Re-using OPNFV framework tests for LFN projects

Eric Debeau, Cédric Ollivier, Morgan Richomme
Orange

March, 26 2018
Agenda

• OPNFV testing tools

• Xtesting project

• Experience with ONAP in Orange OpenLab
Testing in OPNFV DNA

- OPNFV is an integration project
- Testing is key to verify the global solution
OPNFV testing tools

Functest

• A framework
  - handle all interactions with OPNFV CI/CD components (entry points, results publication, status codes, etc.)
  - ease the development of third-party test cases by offering multiple drivers: Python, Bash, unittest, robot framework and VNF.

• Test cases mainly integrating upstream components:
  - OpenStack Rally, Tempest
  - OpenDayLight Neutron Suite
  - OPNFV snaps
Functest evolution

- Functest has to verify Kubernetes deployment but its original framework is linked to OpenStack (e.g. credentials sourcing, rally verifiers, etc.)

- Hosting both OpenStack and Kubernetes in the same Python package would increase dependencies and complicate container slicing

Why not refactoring the first Functest Framework?
To facilitate the tester life

- Functest python and containers framework could be very useful out of OPNFV (ease developing test cases, manage requirements and offer lightweight Docker images)

- A new Functest design could simplify test integration in a complete [OPNFV-based CI/CD toolchain](#) (e.g. Testing Containers, Test API and dashboard)

Let the developer only work on the test suites without diving into CI/CD integration
Xtesting framework

• Functest framework were moved to a new xtesting repository (functest only hosts OpenStack test cases)

• It has been updated and improved to follow all Xtesting technical guidelines:
  - unlink to OpenStack and OPNFV
  - support both Python2 and Python3 (required by Functional Gating)
  - be fully covered by unit tests and well rated by Pylint (10/10)
Xtesting deliverables

• Xtesting is released as a Python package and then is unlinked to OPNFV Milestones (Functest Python package now depends on it)

• opnfv/xtesting is proposed to build third-parties containers (both amd64 and arm64 architectures).

• The API documentation is automatically built online
## Functest & Xtesting in Orange ONAP OpenLab

- Verify the infrastructure (deployed by OPNFV XCI) via Functest

<table>
<thead>
<tr>
<th>TEST CASE</th>
<th>PROJECT</th>
<th>TIER</th>
<th>DURATION</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection_check</td>
<td>functest</td>
<td>healthcheck</td>
<td>00:07</td>
<td>PASS</td>
</tr>
<tr>
<td>api_check</td>
<td>functest</td>
<td>healthcheck</td>
<td>07:46</td>
<td>PASS</td>
</tr>
<tr>
<td>snaps_health_check</td>
<td>functest</td>
<td>healthcheck</td>
<td>00:36</td>
<td>PASS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST CASE</th>
<th>PROJECT</th>
<th>TIER</th>
<th>DURATION</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vping_ssh</td>
<td>functest</td>
<td>smoke</td>
<td>00:57</td>
<td>PASS</td>
</tr>
<tr>
<td>vping_userdata</td>
<td>functest</td>
<td>smoke</td>
<td>00:33</td>
<td>PASS</td>
</tr>
<tr>
<td>tempest_smoke_serial</td>
<td>functest</td>
<td>smoke</td>
<td>13:22</td>
<td>PASS</td>
</tr>
<tr>
<td>rally_sanity</td>
<td>functest</td>
<td>smoke</td>
<td>24:07</td>
<td>PASS</td>
</tr>
<tr>
<td>refstack_defcore</td>
<td>functest</td>
<td>smoke</td>
<td>05:21</td>
<td>PASS</td>
</tr>
<tr>
<td>patrole</td>
<td>functest</td>
<td>smoke</td>
<td>04:29</td>
<td>PASS</td>
</tr>
<tr>
<td>snaps_smoke</td>
<td>functest</td>
<td>smoke</td>
<td>46:54</td>
<td>PASS</td>
</tr>
<tr>
<td>odl</td>
<td>functest</td>
<td>smoke</td>
<td>00:00</td>
<td>SKIP</td>
</tr>
<tr>
<td>odl_netvirt</td>
<td>functest</td>
<td>smoke</td>
<td>00:00</td>
<td>SKIP</td>
</tr>
<tr>
<td>neutron_trunk</td>
<td>functest</td>
<td>smoke</td>
<td>00:00</td>
<td>SKIP</td>
</tr>
</tbody>
</table>
### Functest & Xtesting in Orange ONAP OpenLab

