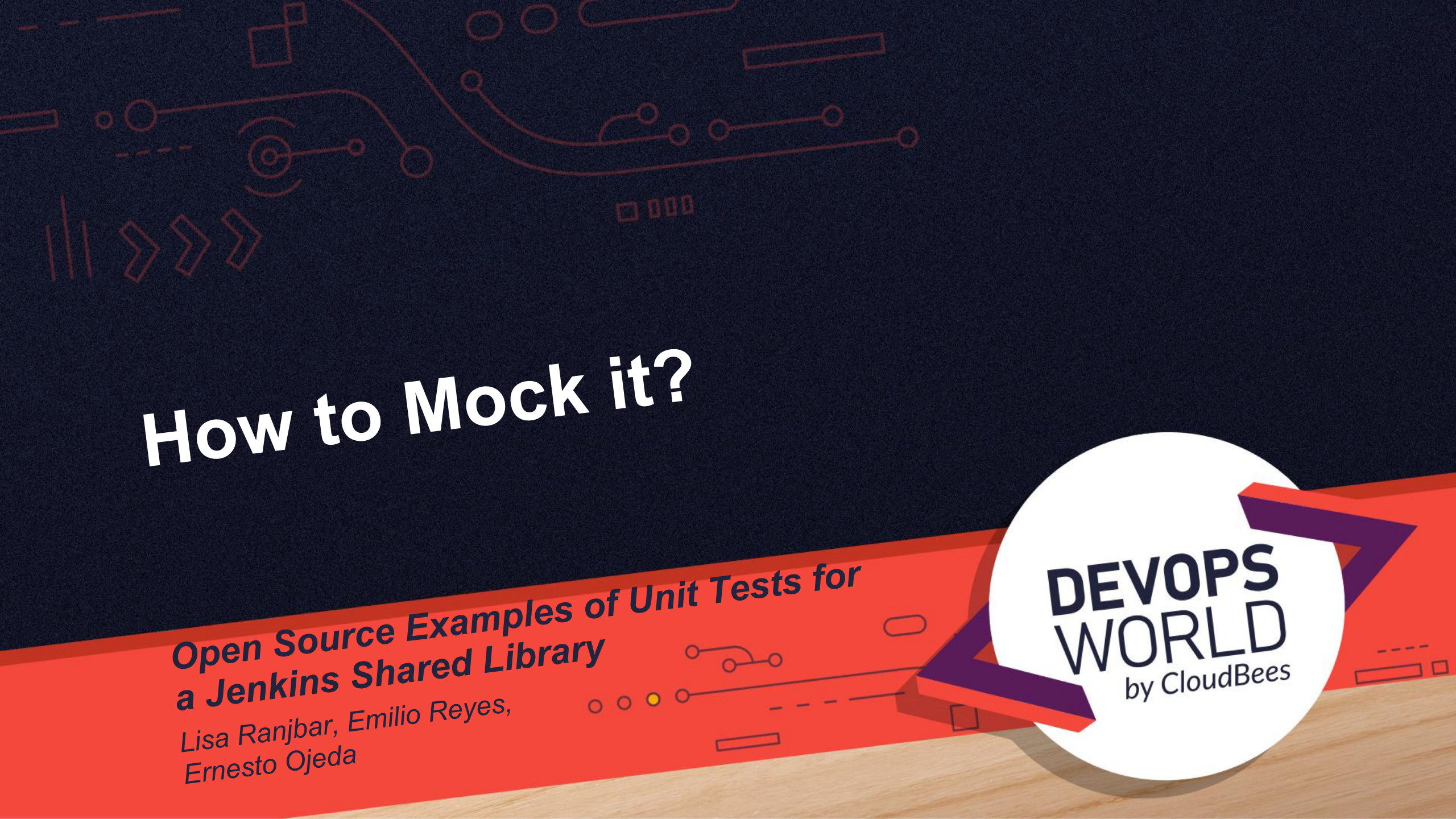
How to Mock it?

