OSM Hackfest Session 11
Robot Framework

Javier García (Tech Mahindra)
Intro to OSM CI/CD
Intro to OSM CI/CD

- OSM code is built and released using CI/CD pipeline
- Everything is automated
  - triggered by commits and scheduled jobs
- Requires validation from MDL leaders and provides feedback to developers
- Uses the following tools:
  - Jenkins, Docker, FOSSology, JFrog Artifactory, Robot Framework and custom scripts
  - ETSI HIVE
CI/CD Pipeline

- Trigger per-module pipelines
- Stage 1
- Scan
- U Test
- Build
- Archive
- Stage 2
- Scan
- U Test
- Build
- Archive
- Artifact Storage
- Stage 3
- System Install
- Stage 4
- NFVI & VIM
- System Tests
- Smoke Tests
- Artifactory

© ETSI 2019
CI/CD Stage 1: Launch

- Triggered by Gerrit upon code commit (or patch set submit)
- Launch multi-branch pipelines
- Calls Stage 2 to build required modules
CI/CD Stage 2: Per-module pipeline

- Triggered by Stage 1, on specific modules
- Operates within a Docker container
- Per-module callbacks:
  - License Scan (FOSSology)
  - Unit Test
  - Package build
  - Artifact creation & storage (JFrog Artifactory)
CI/CD Stage 3: System Integration

• System Install from binaries
  • From Stage 2 & Artifact Storage

• Smoke Tests
  • Based on Robot Framework and pytest
  • Test OSM functionality
    • API checks, VNFD, NSD upload…
    • Leveraging OSM Client library
  • Basic testing can be extended with VIM-EMU
CI/CD Stage 4: System Testing

- Automated system and interop testing
- Multiple NFVI, VIM and Controllers
- As Stage 4 of the pipeline standalone job
- Uses HIVE: Hub for Interoperability and Validation at ETSI
Robot Framework

OSM E2E Testing
What we will be learning

• What is Robot Framework?
• Robot setup
• Execute OSM DevOps test suite
• Learn Robot test scripts basics
• Create your own OSM tests
  • Simple CLI test
  • Simple GUI test
What is Robot Framework?

• Generic open source test automation framework.
• Suitable for both end-to-end acceptance testing and acceptance-test-driven development (ATDD).
• The test syntax follows a tabular style and plain text format which makes writing test cases more user-friendly and easy to read.
Why Robot Framework?

• **Keyword driven**, tabular and easy to understand syntax for test case development

• Allows creation of **reusable keywords**

• Allows creation of **custom keywords**

• Platform and application **independent**

• Support for **standard and external libraries** for test automation

• **Tagging** to categorize and select test cases to be executed

• **Easy-to-read reports and logs** in HTML format
Robot Setup

- Check osmclient and OSM IM packages are installed
  sudo apt-get update
  sudo apt-get -y install python-osmclient python-osm-im

- Install Robot and its dependencies
  pip install haikunator requests robotframework robotframework-requests robotframework-seleniumlibrary
  pip list #check installed packages
  robot --version #verify installation

- Install and configure lighttpd to visualize HTML reports
  sudo apt-get install lighttpd
  sudo vi /etc/lighttpd/lighttpd.conf
  - server.document-root <path to Robot framework report output folder- mkdir in /tmp>
  - server.port <change to available port-8088> (Make sure security groups are defined for lighttpd port)
  systemctl restart lighttpd.service
Execute OSM DevOps test suite

• Clone OSM devops repo
  git clone "https://osm.etsi.org/gerrit/osm/devops"

• Test suites are under “robot-systest/testsuite”

• Make sure OSM env variables are available (OSM_HOSTNAME)

• Execute basic VIM test suite
  robot -d robot-systest/reports -i vim
  robot-systest/testsuitetestsuite/cli/TS02__Test_Vim.robot

• Take a look at the available test suites for more tags (e.g. “comprehensive”)
Execute OSM DevOps test suite (II)
Test script architecture

- Different sections in data:

<table>
<thead>
<tr>
<th>Section</th>
<th>Used For</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings</td>
<td>1) Importing test libraries, resource files and variable files.</td>
</tr>
<tr>
<td></td>
<td>2) Defining metadata for test suites and test cases.</td>
</tr>
<tr>
<td>Variables</td>
<td>Defining variables that can be used elsewhere in the test data.</td>
</tr>
<tr>
<td>Test Cases</td>
<td>Creating test cases from available keywords.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Creating user keywords from existing lower-level keywords</td>
</tr>
</tbody>
</table>
Test script architecture: Settings

*** Settings ***
1) Importing test libraries, resource files and variable files.
2) Defining metadata for test suites and test cases.

