Re-using OPNFV framework tests for LFN projects

Eric Debeau, Cédric Ollivier, Morgan Richomme
Orange

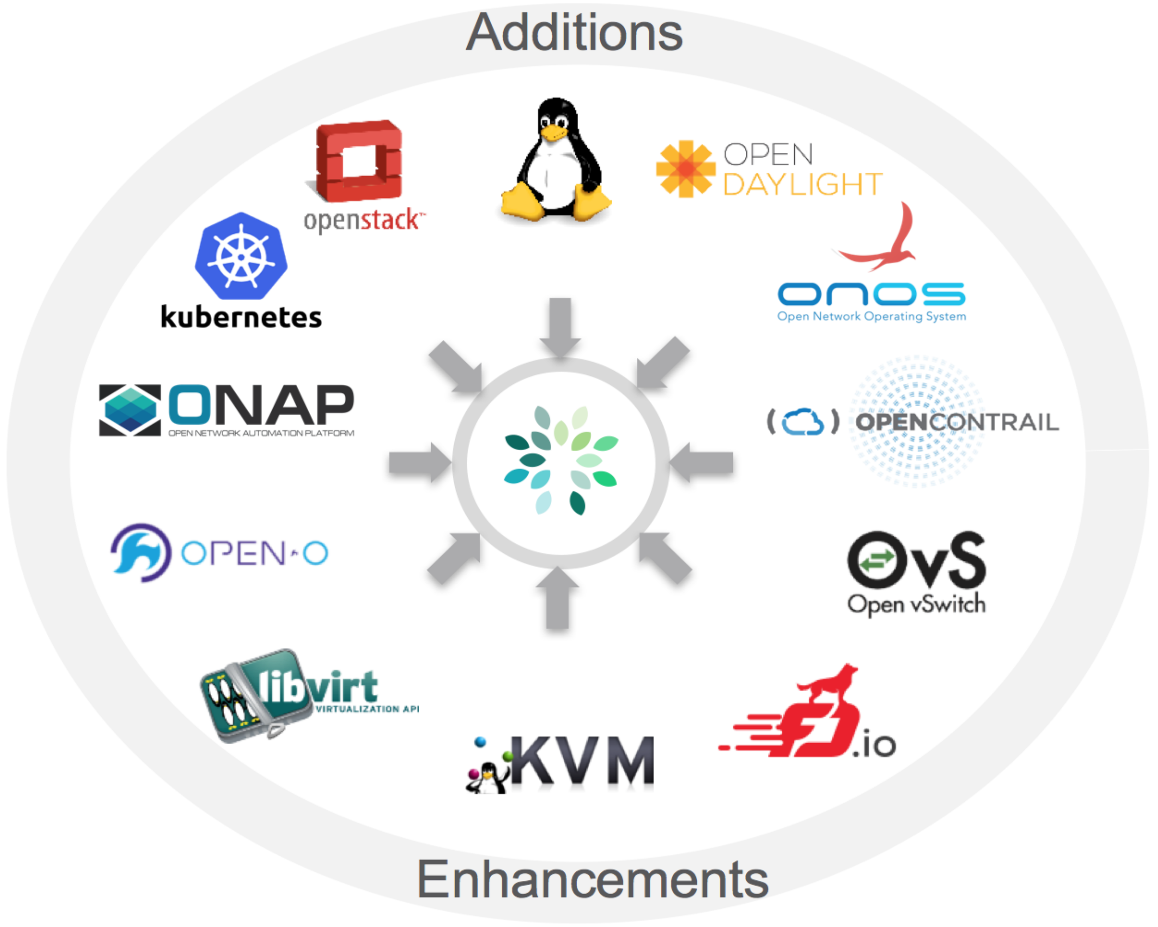
March, 26 2018

Agenda

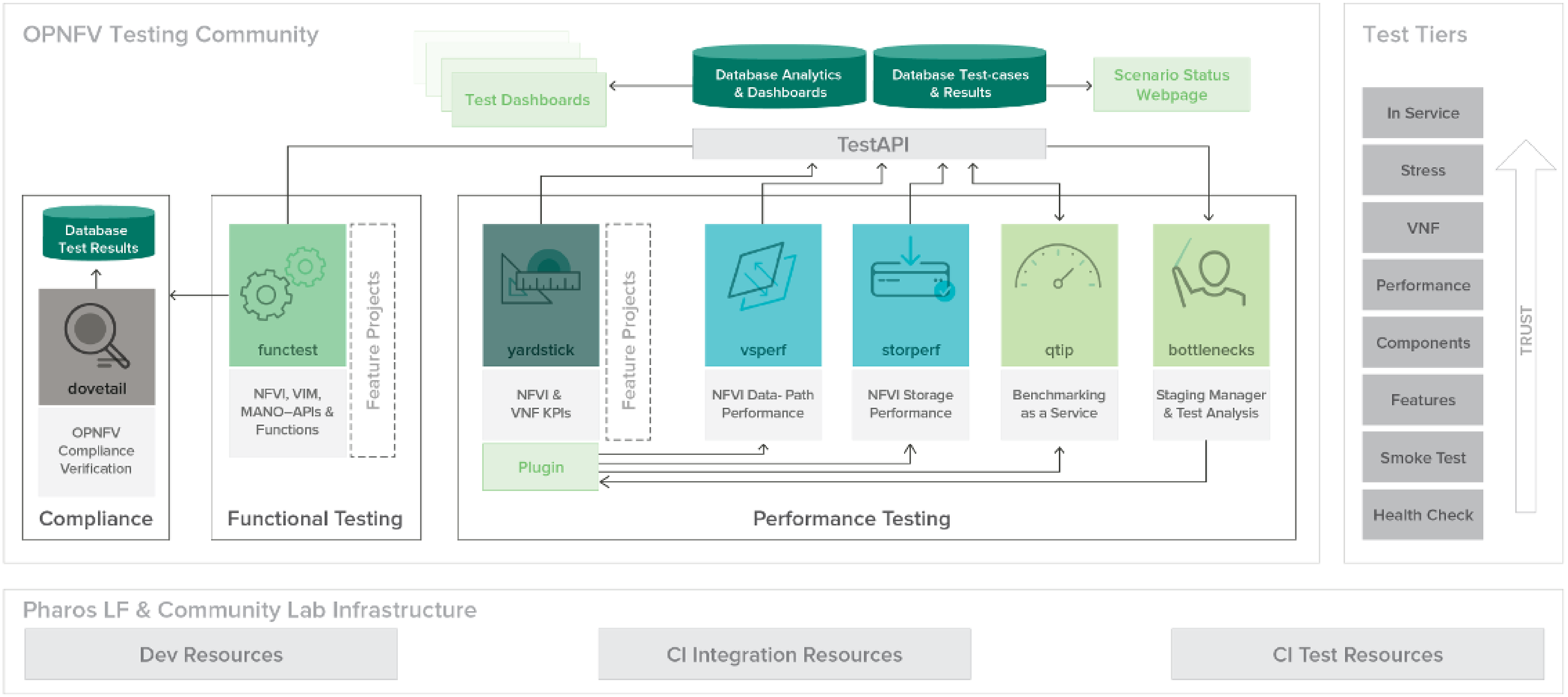
- OPNFV testing tools
- Xtesting project
- Experience with ONAP in Orange OpenLab

Testing in OPNFV DNA

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


Additions
openstack
OPEN DAYLIGHT
kubernetes
onos Open Network Operating System
ONAP OPEN NETWORK AUTOMATION PLATFORM
OPENCONTRAIL
OPEN-O
OvS Open vSwitch
libvirt VIRTUALIZATION API
KVM
.io
Enhancements