Open Source Examples of Unit Tests for a Jenkins Shared Library

Lisa Ranjbar, Emilio Reyes, Ernesto Ojeda



Introduction

DEVOPS WORLD
by CloudBees

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

EDGE X FOUNDRY™

LF EDGE

https://www.edgexfoundry.org/

© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

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
)
```


© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

Jenkins-Spock Framework cont.

- loadPipelineScriptForTest()
- getPipelineMock()
  - Stub out interactions between functions
- getBinding()
  - 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.')
}
```


© 2020 All Rights Reserved.

Mocking Best Practices

DEVOPS WORLD
by CloudBees

Mock Pipeline Plugins

- Allow Jenkins-Spock to automatically mock Pipeline Steps
  - **sshagent(credentials: [...])**
    - Error: java.lang.IllegalStateException: 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: https://www.jenkins.io/doc/pipeline/steps/
    - Search for Plugin ID in Maven Repository: https://mvnrepository.com/
    - 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

© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

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  }
```

Unit Test

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


© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

Mock Nested Objects - Example

Source

```
18  def gitSemver() {
19      def imageName = env.ARCH && env.ARCH == 'arm64'
20          ? "${env.REPO}/semver-arm64:latest"
21          : "${env.REPO}/semver:latest"
22      docker.image(imageName).inside {
23          sh 'git semver'
24      }
25  }
```

Unit Test

```
37      def "Test edgeXPatterns gitSemver [Should] call expected [When] called" () {
38          setup:
39              edgeXPatterns.getBinding().setVariable('env', ['REPO': 'nexus1'])
40          when:
41              edgeXPatterns.gitSemver()
42          then:
43              1 * getPipelineMock('docker.image').call('nexus1/semver:latest') >>
44                      explicitlyMockPipelineVariable()
45              1 * getPipelineMock('sh').call('git semver')
46      }
```


© 2020 All Rights Reserved.

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.IllegalStateException: There is no pipeline variable mock for [utils].
  - Pipeline libraries and scripts are also variables
  - Mock using `explicitlyMockPipelineVariable()`
  - Expect and stub interactions using `getPipelineMock()`

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

© 2020 All Rights Reserved.

Mock External Methods - Example

Source

```
66  def extcall() {
67      docker.image('semver:latest').inside() {
68          if(utils.isDryRun()) {
69              echo('sh git semver')
70          } else {
71              sh 'git semver'
72          }
73      }
74  }
```

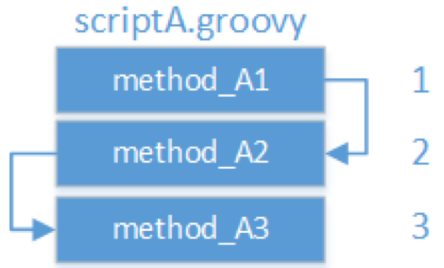
Unit Test

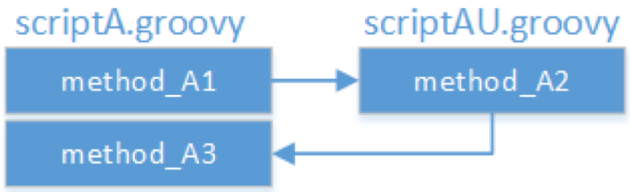
```
86      def "Test edgeXPatterns excall [Should] call expected [When] DryRun true" () {
87          setup:
88              explicitlyMockPipelineVariable('utils')
89              getPipelineMock('utils.isDryRun').call() >> true
90              getPipelineMock('docker.image').call('semver:latest') >> explicitlyMockPipelineVariable()
91          when:
92              edgeXPatterns.extcall()
93          then:
94              1 * getPipelineMock('echo').call('sh git semver')
95      }
```


© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

Unit Testing Setup

Jenkins-Spock, Gradle, etc...

DEVOPS WORLD
by CloudBees

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

© 2020 All Rights Reserved.

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: https://gradle.org/maven-vs-gradle/

© 2020 All Rights Reserved.

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

© 2020 All Rights Reserved.

CI/CD for the Pipeline Library

Best practices for running on Jenkins

DEVOPS WORLD
by CloudBees

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

```
...