   Documentation   Example using the space separated plain text format.
   Library         OperatingSystem

Import libraries
Library libraryName arg1 arg2...

Import External Keyword resources
Resource ..../keywords/myKWords.robot

Setup and Teardown
Suite Setup     My Suite Setup Keyword
Suite Teardown  My Suite Setup Keyword
Test Setup      My Test Setup Keyword
Test Teardown   My Test Setup Keyword variables.

Tags
Force Tags    TAG1    TAG2
Default tags  TAG
Test script architecture: Variables

*** Variables ***

Define variables at a "tests suite scope". Variables declared here are accessible from every test cases, keywords or settings.

Creating scalar variables
*** Variables ***

${variable_name} value

Creating list variables
*** Variables ***

@{list_name} first second third
@{list_name_2} @{NAMES} dummytext

Accessing list items

${list_name}[index]

Creating dictionary variables
*** Variables ***

&{dictionary_name} first=1 second=${variable_name} ${3}=third
&{EVEN MORE} &{MANY} first=override empty=

Accessing dictionary items

${dictionary_name}[key]
Test script architecture: Test Cases

*** Test Cases ***
List of test cases with each test steps inside. Settings of a test cases are:

[Documentation] Used for specifying the test documentation
[Tags] Used tagging test cases
[Setup], [Teardown] Specify test setup (executed before the test) and teardown (executed after the test, even if test failed)
[Template] Specify the template keyword to use for each step
[Timeout] Set the test case execution timeout (Test fails if timeout is reached)

*** Test Cases ***
My Test
[Documentation] Example test
Log ${MESSAGE}
My Keyword /tmp

Another Test
Should Be Equal ${MESSAGE} Hello, world!
Test script architecture: Keywords

*** Keywords ***
Contains keywords commons to your test suite. Keywords declared here can be used anywhere in the suite, even in setup and teardown calls. Keywords settings are:

[Documentation] Used for specifying the keyword documentation
[Arguments] Specify the keyword arguments
[Return] Specify the keyword return value
[Timeout] Set the keyword execution timeout (Test fails if timeout is reached)

*** Keywords ***
My Keyword
  [Arguments]  ${path}
    Directory Should Exist  ${path}
Create your own OSM test case

• Download help libraries with keywords for OSM
  
  wget

• Unzip
  
  unzip robot_script.zip

• Create a file for a simple VIM test using OSM CLI
  
  vi hackfest_test.robot
Test Script: VIM Test using OSM CLI

*** Settings ***
Documentation  Test Suite to create and delete vim account
Library        Collections
Library        OperatingSystem
Resource       osm_cli_lib.robot

*** Variables ***
${success_return_code}  0
${vim_name}  "helloworld-os"
${vim_user}  "roboittest"
${vim_password}  "dummy"
${auth_url}  "https://127.0.0.1/"
${vim_type}  "openstack"
${description}  "a test vim"
${tenant}  "roboittest2"

...
Test Script: VIM Test using OSM CLI (II)

*** Test Cases ***
Create Vim Account
  [Tags] vim
  Create Vim Cli ${vim_name} ${vim_type} ${auth_url} ${vim_user} ${vim_password} ${tenant} ${description}

List Vim Accounts
  [Tags] vim
  Get Vim List Cli

Delete Vim Account
  [Tags] vim
  Delete Vim Cli ${name}
Test Script: VIM Test using OSM CLI (III)

- Execute the test case

  `robot -d devops/robot-systest/reports -i vim hackfest_test.robot`

```plaintext
Create Vim Account

List Vim Accounts

List Vim Accounts

Delete Vim Account

Vim Setup Test :: Test Suite to create and delete vim account

3 critical tests, 3 passed, 0 failed
3 tests total, 3 passed, 0 failed

Output: /home/ubuntu/robot-session/devops/robot-systest/reports/output.xml
Log: /home/ubuntu/robot-session/devops/robot-systest/reports/log.html
```
Check test results in browser