OPNFV testing tools



http://docs.opnfv.org/en/stable-euphrates/testing/ecosystem/overview.html

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

```
+----------------------------+-------------------+----------------------+-------------------+-----------------+
|         TEST CASE          |      PROJECT      |         TIER         |     DURATION      |     RESULT      |
+----------------------------+-------------------+----------------------+-------------------+-----------------+
|      connection_check      |      functest     |      healthcheck     |       00:07       |       PASS      |
|         api_check          |      functest     |      healthcheck     |       07:46       |       PASS      |
|     snaps_health_check     |      functest     |      healthcheck     |       00:36       |       PASS      |
+----------------------------+-------------------+----------------------+-------------------+-----------------+
+------------------------------+-------------------+----------------+-------------------+-----------------+
|          TEST CASE           |      PROJECT      |      TIER      |     DURATION      |     RESULT      |
+------------------------------+-------------------+----------------+-------------------+-----------------+
|          vping_ssh           |      functest     |     smoke      |       00:57       |       PASS      |
|        vping_userdata        |      functest     |     smoke      |       00:33       |       PASS      |
|     tempest_smoke_serial     |      functest     |     smoke      |       13:22       |       PASS      |
|         rally_sanity         |      functest     |     smoke      |       24:07       |       PASS      |
|       refstack_defcore       |      functest     |     smoke      |       05:21       |       PASS      |
|           patrole            |      functest     |     smoke      |       04:29       |       PASS      |
|         snaps_smoke          |      functest     |     smoke      |       46:54       |       PASS      |
|             odl              |      functest     |     smoke      |       00:00       |       SKIP      |
|         odl_netvirt          |      functest     |     smoke      |       00:00       |       SKIP      |
|        neutron_trunk         |      functest     |     smoke      |       00:00       |       SKIP      |
+------------------------------+-------------------+----------------+-------------------+-----------------+
```

Functest & Xtesting in Orange ONAP OpenLab

- Runs some VNF test-cases

```
+----------------------+-------------------+---------------+-------------------+-----------------+
|      TEST CASE       |      PROJECT      |      TIER     |      DURATION     |      RESULT     |
+----------------------+-------------------+---------------+-------------------+-----------------+
|     cloudify_ims     |     functest      |      vnf      |       28:15       |       PASS      |
|     vyos_vrouter     |     functest      |      vnf      |       17:59       |       PASS      |
|       juju_epc       |     functest      |      vnf      |       46:44       |       PASS      |
+----------------------+-------------------+---------------+-------------------+-----------------+
```