stage('Test') {
  agent {
    docker {
      image "edgex-devops/egp-unit-test:gradle"
      args ...
          }
  }
  steps {
          sh 'gradle clean test --parallel'
     ...
  }
}
```

see Jenkinsfile...

© 2020 All Rights Reserved.

Developer Experience

IDE Integration

DEVOPS WORLD
by CloudBees

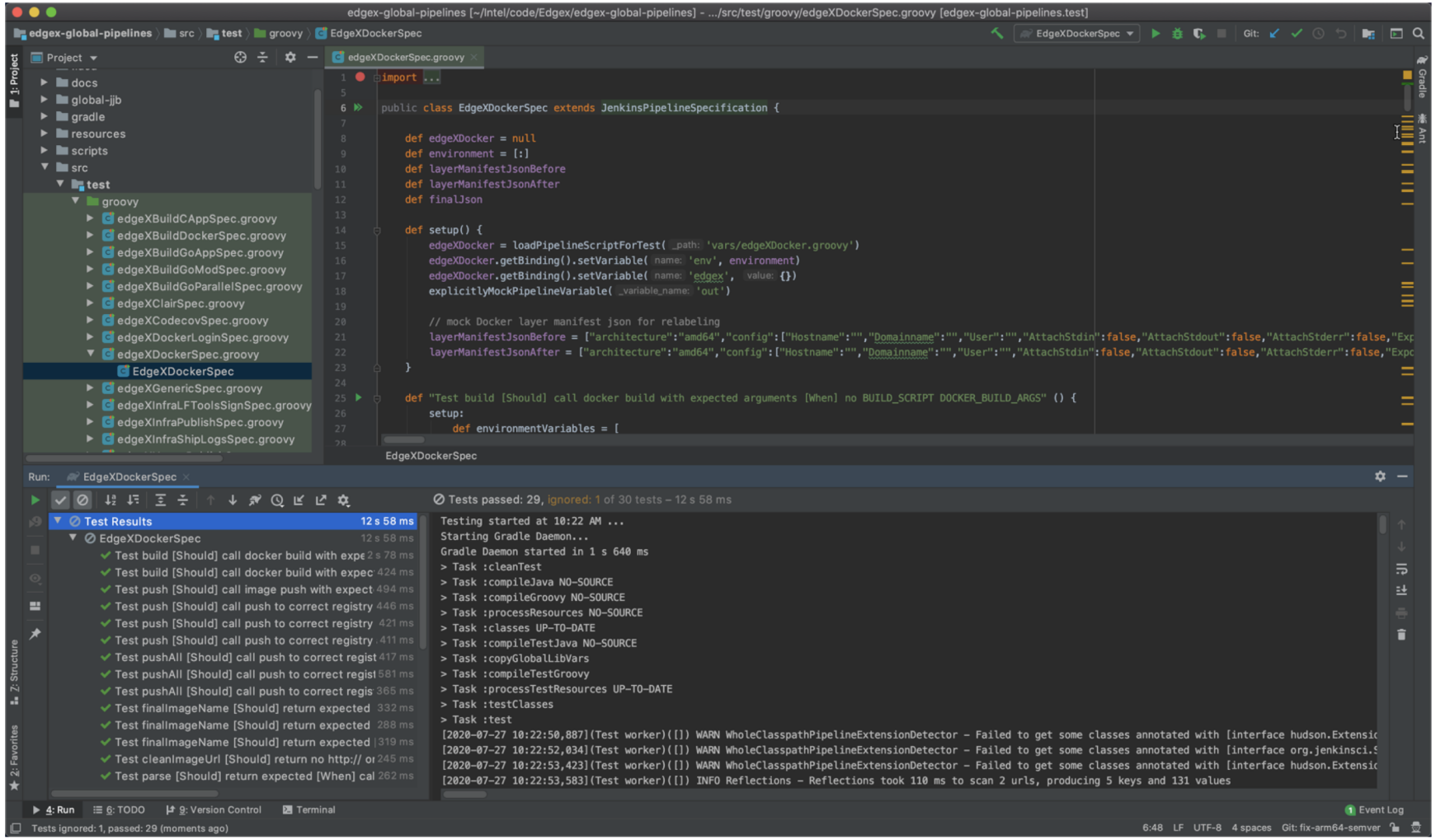
IDE Integration

- 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/

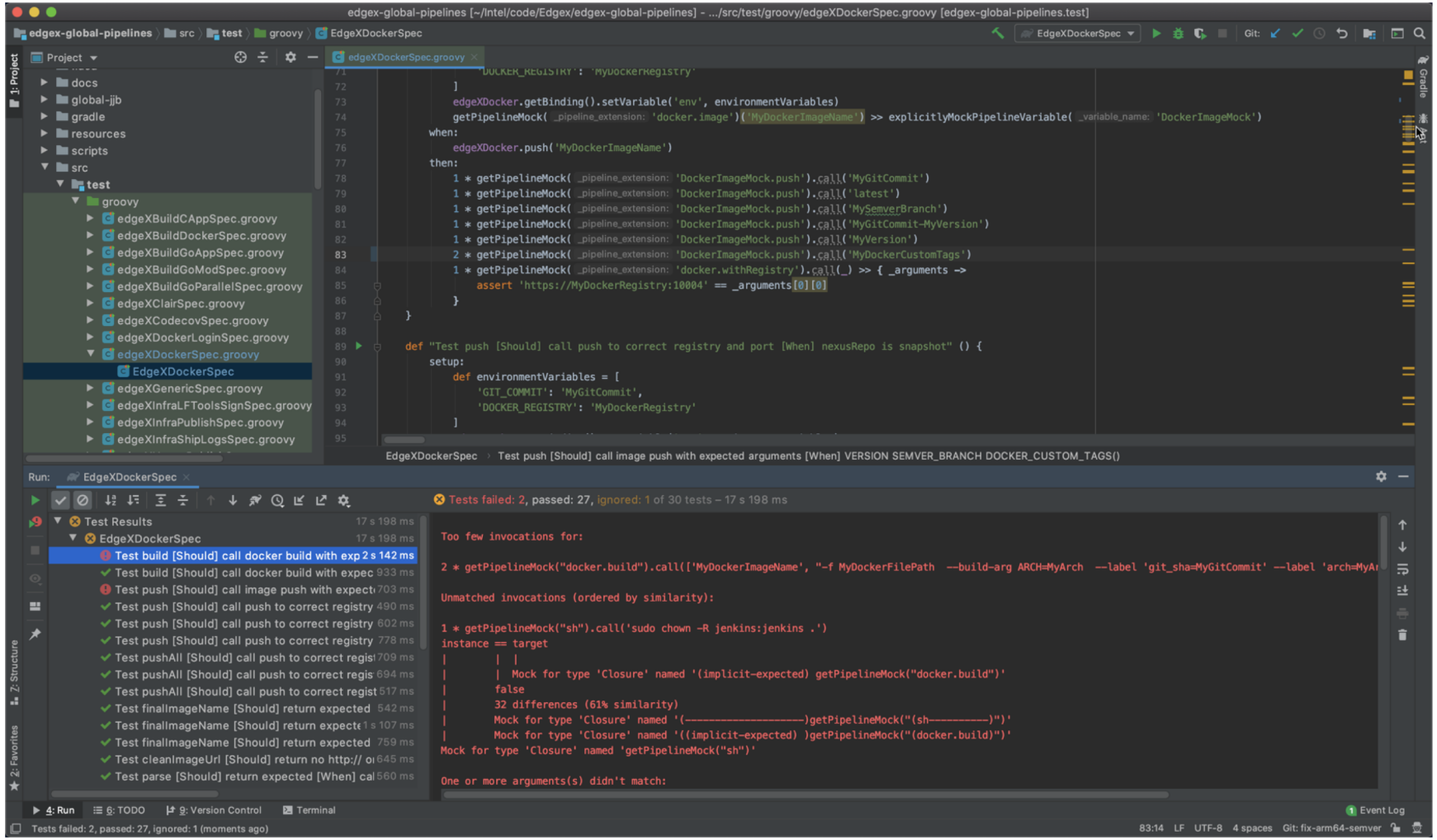
