OPNFV User Guide

Release arno.2015.1.0 (71fa5c6)

OPNFV

February 01, 2016

CHAPTER ONE

ABSTRACT

OPNFV is a collaborative project aimed at providing a variety of virtualisation deployments intended to host applications serving the networking and carrier industry. This document provides guidance and instructions for using platform features designed to support these applications, made available in the Brahmaputra release of OPNFV.

This document is not intended to replace or replicate documentation from other open source projects such as OpenStack or OpenDaylight, rather highlight the features and capabilities delivered through the OPNFV project.

CHAPTER TWO

OVERVIEW

OPNFV provides a variety of virtual infrastructure deployments designed to host virtualised network functions (VNFs). This guide intends to help users of the platform leverage the features and capabilities delivered by the OPNFV project.

OPNFV Continuous Integration builds, deploys and tests combinations of virtual infrastructure components in what are defined as scenarios. A scenario may include components such as OpenStack, OpenDaylight, OVS, KVM etc. where each scenario will include different source components or configurations. Scenarios are designed to enable specific features and capabilities in the platform that can be leveraged by the OPNFV user community.

2.1 OPNFV Scenarios

Each OPNFV scenario provides unique features and capabilities, it is important to ensure you have a scenario deployed on your infrastructure that provides the right capabilities for your needs before working through the user guide.

This user guide outlines how to work with key components and features in the platform, each feature description section will indicate the scenarios that provide the components and configurations required to use it.

Each scenario provides a set of platform capabilities and features that it supports. It is possible to identify which features are provided by reviewing the scenario name, however not all features and capabilities are discernible from the name itself.

2.1.1 Scenario Naming

In OPNFV, scenarios are identified by short scenario names. These names follow a scheme that identifies the key components and behaviours of the scenario, the rules for scenario naming are as follows:

> os-[controller]-[feature]-[mode]-[option]

For example: *os-nosdn-kvm-noha* provides an OpenStack based deployment using neutron including the OPNFV enhanced KVM hypervisor.

The [feature] tag in the scenario name describes the main feature provided by the scenario. This scenario may also provide support for advanced fault management features which is not apparent in the scenario name. The following section describes the features available in each scenario.

2.1.2 Brahmaputra feature support matrix

The following table provides an overview of the available scenarios and supported features in the Brahmaputra release of OPNFV.

For details on which scenario's are best for you and how to install and configure them on your infrastructure the OPNFV Configuration guide provides a definitive reference.

The user guide will describe how to enable and utilise features and use cases implemented and tested on deployed OPNFV scenarios. For details of the use cases and tests that have been run you should check the validation procedures section of the features configuration guide. This will provide information about the specific use cases that have been validated and are working on your deployment.

2.2 General usage guidelines

The user guide for OPNFV features and capabilities provide step by step instructions for using features that have been configured according to the installation and configuration instructions.

This guide is structured in a manner that will provide usage instructions for each feature in its own section. Identify the feature capability you would like to leverage and read through that user guide section to understand the available usage. The combination of platform features, if available in a given scenario and not otherwise indicated, should function by following each features section. Dependencies between features will be highlighted in the user guide text.

You may wish to use the platform in a manner that the development team have not foreseen, or exercise capabilities not fully validated on the platform. If you experience issues leveraging the platform for the uses you have envisioned the OPNFV user mailing list provides a mechanism to establish a dialog with the community to help you overcome any issues identified.

It may be that you have identified a bug in the system, or that you are trying to execute a use case that has not yet ben implemented. In either case OPNFV is in essence a development project looking to ensure the required capabilities for our users are available.

CHAPTER

THREE

USAGE OF PLATFORM COMPONENTS

This section of the user guide provides general system component user guides and references. This provides user information for common components of the platform where as a user you may need to perform operations, depending on the scenario deployed and types of activities you are doing.

3.1 OpenStack User Guide

Add openstack specific user guide information here. Needs to include OPNFV default ports and addresses and references to upstream documentation applicable for an OPNFV deployment.

3.2 OpenDaylight User Guide

Add opendaylight specific user guide information here. Needs to include OPNFV default ports and addresses and references to upstream documentation applicable for an OPNFV deployment.