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

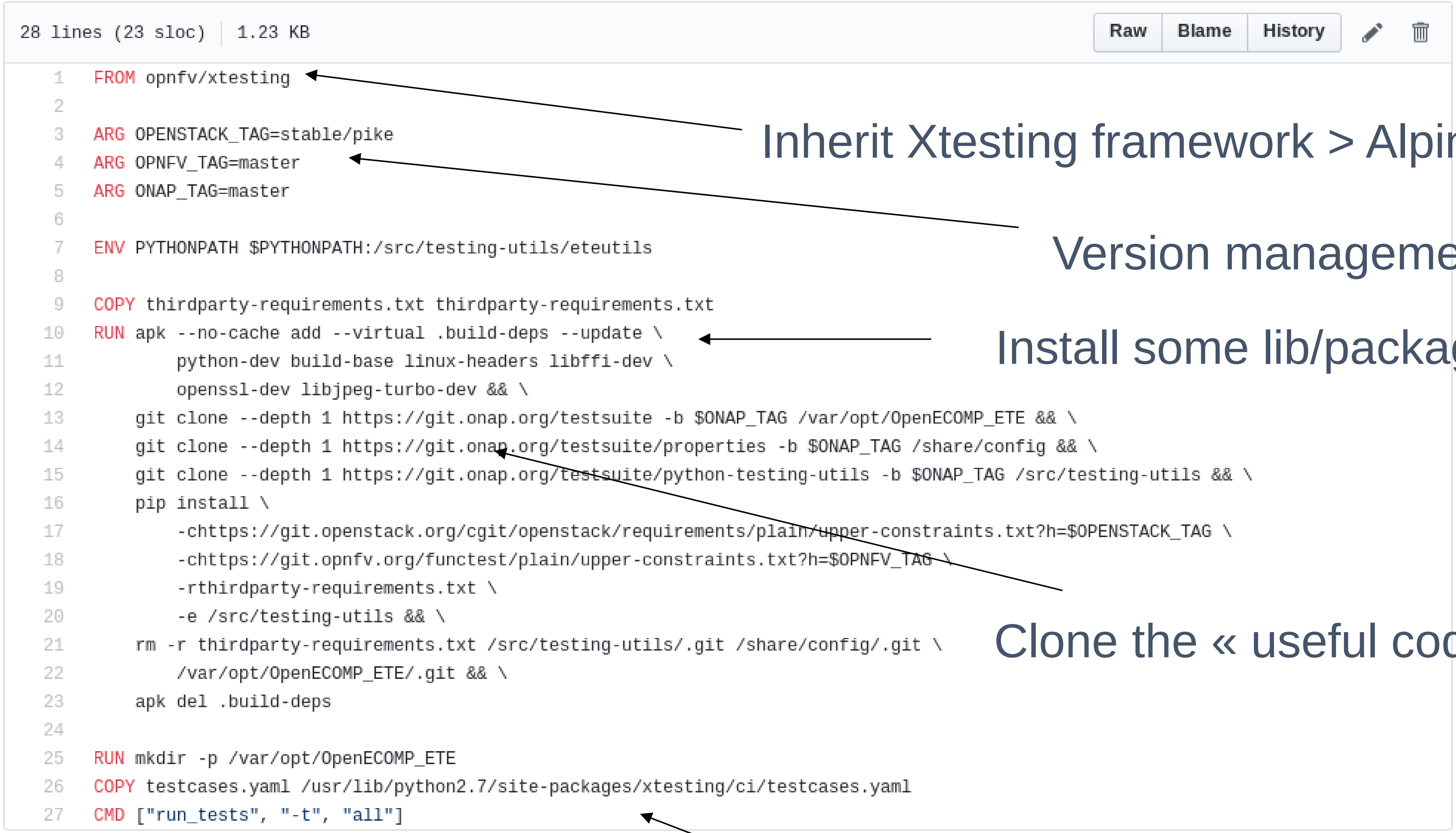
Dockerfile

Dependencies

Testcases.yaml

Orange-OpenSource/xtesting-onap-robot

Dockerfile



```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 add --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 $ONAP_TAG /var/opt/OpenECOMP_ETE && \
    git clone --depth 1 https://git.onap.org/testsuite/properties -b $ONAP_TAG /share/config && \
    git clone --depth 1 https://git.onap.org/testsuite/python-testing-utils -b $ONAP_TAG /src/testing-utils && \
    pip install \
        -chttps://git.openstack.org/cgit/openstack/requirements/plain/upper-constraints.txt?h=$OPENSTACK_TAG \
        -chttps://git.opnfv.org/functest/plain/upper-constraints.txt?h=$OPNFV_TAG \
        -rthirdparty-requirements.txt \
        -e /src/testing-utils && \
    rm -r thirdparty-requirements.txt /src/testing-utils/.git /share/config/.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/ci/testcases.yaml
CMD ["run_tests", "-t", "all"]
```