• For general report
  • http://<web_server_ip>:<port>/report.html

• For a more detailed description of the steps
  • http://<web_server_ip>:<port>/log.html
Requisites for GUI testing

- A browser (e.g. Chrome) and its Selenium driver (link)

```
sudo apt-get update
sicudo apt-get install -y unzip xvfb libxi6 libgconf-2-4 curl
sicudo apt-get install default-jdk
curl -sS -o - https://dl-ssl.google.com/linux/linux_signing_key.pub | sudo apt-key add -
sicudo apt-get -y update
sicudo apt-get -y install google-chrome-stable

#Download and install Selenium Chrome driver
wget https://chromedriver.storage.googleapis.com/2.41/chromedriver_linux64.zip
unzip chromedriver_linux64.zip

sicudo mv chromedriver /usr/bin/chromedriver
sicudo chown root:root /usr/bin/chromedriver
sicudo chmod +x /usr/bin/chromedriver
```
Test Script: VIM Test using OSM GUI

*** Settings ***
Documentation Test Suite to create and delete vim account
Library Collections
Library OperatingSystem
Library SeleniumLibrary
Resource osm_cli_lib.robot
Resource osm_gui_lib.robot

*** Variables ***
[...]
${vim_type} "openstack"
${description} "a test vim"
${tenant} "robottest2"
${osm_host} 172.21.248.5
${osm_user} admin
${osm_password} admin
Test Script: VIM Test using OSM GUI (II)

*** Test Cases ***
[...]

Valid Login
[Tags] gui_login_test
[Setup] Set Server URL ${osm_host}
Open Browser To Login Page
Enter Credentials ${osm_user} ${osm_password}
Submit Credentials
Home Page Should Be Open
[Teardown] Close Browser
• Execute the test case

```bash
robot -d devops/robot-systest/reports -i gui_login_test hackfest_test.robot
```
Useful information
References

Develops/User guide:


https://github.com/robotframework/QuickStartGuide/blob/master/QuickStart.rst


https://bulkan.github.io/robotframework-requests/

Editor:

https://pypi.org › project › robotframework-ride
https://macromates.com/
https://github.com/nokia/RED
http://www.jetbrains.com/pycharm/
Join community #Devops


<table>
<thead>
<tr>
<th></th>
<th>Feature</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>034</td>
<td>Disable port security at network level</td>
<td>PASSED</td>
</tr>
<tr>
<td>37</td>
<td>035</td>
<td>Feature 1396 - Eclipse tftp6</td>
<td>PASSED</td>
</tr>
<tr>
<td>38</td>
<td>038</td>
<td>Feature 1417 - Support of ROUs</td>
<td>Not applicable</td>
</tr>
<tr>
<td>39</td>
<td>037</td>
<td>Feature 1420 - VNF SVV upgrade (Adam)</td>
<td>PASSED</td>
</tr>
<tr>
<td>40</td>
<td>038</td>
<td>Feature 638 - Service chaining</td>
<td>PASSED</td>
</tr>
<tr>
<td>41</td>
<td>039</td>
<td>Feature 1413 - OSM platform resiliency to single component failure</td>
<td>FAILED</td>
</tr>
<tr>
<td>42</td>
<td>040</td>
<td>Feature 1412 - OSM platform recovery after major failure</td>
<td>PASSED</td>
</tr>
<tr>
<td>43</td>
<td>041</td>
<td>Feature 5550 - Allow to specify management P addresses as parameters at instantiation time</td>
<td>PASSED</td>
</tr>
<tr>
<td>44</td>
<td>042</td>
<td>Feature 5945 - Enable dynamic connectivity setup in multi-data Network Services (only CRUD over WIM)</td>
<td>PASSED</td>
</tr>
<tr>
<td>45</td>
<td>043</td>
<td>Feature - Control of LCM operations over a NS instance</td>
<td>PASSED</td>
</tr>
</tbody>
</table>

#devops on slack
#bi-weekly meeting

OSM TECH – DevOps bi-weekly calls

Wednesdays @ 16:00 CEST

To join: https://www.gotomeet.me/OSMTECH

Access Code: 119-703-237
Q & A

Thank You