Quick Start Guide for Running Yardstick*/NSB for NFVI Characterization

Version 1.0 from July 3, 2018

1 Introduction ................................................................................................................. 4
  1.1 Scope ......................................................................................................................... 4
  1.2 Logical Architecture ................................................................................................. 5
  1.3 Hardware Requirements ......................................................................................... 6
  1.4 Software Requirements ........................................................................................... 7
  1.5 Pre-requisites on the Jump Server, the TG, and the SUT ........................................ 7
    1.5.1 Network Connectivity ......................................................................................... 7
    1.5.2 Check Package Sources (Optional) ...................................................................... 7
    1.5.3 Add Additional Packages (Optional) .................................................................... 8
    1.5.4 SSH Root Logon .................................................................................................... 8
    1.5.5 Add a User as sudo without a Password ................................................................. 8
2 Bare-metal and standalone virtualized environments ............................................... 8
  2.1 Installation .............................................................................................................. 8
  2.2 Check Yardstick Installation .................................................................................... 9
  2.3 Post-installation: Grafana* and InfluxDB* (Jump Server Only) ............................ 9
    2.3.1 Yardstick Container Operations .......................................................................... 9
    2.3.2 InfluxDB .............................................................................................................. 9
    2.3.3 Grafana ............................................................................................................. 11
  2.4 Bare-Metal Environment ....................................................................................... 14
    2.4.1 Test Configuration ............................................................................................ 15
    2.4.2 Post-installation (SUT and TG) .......................................................................... 15
    2.4.3 NFVI Test Execution .......................................................................................... 16
    2.4.4 Viewing Results ............................................................................................... 18
  2.5 Standalone Virtualization Environment ................................................................ 23
    2.5.1 SR-IOV Software Requirements ................................................................. 23
    2.5.2 Prepare the System for SR-IOV tests .............................................................. 23
Yardstick*/NSB Quick-Start Guide (QSG)

2.5.3 Manual Preparation of a VM Disk Image ................................................................. 26
2.5.4 Collecting Data from Tests ..................................................................................... 27

3 Managed Virtualization Environment: OpenStack* ..................................................... 28
3.1 Principles of Operation ......................................................................................... 28
3.2 Deployment Architecture ....................................................................................... 28
3.3 Yardstick/NSB Network Architecture ..................................................................... 29
  3.3.1 OpenStack Network Architecture ...................................................................... 29
  3.3.2 NSB Network Architecture ............................................................................. 32
  3.3.3 Reduced NSB Architecture for NFVI Testing ............................................... 33
3.4 Test Execution Architecture .................................................................................. 33
3.5 Installation Procedures ......................................................................................... 35
  3.5.1 Server Capacity ............................................................................................... 35
  3.5.2 Software Tool Installation .............................................................................. 36
  3.5.3 OpenStack and Docker* Installation .............................................................. 36
  3.5.4 Yardstick Image/Container Installation for OpenStack .................................. 46

4 NFVI Characterization ............................................................................................. 49
4.1 Prerequisites and Requirements ........................................................................... 50
4.2 KPI Collection ....................................................................................................... 51
4.3 VNF Selection ....................................................................................................... 52
4.4 Configuration Files ............................................................................................... 53
4.5 Execute NFVI Test Cases ...................................................................................... 55
  4.5.1 List of NFVI Test Cases ................................................................................ 55
  4.5.2 Execute an NFVI Test Case ........................................................................... 57
4.6 Miscellaneous Procedures ...................................................................................... 57
4.7 PROX Application ................................................................................................ 60
  4.7.1 Execute PROX ................................................................................................ 60

5 Appendixes ............................................................................................................. 62
5.1 /proc/cpuinfo Example ......................................................................................... 62
5.2 /proc/meminfo Example ....................................................................................... 62
5.3 Controller Docker Images .................................................................................... 63
5.4 Yardstick Command-Line Output Example ........................................................... 64
5.5 Yardstick Installation Commands on Clean Ubuntu 16.04* ................................................................. 65
6 Troubleshooting and Known Issues ........................................................................................................... 67
6.1 Version: stable/euphrates ........................................................................................................................ 67
7 References ................................................................................................................................................... 69
Abstract

This is a quick-start guide (QSG) to assist in the setup of a Yardstick*/Network Services Benchmarking (NSB) cluster for evaluation and training purposes.