© 2020 All Rights Reserved.

IntelliJ IDEA Integration



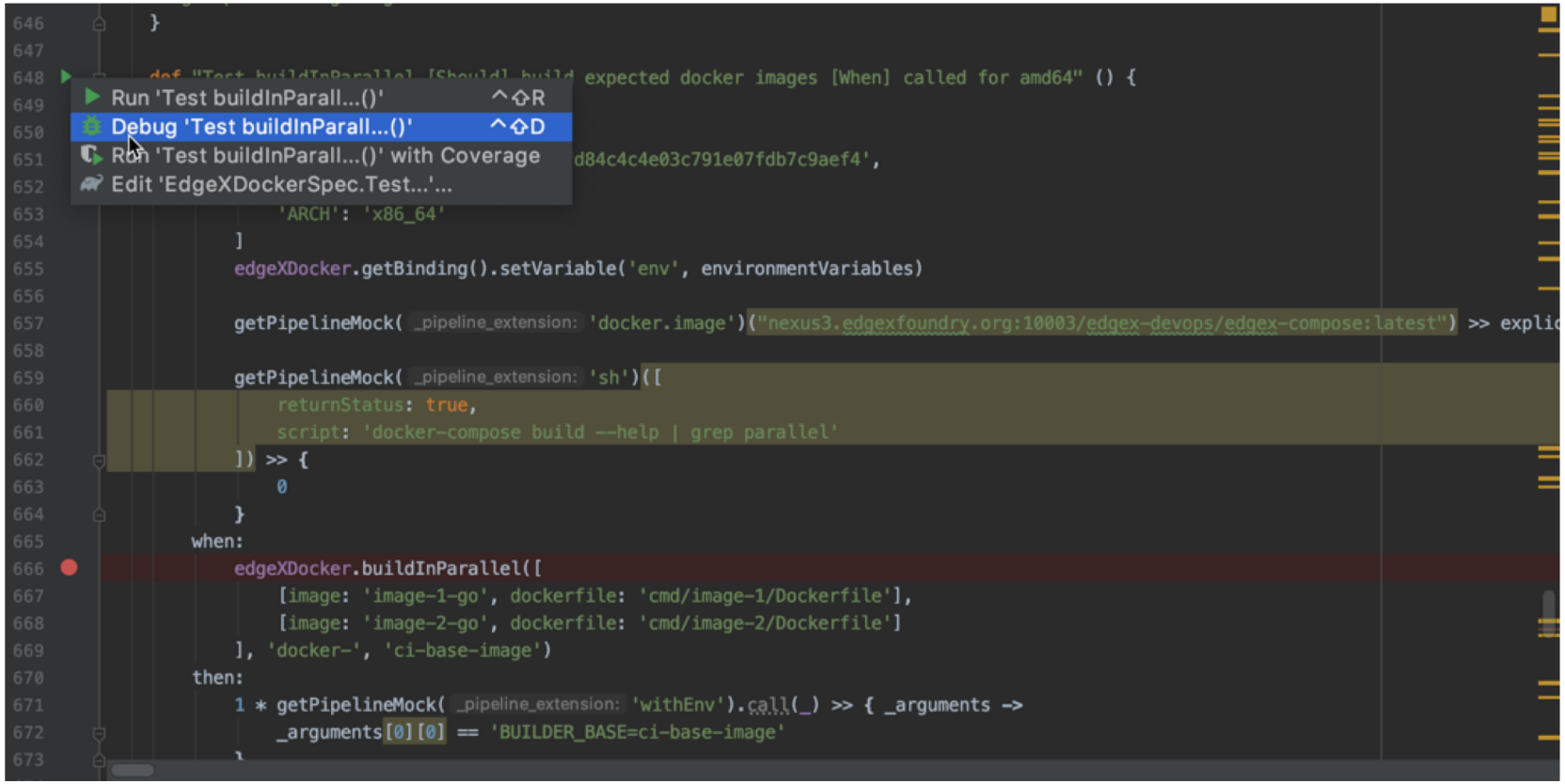
© 2020 All Rights Reserved.

IntelliJ IDEA Integration



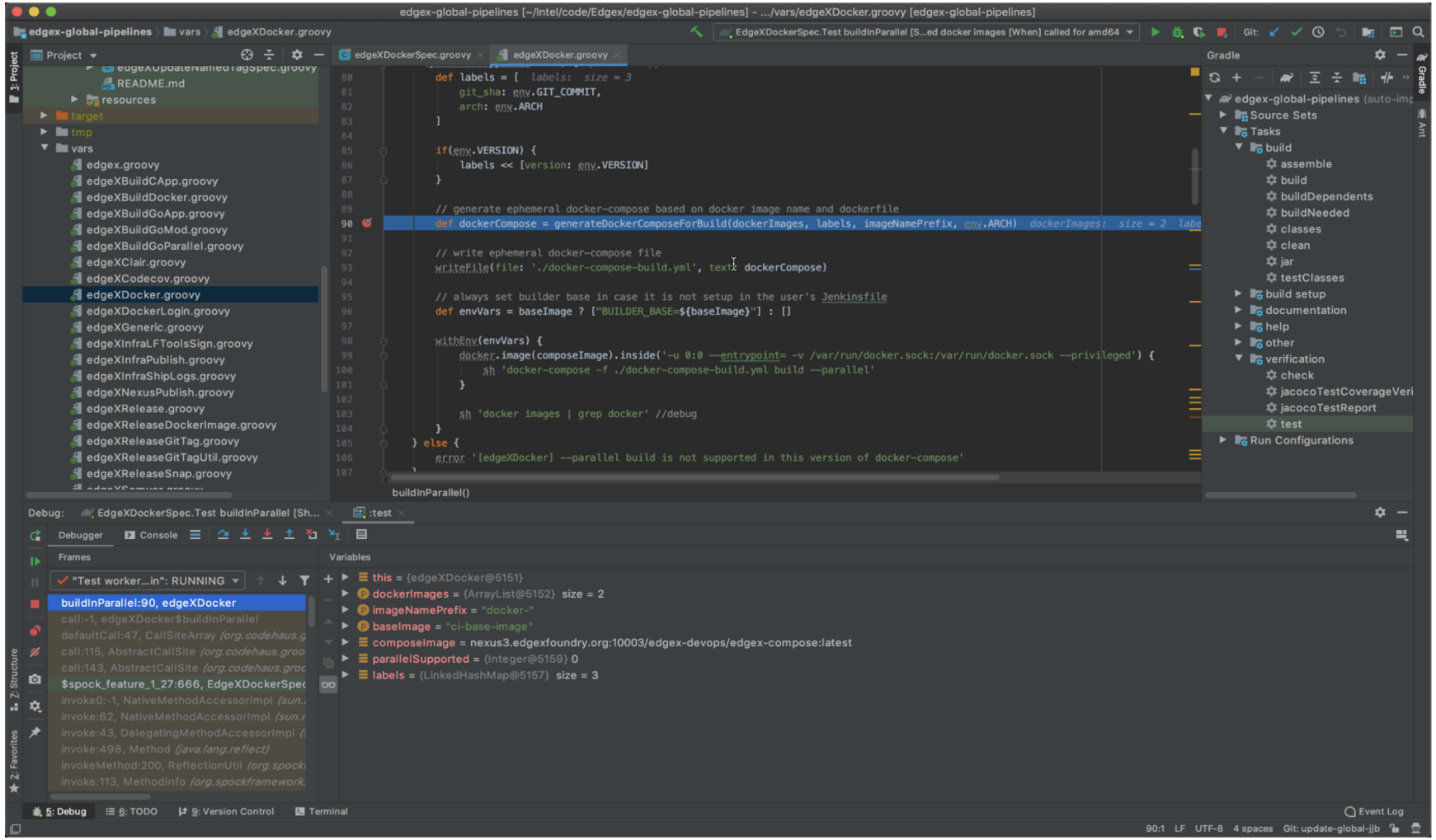
© 2020 All Rights Reserved.

IntelliJ IDEA Integration



© 2020 All Rights Reserved.

IntelliJ IDEA Integration



© 2020 All Rights Reserved.

Conclusion

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

© 2020 All Rights Reserved.