3.3 ONOS User Guide

Add ONOS specific user guide information here. Needs to include OPNFV default ports and addresses and references to upstream documentation applicable for an OPNFV deployment.

3.4 OVS User Guide

Add OVS specific user guide information here. Needs to include OPNFV default ports and addresses and references to upstream documentation applicable for an OPNFV deployment.

3.5 Description of the test cases

Functest is an OPNFV project dedicated to functional testing. In the continuous integration, it is launched after an OPNFV fresh installation. The Functest target is to verify the basic functions of the infrastructure.

Functest includes different test suites which several test cases within. Test cases are developed in Functest and in feature projects.

The current list of test suites can be distributed in 3 main domains:

```
+----------------+----------------+-------------------------------------------+
| Domain         | Test suite     | Comments                                  |
+================+================+===========================================+
|                | vPing          | NFV "Hello World"                         |
|                +----------------+-------------------------------------------+
|    VIM         | vPing_userdata | Ping using userdata and cloud-init        |
|                |                | mechanism                                 |
|                +----------------+-------------------------------------------+
|(Virtualised    | Tempest        | OpenStack reference test suite `[2]`_     |
| Infrastructure +----------------+-------------------------------------------+
| Manager)       | Rally scenario | OpenStack testing tool testing OpenStack  |
|                |                | modules `[3]`_                            |
+----------------+----------------+-------------------------------------------+
|                | OpenDaylight   | Opendaylight Test suite                   |
|                +----------------+-------------------------------------------+
| Controllers    | ONOS           | Test suite of ONOS L2 and L3 functions    |
|                +----------------+-------------------------------------------+
|                | OpenContrail   |                                           |
+----------------+----------------+-------------------------------------------+
| Features       | vIMS           | Show the capability to deploy a real NFV  |
|                |                | test cases.                               |
|                |                | The IP Multimedia Subsytem is a typical   |
|                |                | Telco test case, referenced by ETSI.      |
|                |                | It provides a fully functional VoIP System.|
|                +----------------+-------------------------------------------+
|                | Promise        | Resource reservation and management project|
|                |                | to identify NFV related requirements and  |
|                |                | realize resource reservation for future   |
|                |                | usage by capacity management of resource  |
|                |                | pools regarding compute, network and      |
|                |                | storage.                                  |
|                +----------------+-------------------------------------------+
|                | SDNVPN         |                                           |
+----------------+----------------+-------------------------------------------+
```

Most of the test suites are developed upstream. For example, Tempest is the OpenStack integration test suite. Functest is in charge of the integration of different functional test suites.

The Tempest suite has been customized but no new test cases have been created. Some OPNFV feature projects (e.g. SDNVPN) have created Tempest tests cases and pushed to upstream.

The tests run from CI are pushed into a database. The goal is to populate the database with results and to show them on a Test Dashboard.

There is no real notion of Test domain or Test coverage yet. Basic components (VIM, controllers) are tested through their own suites. Feature projects also provide their own test suites.

vIMS test case was integrated to demonstrate the capability to deploy a relatively complex NFV scenario on top of the OPNFV infrastructure.

Functest considers OPNFV as a black box. OPNFV, since Brahmaputra, offers lots of possible combinations:

- 3 controllers (OpenDayligh, ONOS, OpenContrail)
- 4 installers (Apex, Compass, Fuel, Joid)

However most of the tests shall be runnable on any configuration.

3.6 Executing the functest suites

3.6.1 Manual testing

Once the Functest docker container is running and Functest environment ready (through /home/opnfv/repos/functest/docker/prepare_env.sh script), the system is ready to run the tests.

