Automating your CI/CD Stack   
with Java and Groovy:

Solution Guide

Jeanne Boyarsky & Scott Selikoff

Tuesday Oct 23, 2018

Oracle Code One – HOL4957

This document is a list of answers to the things the lab asks you to figure out on your own.

Each answer is on a separate page so you don’t see them by accident.

### 2.3 Writing your own script in Jenkins

**Task:**

Fill in the blank to print out all the strings that end with “day”.

def list = ['monday', 'wednesday', 'chocolate', 'friday']

println \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Solution:**

#### Groovy way:

def list = ['monday', 'wednesday', 'chocolate', 'friday']

println list.grep { it.endsWith 'day' }

#### Java developers might be more comfortable with this:

def list = ['monday', 'wednesday', 'chocolate', 'friday']

println list.grep { d -> d.endsWith('day') }

### 3.2 Looking in the Nexus task log on the command line

**Task:**

Now find the main nexus.log

**Solution:**

cd /nexus-data/log

tail –f nexus.log

### 3.2 Looking in the Nexus task log on the command line

**Task:**

In your browser, re-run your Groovy task. How many lines do you see added to the tail output? It’s not 8 lines. Can you figure out the difference between the nexus log and task log?

**Solution:**

6 lines

|  |  |
| --- | --- |
| **Nexus Log** | **Task Log** |
| Task name start | Task information |
| Task state change to start | Task id |
| Location of log file | Task type |
| 2 lines output | Task name |
| Task state change to waiting | Task start |
|  | 2 lines output |
|  | Task complete |

### 3.3 Writing your own script in Nexus

**Task:**

Now try to write your own Groovy script that prints out a countdown.

**Solution:**

(10..1).each{ log.info '--> ' + it }

log.info 'Blast off!'

### 3.4 log vs println

**Task:**

Check the output in both the browser and command line task log. Where do you see it?

**Solution:**

println only goes to the task log. This output does not appear in the nexus log in the browser.

### 4.2 Creating a user

**Task:**

Add four more lines of code so Owen, Sam, Sophia and Daisy get accounts.

**Solution:**

import jenkins.model.Jenkins

def instance = Jenkins.getInstance()

def realm = Jenkins.getInstance().securityRealm

realm.createAccount('olivia', 'olivia')

realm.createAccount('owen', 'owen')

realm.createAccount('sam', 'sam')

realm.createAccount('sophia', 'sophia')

realm.createAccount('daisy', 'daisy')

instance.save()

### 4.3 Setting up the authorization strategy

**Task:**

Update the groovy code to set up the sea lions.

**Solution:**

import hudson.security.\*

import jenkins.model.\*

import com.michelin.cio.hudson.plugins.rolestrategy.\*

def instance = Jenkins.getInstance()

def auth = new RoleBasedAuthorizationStrategy()

// enable all permissions for administrators

def allPermissionIds = Permission.all

.grep { it.enabled }

.collect { it.id }

.join(',')

auth.doAddRole('globalRoles', 'admin', allPermissionIds, 'true', '.\*')

auth.doAssignRole('globalRoles', 'admin', 'admin')

// all authenticated users can login

def readOnly = 'hudson.model.Hudson.Read'

auth.doAddRole('globalRoles', 'authenticated', readOnly, 'true', '.\*')

// create roles for Ospreys

def jobReadOnlyPermissions = 'hudson.model.Item.Read,hudson.model.Item.Discover,hudson.model.Item.Workspace'

auth.doAddRole('projectRoles', 'authenticated', jobReadOnlyPermissions, 'true', 'osprey.\*')

def allJobAndRunPermissions = Permission.all

.grep { it.enabled }

.grep { it.id.startsWith('hudson.model.Item.') || it.id.startsWith('hudson.model.Run.') }

.collect { it.id }

.join(',')

auth.doAddRole('globalRoles', 'osprey-team', readOnly, 'true', '.\*')

auth.doAddRole('projectRoles', 'osprey-team', allJobAndRunPermissions, 'true', 'osprey.\*')

auth.doAssignRole('globalRoles', 'osprey-team', 'olivia')

auth.doAssignRole('globalRoles', 'osprey-team', 'owen')

auth.doAssignRole('projectRoles', 'osprey-team', 'olivia')

auth.doAssignRole('projectRoles', 'osprey-team', 'owen')

auth.doAssignRole('projectRoles', 'authenticated', 'olivia')

auth.doAssignRole('projectRoles', 'authenticated', 'owen')

**auth.doAddRole('globalRoles', 'sea-lion-team', readOnly, 'true', '.\*')**

**auth.doAddRole('projectRoles', 'sea-lion-team', allJobAndRunPermissions, 'true', 'sea-lion.\*')**

**auth.doAssignRole('globalRoles', 'sea-lion-team', 'sophia')**

**auth.doAssignRole('globalRoles', 'sea-lion-team', 'sam')**

**auth.doAssignRole('projectRoles', 'sea-lion-team', 'sophia')**

**auth.doAssignRole('projectRoles', 'sea-lion-team', 'sam')**

**auth.doAssignRole('projectRoles', 'authenticated', 'sophia')**

**auth.doAssignRole('projectRoles', 'authenticated', 'sam')**

instance.setAuthorizationStrategy(auth)

instance.save()