1 Introduction

Yardstick/NSB is a test framework for measuring the performance of a virtual network function (VNF) in a network functions virtualization (NFV) environment. The default Yardstick/NSB use case is to onboard a VNF, source and sink traffic to it via traffic generators corresponding to the functionality of the VNF, and measure a variety of key performance indicators (KPIs) during VNF execution. The stream of KPI data is stored in a database, and it is visualized in a performance-visualization dashboard.

Yardstick/NSB can also be used to characterize the performance of an NFV infrastructure (NFVI)—the assemblage of servers, routers, and switches on which the NFV system runs. This is done by running a special test VNF, called PROX, which implements a suite of test cases and visualizes the output data of the test suite. The PROX test cases implement various execution kernels found in real-world VNFs, and the output of the test cases provides an indication of the fitness of the infrastructure for running NFV services, in addition to indicating potential performance optimizations for the NFVI.

Yardstick/NSB is based on the Yardstick open source project carried out by the Open Platform for NFV* (OPNFV*) project Yardstick. The goal of the Yardstick project is to verify infrastructure compliance from the perspective of a VNF. Yardstick/NSB is so named because the NSB part, designed to measure performance, is intended for contribution to the Yardstick project, and it will be maintained by the Yardstick community; however, as of this writing, the full adoption of NSB functionality has not yet happened.

1.1 Scope

The purpose of this QSG is to provide an understanding of Yardstick/NSB and an installation procedure for creating a minimal-scale test system, capable of running the NFVI test cases and visualizing the output. This guide focuses on NFVI characterization. In order to follow the QSG in detail, you will require three servers and a 10 gigabit-per-second (Gbps) Ethernet switch, and the servers will need access to the Internet.

Yardstick/NSB can be run in three environments or contexts: native Linux* (that is, “bare metal“), standalone virtualized environment (a VNF running in a virtual machine [VM], but without a full virtualization environment), and managed virtualized environment (a VNF and test framework running in VMs managed by an environment such as VMware NSX* or an open source or commercial release of OpenStack*). The procedures for deploying these three environments differ significantly from each other.

The three NFVI contexts are described in this document: section 2 guides you through the installation of NSB on a bare-metal or a standalone virtualized environment and section 3 concentrates on the OpenStack context. For the OpenStack context, it is assumed that you have installed a minimal configuration of the community OpenStack Ocata* and Kolla* release, in addition to Docker Community Edition* (Docker CE*),
as described in the Kolla-Ansible deployment guide. You will then be guided through a minimal installation of Yardstick/NSB, including the VM images of the PROX test VNF and its test cases.

This QSG will not teach you the fundamentals of OpenStack, Docker, NFV architecture, Yardstick architecture, or VNF onboarding. These are each complex topics in their own right, and you’re urged to read the references included at the end of this document for more information. If you’re unexperienced with Linux, OpenStack, Docker, or NFV, you might find it difficult to follow the educational materials and procedures contained in this QSG.

1.2 Logical Architecture

Yardstick/NSB is a test framework. It implements a test system (TS) that drives traffic to a system under test (SUT) and measures the SUT’s response. The response is measured both by consuming and measuring traffic received from the SUT (for example, measuring throughput, latency, jitter, and dropped packets). The response is also measured by collecting measurements from instrumentation installed in the SUT (for example, measuring CPU, memory, and network utilization). Figure 1 depicts the logical architecture of Yardstick/NSB.

The following subsection describes the logical architecture of Yardstick/NSB and the NFVI in which it is deployed.

The logical architecture of the Yardstick/NSB system consists of a TS and an SUT. The SUT can also be referred to as a VNF under test. The TS includes a control component, which orchestrates the execution of a performance test; a traffic generator, which sources and sinks traffic to and from the SUT and takes measurements; and a data-collection component, which collects KPI information, stores it in a database, and provides visualizations of the KPIs.

The control component is based on the OPNFV Yardstick project. Yardstick is designed to be a functional testing environment for VNFs, testing VNF onboarding, instantiation, and other use cases. Launching a test
VNF and traffic generator is done by processing template files that describe how to configure the VNF, traffic generator, and test cases.

If the SUT is characterizing a VNF (which is the default use case for Yardstick/NSB), the NFVI (the physical servers, switches, routers, operating systems, and virtualization environments) is also part of the SUT. The NFVI is not depicted in Figure 1, but it will be shown in later sections of the QSG.

If the SUT is characterizing an NFVI, then the TS includes a test VNF designed specifically to measure aspects of the NFVI when it is executed, similar to the way that CPU-based benchmark suites, such as the Standard Performance Evaluation Corporation* SPECmark* benchmarks, include workloads that characterize commercial software (for example, integer arithmetic, floating point arithmetic, pattern matching, and linear algebra). In this case, the test VNF is considered to be “fixed,” as part of the TS, and the NFVI is the SUT.