- Runs some VNF test-cases

<table>
<thead>
<tr>
<th>TEST CASE</th>
<th>PROJECT</th>
<th>TIER</th>
<th>DURATION</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudify_ims</td>
<td>functest</td>
<td>vnf</td>
<td>28:15</td>
<td>PASS</td>
</tr>
<tr>
<td>vyos_vrouter</td>
<td>functest</td>
<td>vnf</td>
<td>17:59</td>
<td>PASS</td>
</tr>
<tr>
<td>juju_epc</td>
<td>functest</td>
<td>vnf</td>
<td>46:44</td>
<td>PASS</td>
</tr>
</tbody>
</table>
Re-use may be painful, so let’s check in OpenLab

- Re-use existing Robot tests in a specialized Docker container (<100 MB) instead of the classical ONAP testing virtual machine (> 1GB).
- Store test results
- Tests can be triggered from a Jenkins jobs
- Evaluate the complexity to reuse Xtesting framework for ONAP and use it in Orange OpenLab
Is it complex? 3 files to edit

Dockerfile

Dependencies

Testcases.yaml

Orange-OpenSource/xtesting-onap-robot
Dockerfile

FROM opnfv/xtesting
ARG OPENSTACK_TAG=stable/pike
ARG OPNFV_TAG=master
ARG ONAP_TAG=master

ENV PYTHONPATH $PYTHONPATH:/src/testing-utils/eteutils

COPY thirdparty-requirements.txt thirdparty-requirements.txt
RUN apk --no-cache --virtual .build-deps --update \
    python-dev build-base linux-headers libffi-dev \
    openssl-dev libjpeg-turbo-dev && \
    git clone --depth 1 https://git.onap.org/testsuite -b SONAP_TAG /var/opt/OpenECOMP_ETE && \
    git clone --depth 1 https://git.onap.org/testsuite/properties -b SONAP_TAG /share/config && \
    git clone --depth 1 https://git.onap.org/testsuite/python-testing-utils -b SONAP_TAG /src/testing-utils && \
    pip install \
    -c https://git.openstack.org/cgit/openstack/requirements/plain/upper-constraints.txt?h=OPENSTACK_TAG \ 
    -c https://git.onap.org/functest/plain/upper-constraints.txt?h=SONAP_TAG \ 
    -r thirdparty-requirements.txt \ 
    -e /src/testing-utils && \
    rm -r thirdparty-requirements.txt /src/testing-utils/.git /var/opt/OpenECOMP_ETE/.git && \
    apk del .build-deps

RUN mkdir -p /var/opt/OpenECOMP_ETE
COPY testcases.yaml /usr/lib/python2.7/site-packages/xtesting/cl/testcases.yaml

CMD ["run_tests", ",-t", ",all"]
## Dependencies

The libraries needed by ONAP tests with the reference version:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>selenium&lt;=3.0.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>requests==2.11.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>robotframework-selenium2library==1.8.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>robotframework-databaselibrary==0.8.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>robotframework-extendedselenium2library==0.9.1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>robotframework-requests==0.4.5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>robotframework-sshlibrary==2.1.2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>robotframework-sudslibrary==0.8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>robotframework-ftplibrary==1.3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>robotframework-rammbock==0.4.0.1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>deepdiff==2.5.1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>dnspython==1.15.0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>robotframework-httplibrary==0.4.2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>robotframework-archivelibrary==0.3.2</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>PyYAML==3.12</td>
<td></td>
</tr>
</tbody>
</table>
Test cases

Test case name

Test case criteria 100 = 100 % OK

Blocking: if true
=> stop all tests if test not PASS

Run: indicate the type of tests (Robot Framework, Python, Bash, VNF,..) and associated arguments

Here we use Robot Framework tags to run subset of the upstream suite
Every test is stored

- Tests may be included in CI/CD
- Automatically push the results to the Test DB through the test API
- Automatically integrated in different dashboards
Benefits for the LFN projects

• Xtesting allows a proper design inside OPNFV

• Xtesting and Functest help other LFN projects:
  - verifying the infrastructure on top of which the components are deployed
  - ease verifying the components as well in the same CI/CD toolchain

All contributions coming from LFN projects are more than welcome!
Slim Dockers to go fast

• ~1GB versus ~100 MB for the same tests
  - Xtesting is a good vehicule for tests slimification

nexus3.onap.org:10001/openecomp/testsuite

colvert22/xtesting-onap-robot

• Slim Dockers are mandatory for a real CI/CD system...especially for gating
MERCI ;-}