### 5.1 Learn what objects are available

**Task:**

Write your own Groovy task to output the classes in use here

**Solution:**

log.info "Core: ${core.class}"

log.info "Repository: ${repository.class}"

log.info "Blob Store: ${blobStore.class}"

log.info "Security: ${security.class}"

log.info "Container: ${container.class}"

Core: class org.sonatype.nexus.internal.provisioning.CoreApiImpl

Repository: class org.sonatype.nexus.script.plugin.internal.provisioning.RepositoryApiImpl

Blob Store: class org.sonatype.nexus.internal.provisioning.BlobStoreApiImpl

Security: class org.sonatype.nexus.security.internal.SecurityApiImpl

Container: class

org.sonatype.nexus.internal.app.GlobalComponentLookupHelperImpl

### 5.3 Setup Nexus Access

**Task:**

Try writing the Groovy code to set up the Sea Lion’s access. Remember that they get access to the Osprey code in addition to their own

**Solution:**

def seaLionSelectorConfig = new SelectorConfiguration(

name: 'sea-lion-selector',

type: 'csel',

description: 'sea-lion packages',

attributes: ['expression': 'format == "maven2" and coordinate.groupId =^ "net.selikoff.oraclecodeone.groovy.sea-lion"']

)

if (selectorManager.browse().find { it -> it.name == seaLionSelectorConfig.name } == null) {

selectorManager.create(seaLionSelectorConfig)

}

def seaLionReleaseProperties = ['contentSelector' : seaLionSelectorConfig.name,

'repository' : 'custom-releases', 'actions' : 'browse,read,edit']

def seaLionReleasePrivilege = new org.sonatype.nexus.security.privilege.Privilege(

id: "sea-lion-release-priv",

version: '',

name: "sea-lion-release-priv",

description: "Content Selector Release privilege",

type: "repository-content-selector",

properties: seaLionReleaseProperties

)

def seaLionSnapshotProperties = ['contentSelector' : 'sea-lion-selector',

'repository' : 'custom-snapshots', 'actions' : 'browse,read,edit']

def seaLionSnapshotPrivilege = new org.sonatype.nexus.security.privilege.Privilege(

id: "sea-lion-snapshot-priv",

version: '',

name: "sea-lion-snapshot-priv",

description: "Content Selector Snapshot privilege",

type: "repository-content-selector",

properties: seaLionSnapshotProperties

)

addOrReplacePrivilege(seaLionReleasePrivilege)

addOrReplacePrivilege(seaLionSnapshotPrivilege)

def seaLionRole = new org.sonatype.nexus.security.role.Role(

roleId: "sea-lion-role",

source: "Nexus",

name: "sea-lion-role",

description: "Sea Lion Role",

readOnly: false,

privileges: [ seaLionSnapshotPrivilege.id, seaLionReleasePrivilege.id, ospreySnapshotPrivilege.id, ospreyReleasePrivilege.id ],

roles: []

)

addOrReplaceRole(seaLionRole)

if (! securitySystem.searchUsers(new UserSearchCriteria('sam')))

security.addUser("sam", "Sam", "S", "sam@none.com", true, "sam", [ 'sea-lion-role', 'limited-anon' ])

if (! securitySystem.searchUsers(new UserSearchCriteria('sophia')))

security.addUser("sophia", "Sophia", "S", "sophia@none.com", true, "sophia", [ 'sea-lion-role', 'limited-anon' ])

### 6.6 Editing a pipeline job

**Task:**

The Sea Lions have decided they want to deploy to Nexus on every other build. For even numbered builds they want to just print a message to the console that they are skipping deploy. For odd numbered builds, they want the pipeline to behave normally.

**Solution:**

stage ('Publish') {

def buildNum = env['BUILD\_NUMBER'] as Integer

if (buildNum % 2 == 0) {

println 'skip publish'

} else {

withCredentials…

}

}

### 9 Extra credit

**Task:**

Jenkins validation logic

**Solution:**

def validateProject(job, expectedProject) {

def jobXml = job.configFile.asString()

// get the code inside the tags (?s) matches line breaks

def groovyCode = jobXml

.replaceFirst('(?s)^.\*<script>', '')

.replaceFirst('(?s)</script>.\*$', '')

// remove expected gradle config

def codeWithoutGoodBuildGradleCalls = groovyCode.replaceAll("$expectedProject/build.gradle", '');

// fail if any others

if (codeWithoutGoodBuildGradleCalls.contains('build.gradle')) {

throw new IllegalStateException("${job.name} calls a build.gradle file for a different project.")

}

}

jenkins.model.Jenkins.instance.getAllItems()

.grep { it.name.contains 'osprey' }

.forEach { validateProject(it, 'osprey-project') }

jenkins.model.Jenkins.instance.getAllItems()

.grep { it.name.contains 'sea-lion' }

.forEach { validateProject(it, 'sea-lion-project') }