Corresponding to the test VNF is a traffic generator, which sources and sinks traffic to and from the test VNF according to each of the test workloads. For a given test run, the test VNF and traffic generator are configured by the TS to execute a workload, and they source and sink traffic corresponding to the workload.

1.3 Hardware Requirements

This subsection describes the logical architecture of Yardstick/NSB, and the NFVI in which it is deployed.

In general, there are three roles required to run the tests:

- A jump server, used for running Yardstick/NSB scripts, a database (InfluxDB*), and the dashboard (Grafana*)
- An SUT
- A test generator (TG)

Each role can be realized with either a physical machine (bare metal) or a VM. In case of limited resources, it is also possible to merge some of the roles—for example, a jump server with a TG—but for most cases, it is not recommended because such configuration might affect test results. This QSG will concentrate on the installation using three servers/roles.

Figure 2. NSB high-level overview shows the configuration of a bare-metal environment in which each role is realized by using a separate physical machine.

Figure 2. NSB high-level overview

Exact hardware requirements might change over time as NSB evolves and new test cases are added. For this QSG’s purposes, the following hardware is required:
- **Disk**: A minimum of 20 GB is required.
- **Memory**: A minimum of 8 GB is required.
- **CPU**: A minimum of eight cores are needed on both the TG and the SUT.
- **Network interface controller (NIC)**: Data Plane Development Kit (DPDK)-supported NIC. On the TG and the SUT, at least one network interface (1 gigabit Ethernet [GbE]) is necessary for management and one interface for the data plane (usually 10 or 40 GbE). The jump server only requires the management interface.

### 1.4 Software Requirements

The only software requirement is Ubuntu 16.04* for each server/role. The software packages required for installation are already included in the Ubuntu distribution or installed after running the setup script.

### 1.5 Pre-requisites on the Jump Server, the TG, and the SUT

#### 1.5.1 Network Connectivity

- An Internet connection is needed for package installation. In order to verify connectivity, ping a web site; for instance google.com.
- A management network is needed for Secure Shell (SSH) connectivity between the servers/roles. In order to verify connectivity, log on using an SSH client to each role TG and SUT from the Yardstick/NSB jump server.
- Test the data network’s connectivity. All network interfaces that take part in Yardstick/NSB tests must be connected to a 10/40 Gbps switch or connected back to back between physical machines. In case of limited resources, it is possible to use the management network for this purpose, but it is not recommended due to the low performance of such networks and other factors that might influence the results.
- HTTP/HTTPS connectivity to package repositories is required. If your servers are behind the proxy and firewall, you need to set up proxy system variables:

  ```bash
  user@NSB:~ export https_proxy=https://mysshproxyserver.com:mysshproxyport
  user@NSB:~
  ```

  To guarantee that those environment variables are not cleared when running sudo, run visudo.

  ```bash
  user@NSB:~ sudo visudo
  ```

  Add the following line after “defaults env_reset”:

  ```bash
  Defaults env_keep = "http_proxy https_proxy"
  ```

#### 1.5.2 Check Package Sources (Optional)

By default, there are no additional actions required after installation of Ubuntu 16.04. However, in some circumstances there might be a need to check or modify apt configuration files. If needed, add Debian* repositories to the sources.list file.

```bash
user@NSB:~ sudo vi /etc/apt/sources.list
deb http://archive.ubuntu.com/ubuntu xenial main restricted universe multiverse
user@NSB:~
```

```bash
user@NSB:~ sudo apt-get update
```
1.5.3 Add Additional Packages (Optional)
Depending on the installation process or individual needs, additional packages can be installed for convenience:

```bash
user@NSB:# sudo apt-get update
user@NSB:# sudo apt-get install openssh-server xfce4 xfce4-goodies tightvncserver tig \
apt-transport-https ca-certificates
```

1.5.4 SSH Root Logon
Permit SSH root logon by editing the sshd_config file and changing the “PermitRootLogin” value to yes.

```bash
user@NSB: # sudo vi /etc/ssh/sshd_config
…
PermitRootLogin yes
…
user@NSB: # sudo service ssh restart
```

1.5.5 Add a User as sudo without a Password
To elevate Linux user privileges, edit the /etc/sudoers file by adding a line that reads `username ALL=(ALL) NOPASSWD:ALL`

```bash
user@NSB: # sudo visudo
add at the end: username ALL=(ALL) NOPASSWD:ALL
```