Inherit Xtesting framework > Alpines

Version management

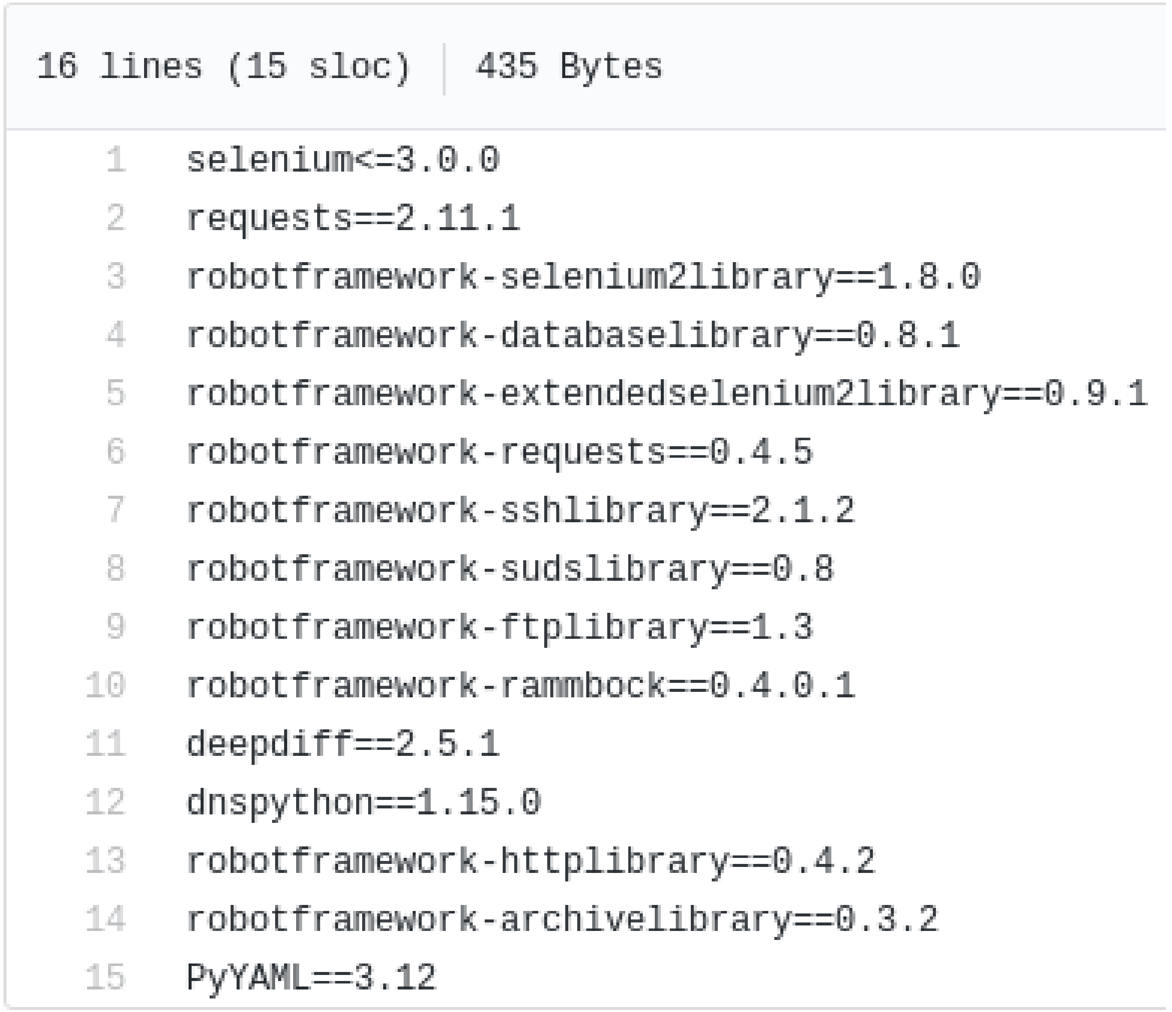
Install some lib/packages

Clone the « useful code »

Copy the test cases definition

Run the tests

Dependencies



```
16 lines (15 sloc) | 435 Bytes
selenium<=3.0.0
requests==2.11.1
robotframework-selenium2library==1.8.0
robotframework-databaselibrary==0.8.1
robotframework-extendedselenium2library==0.9.1
robotframework-requests==0.4.5
robotframework-sshlibrary==2.1.2
robotframework-sudslibrary==0.8
robotframework-ftplibrary==1.3
robotframework-rammbock==0.4.0.1
deepdiff==2.5.1
dnspython==1.15.0
robotframework-httplibrary==0.4.2
robotframework-archivelibrary==0.3.2
PyYAML==3.12
```

