaces Git: fix-arm64-semver 🍗 👮			









# Conclusion

DEVOPS DEVOPS WORLD by CloudBees

## Conclusion

- Jenkins-spock makes it easy to mock your pipeline steps
- Build continuous confidence in your shared pipeline libraries
- Ease of use and increased build velocity with Gradle
- IDE integration makes for a great development experience

https://github.com/edgexfoundry/edgex-global-pipelines