2 Bare-metal and standalone virtualized environments

2.1 Installation
This document concentrates on using the stable/euphrates version of NSB. It can be obtained from OPNFV and installed through the following commands:

```bash
user@NSB: # git clone https://gerrit.opnfv.org/gerrit/yardstick
user@NSB: # cd yardstick
user@NSB: # git checkout stable/euphrates
```

To install a Yardstick Docker container Euphrates release, it is necessary to edit the `/yardstick/ansible/nsb_setup.yml` file and replace the following line:

```
"image: opnfv/yardstick:stable"
```

by:

```
"image: opnfv/yardstick:opnfv-5.1.0"
```

Issue the following command:

```bash
user@NSB: # sudo ./nsb_setup.sh
```

This will execute commands from:

- ./ansible/nsb_setup.yml
- ./ansible/ubuntu_server_baremetal_deploy_samplevnfs.yml

Executing this command will download any Linux packages required for the installation, DPDK, and Trex*, will build a Docker image of Yardstick, and will then run the Yardstick container. The command will run for
more than a dozen minutes and will generate output to the console. Note that the action “starting Yardstick container” takes time—it can take more than 15 minutes to complete.

The /var/lib/tmp (or /var/lib/docker/tmp) directory shows files being downloaded (with more than 600 MB to download).

2.2 Check Yardstick Installation

When the installation is successful, check that the Yardstick image is available with the following command:

```
user@NSB:# docker images
```

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>opnfv/yardstick</td>
<td>opnfv-5.1.0</td>
<td>05012143ffcb</td>
<td>7 hours ago</td>
<td>1.96 GB</td>
</tr>
</tbody>
</table>

Then issue the following command:

```
user@NSB:# docker ps
```

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE COMMAND</th>
<th>CREATED</th>
<th>STATUS</th>
<th>PORTS</th>
<th>NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>9b014d119462</td>
<td>opnfv/yardstick:opnfv-5.1.0</td>
<td>1s ago</td>
<td>Up</td>
<td>5000/tcp</td>
<td>yardstick</td>
</tr>
</tbody>
</table>

2.3 Post-installation: Grafana* and InfluxDB* (Jump Server Only)

All results of NSB tests are displayed on the console by default. To organize them and get better visualization of the results, you might need to install InfluxDB and the Grafana dashboard.

2.3.1 Yardstick Container Operations

In order to connect to the Yardstick/NSB container, run following command:

```
user@NSB:# sudo docker exec -it yardstick /bin/bash
```

After this command, you can run other commands inside the Yardstick container. Exit the container using the following command:

```
root@ab12cd34ef56:# exit
```

2.3.2 InfluxDB

- Start the InfluxDB container (from the Yardstick container) with the following command. Note that some warnings might be displayed.

```
root@ab12cd34ef56:# yardstick env influxdb
```

- Find the name of the InfluxDB container from the host with the following command:

```
user@NSB:# sudo docker ps
```

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE COMMAND</th>
<th>CREATED</th>
<th>STATUS</th>
<th>PORTS</th>
<th>NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>b32cf7c26ef61</td>
<td>tutum/influxdb:0.13</td>
<td>1s ago</td>
<td>Up</td>
<td>0.0.0.0:8083-&gt;8083/tcp, 0.0.0.0:8086-&gt;8086/tcp</td>
<td>practical_bassi</td>
</tr>
<tr>
<td>9bcf3e19462</td>
<td>opnfv/yardstick:stable</td>
<td>1s ago</td>
<td>Up</td>
<td>5000/tcp</td>
<td>yardstick</td>
</tr>
</tbody>
</table>

Notice that the InfluxDB container has an arbitrary name, “practical_bassi.” The Docker command that built the container did not supply a name, so a name is generated for you.

- Connect to the InfluxDB container (from the host):
Yardstick*/NSB Quick-Start Guide (QSG)

```
user@NSB:~# sudo docker exec -it practical_bassi /bin/bash
```

- Find the IP address of the container (from the InfluxDB container):

```
root@b32cf7c26e61:# ip a
22: eth0@if23: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
inet 172.17.0.3/16 scope global eth0
  valid_lft forever preferred_lft forever
inet6 fe80::42:acff:fe11:3/64 scope link
  valid_lft forever preferred_lft forever
```

- Configure Yardstick (from the Yardstick container):

```
root@ab12cd34ef56:# vi /etc/yardstick/yardstick.conf

[DEFAULT]
debug = False
dispenser = influxdb

[dispatcher_http]
timeout = 5
target = http://127.0.0.1:8000/results

[dispatcher_file]
file_path = /tmp/yardstick.out
max_bytes = 0
backup_count = 0

[dispatcher_influxdb]
timeout = 5
target = http://172.17.0.3:8086 # Set the target IP of the influx container
db_name = yardstick
username = root
password = root

[nsb]
trex_path = /opt/nsb_bin/trex/scripts
bin_path = /opt/nsb_bin
```

Check InfluxDB using a web browser; browse to http://your_NSB_host_IP:8083.