The libraries needed by ONAP tests with the reference version

Testcases.yaml

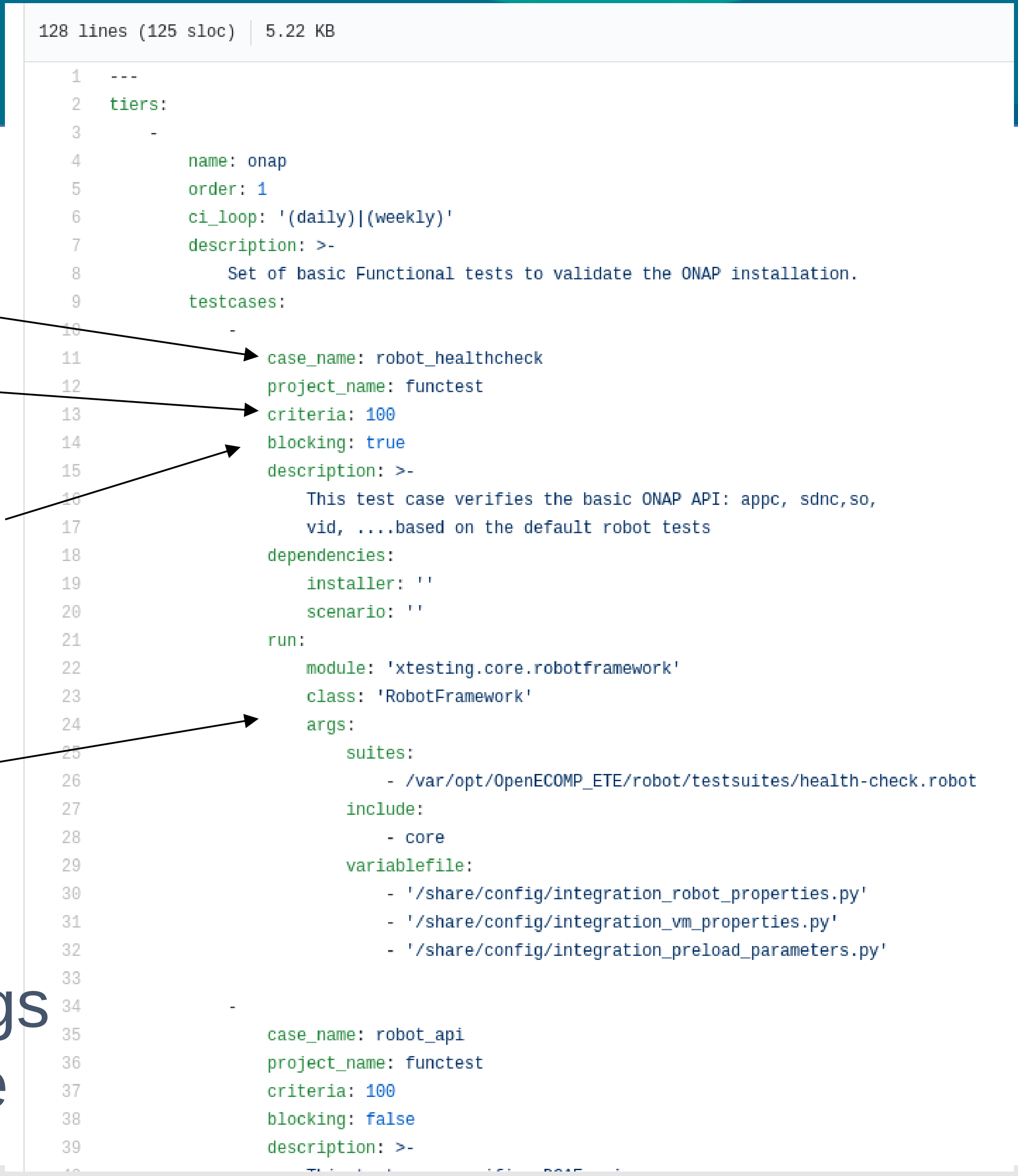
Test case name

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



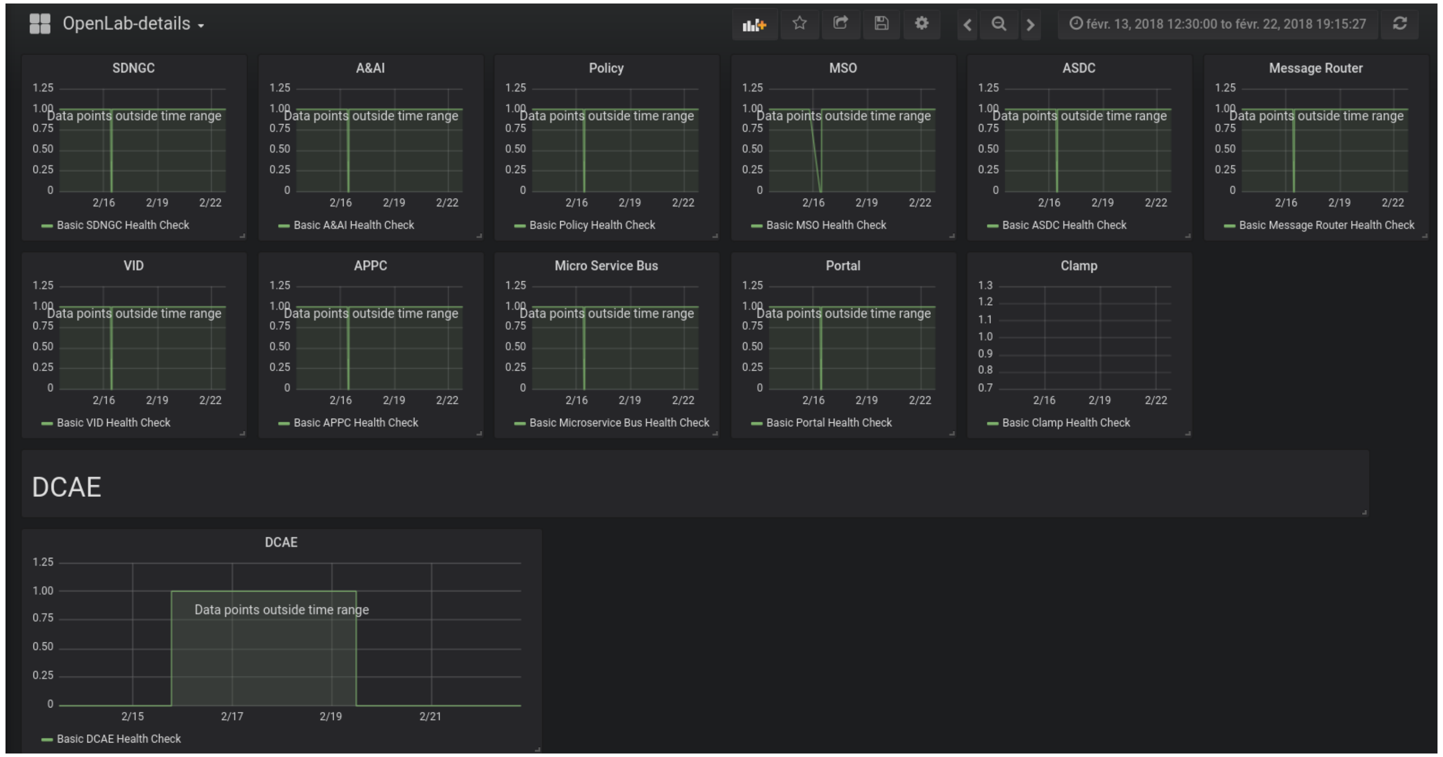
128 lines (125 sloc) | 5.22 KB

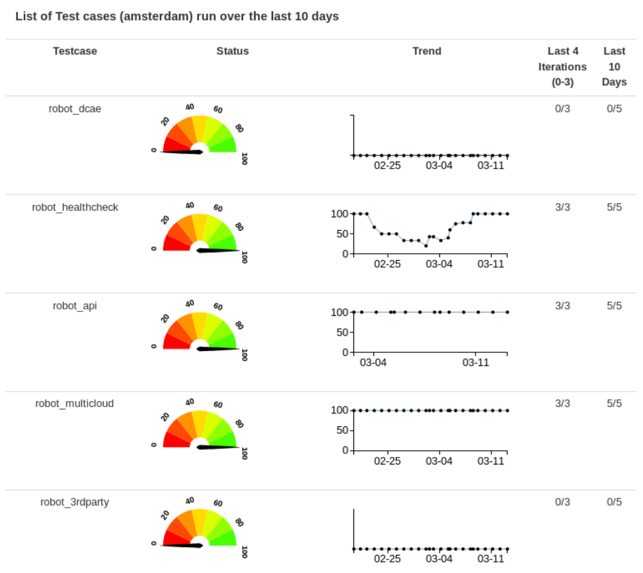
```yaml
---
tiers:
    -
        name: onap
        order: 1
        ci_loop: '(daily)|(weekly)'
        description: >-
            Set of basic Functional tests to validate the ONAP installation.
        testcases:
            -
                case_name: robot_healthcheck
                project_name: functest
                criteria: 100
                blocking: true
                description: >-
                    This test case verifies the basic ONAP API: appc, sdnc,so,