The script *run_tests.sh* is located in $repos_dir/functest/docker and it has several options:

```
./run_tests.sh -h
Script to trigger the tests automatically.

usage:
    bash run_tests.sh [--offline] [-h|--help] [-t <test_name>]

where:
    -h|--help         show this help text
    -r|--report       push results to database (false by default)
    -n|--no-clean     do not clean up OpenStack resources after test run
    -t|--test         run specific set of tests
      <test_name>     one or more of the following: vping,vping_userdata,odl,rally,tempest,vims,onos,

examples:
    run_tests.sh
    run_tests.sh --test vping,odl
    run_tests.sh -t tempest,rally --no-clean
```

The *-r* option is used by the Continuous Integration in order to push the test results into a test collection database, see in next section for details. In manual mode, you must not use it, your try will be anyway probably rejected as your POD must be declared in the database to collect the data.

The *-n* option is used for preserving all the existing OpenStack resources after execution test cases.

The *-t* option can be used to specify the list of test you want to launch, by default Functest will try to launch all its test suites in the following order vPing, odl, Tempest, vIMS, Rally. You may launch only one single test by using *-t <the test you want to launch>*.

Within Tempest test suite you can define which test cases you want to execute in your environment by editing test_list.txt file before executing *run_tests.sh* script.

Please note that Functest includes cleaning mechanism in order to remove everything except what was present after a fresh install. If you create your own VMs, tenants, networks etc. and then launch Functest, they all will be deleted after executing the tests. Use the *–no-clean* option with run_test.sh in order to preserve all the existing resources. However, be aware that Tempest and Rally create of lot of resources (users, tenants, networks, volumes etc.) that are not always properly cleaned, so this cleaning function has been set to keep the system as clean as possible after a full Functest run.

You may also add you own test by adding a section into the function run_test().

3.6.2 Automated testing

As mentioned in *[1]*, the *prepare-env.sh* and *run_test.sh* can be executed within the container from jenkins. 2 jobs have been created, one to run all the test and one that allows testing test suite by test suite. You thus just have to launch the acurate jenkins job on the target lab, all the tests shall be automatically run.

When the tests are automatically started from CI, a basic algorithm has been created in order to detect whether the test is runnable or not on the given scenario. In fact, one of the most challenging task in Brahmaputra consists in dealing with lots of scenario and installers. Functest test suites cannot be systematically run (e.g. run the ODL suite on an ONOS scenario).

CI provides several information:

- The installer (apex|compass|fuel|joid)
- The scenario [controller]-[feature]-[mode] with
  - controller = (odl|onos|ocl|nosdn)
  - feature = (ovs(dpdk)|kvm)
  - mode = (ha|noha)

Constraints per test case are defined in the Functest configuration file /home/opnfv/functest/config/config_functest.yaml:

```
test-dependencies:
   functest:
       vims:
           scenario: '(ocl)|(odl)|(nosdn)'
       vping:
       vping_userdata:
           scenario: '(ocl)|(odl)|(nosdn)'
       tempest:
       rally:
       odl:
           scenario: 'odl'
       onos:
           scenario: 'onos'
       ....
```

At the end of the Functest environment creation (prepare_env.sh see **'[1]'_**), a file (/home/opnfv/functest/conf/testcase-list.txt) is created with the list of all the runnable tests. We consider the static constraints as regex and compare them with the scenario. For instance, odl can be run only on scenario including odl in its name.

The order of execution is also described in the Functest configuration file:

```
test_exec_priority:

   1: vping
   2: vping_userdata
   3: tempest
   4: odl
   5: onos
   6: ovno
   7: doctor
   8: promise
   9: odl-vpnservice
   10: bgpvpn
   11: openstack-neutron-bgpvpn-api-extension-tests
   12: vims
   13: rally
```

The tests are executed as follow:

- Basic scenario (vPing, vPing_userdata, Tempest)
- Controller suites: ODL or ONOS or OpenContrail
- Feature projects
- vIMS

- Rally (benchmark scenario)

At the end of an automated execution, everything is cleaned. We keep only the users/networks that have been statically declared in 'https://git.opnfv.org/cgit/functest/tree/testcases/VIM/OpenStack/CI/libraries/os_defaults.yaml'_

CHAPTER

FOUR

USING BRAHMAPUTRA FEATURES

This section of the user guide provides feature specific user guides and references. This provides user information when leveraging the features in the platform where as a user you may need to perform operations, many operation in this section refer to the guides in the general system usage section.