                    vid, ....based on the default robot tests
                dependencies:
                    installer: ''
                    scenario: ''
                run:
                    module: 'xtesting.core.robotframework'
                    class: 'RobotFramework'
                    args:
                        suites:
                            - /var/opt/OpenECOMP_ETE/robot/testsuites/health-check.robot
                        include:
                            - core
                        variablefile:
                            - '/share/config/integration_robot_properties.py'
                            - '/share/config/integration_vm_properties.py'
                            - '/share/config/integration_preload_parameters.py'

            -
                case_name: robot_api
                project_name: functest
                criteria: 100
                blocking: false
                description: >-
```

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




List of Test cases (amsterdam) run over the last 10 days

Testcase	Status	Trend	Last 4 Iterations (0-3)	Last 10 Days
robot_dcae			0/3	0/5
robot_healthcheck			3/3	5/5
robot_api			3/3	5/5
robot_multicloud			3/3	5/5
robot_3rdparty			0/3	0/5

Benefits for the LFN projects

- Xtesting allows a proper design inside OPNFV

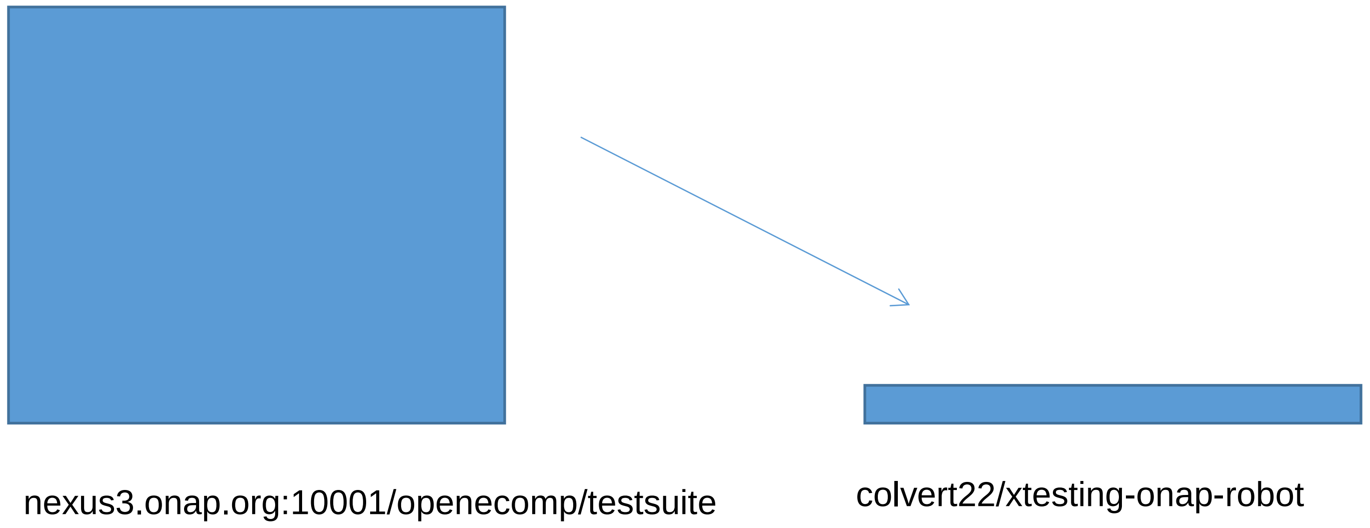
- Xtesting and Functest help other LFN projects:
  - verifying the infrastructure on top of which the components are deployed
  - ease verifying the components as well in the same CI/CD toolchain

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 ;-)