![InfluxDB in a web browser](image)

**Figure 3. InfluxDB* in a web browser**

From the InfluxDB container, you can also log on to the database and check that the Yardstick database is available:

```
root@ab12cd34ef56:# influx
> use yardstick
> Using database yardstick
> quit
```
If you’d like to investigate this database further, visit InfluxDB for more information.

### 2.3.3 Grafana

Grafana is a visualization tool, used to retrieve data from the InfluxDB database and visualize it. Grafana works by defining “dashboards,” which are combinations of visualization panes (for example, line charts and gauges) and forms that assist the user in formulating SQL-like queries for InfluxDB.

In this step, a Docker container that runs Grafana is created, launched, and configured.

Creation and launch of the Grafana container is done via the following command:

```bash
root@ab12cd34ef56:~# yardstick env grafana
```

This might return a warning and an error. You can ignore the following error:

```
HTTPConnectionPool(host='172.17.0.4', port=3000): Max retries exceeded with url: /api/datasources (Caused by NewConnectionError('<requests.packages.urllib3.connection.HTTPConnection object at 0x7f40dc8091d0>: Failed to establish a new connection: [Errno 111] Connection refused',))
```

Check that Grafana is properly running using `docker ps` on the host:

```
CONTAINER ID     IMAGE                   COMMAND             CREATED             STATUS    PORTS                                           NAMES
df49907779      grafana/grafana:4.4.3  "/run.sh"            1 day ago           Up                    0.0.0.0:1948-3000/tcp   condescending_colden
b32cf7c26e      tutum/influxdb:0.13    "/run.sh"            1s ago              Up                    0.0.0.0:8083-8086/tcp   practical_bassi
9b014d1194      opnfv/yardstick:5.1.0  "/usr/bin/supervisord" 1 day ago           Up                    5000/tcp   yardstick
```

Check Grafana using web browser to navigate to `http://ip_address:1948`.

Configuration of the container is done by logging on to the Grafana user interface (UI).

Currently, the logon credentials are admin (user) and admin (password).

![Figure 4. Grafana logon page after successful installation](image)
After logging on to Grafana, you will be prompted to add a data source, which will be the InfluxDB container.

![Grafana Install Page](image)

Figure 5. Grafana* install page

Click **Add data source**, and then fill in the Add data source page as shown in Figure 6. Grafana* Add Data Source page.

The HTTP URL for the data source refers to the IP address of the InfluxDB database, which is found in 2.3.2.

The Basic Auth Details fields refer to the logon and password for Grafana, which are both admin. It is possible to change these later.

The InfluxDB details are the database name, logon, and password. The database name must be **yardstick**. The logon and password are found in the file `/etc/yardstick/yardstick.conf` in the Yardstick container, as shown in 2.3.
After clicking **Add**, the new data source is established and tested.
2.4 Bare-Metal Environment

This QSG assumes that for a bare-metal environment, you have servers connected like in Figure 8.

Figure 7. Data source established in Grafana*

Figure 8. Network diagram for a bare-metal environment
2.4.1 Test Configuration

Before starting any test, you have to adjust configuration parameters to reflect your infrastructure. Each test contains its own configuration files. You can modify them as in the example below:

```
root@ab12cd34ef56: cd yardstick/samples/vnf_samples/nsut/prox/
root@ab12cd34ef56: vi prox-baremetal-2.yaml
```

To get the test working, you need to modify the following parameters for the TG and the SUT:

- ip
- user
- password
- vpci

If a password is provided, “key_filename” must be commented out. Setting it to “” as in prox-baremetal-2.yaml results in an error like “file not found.”

Connect to the TG and the SUT manually to add them in the list of known hosts. Supposing that the IP addresses of the TG and the SUT are respectively 192.168.1.179 and 192.168.1.182, run:

- `ssh root@192.168.1.179` (answer yes to add to /root/.ssh/known_hosts)
- `ssh root@192.168.1.182` (answer yes to add to /root/.ssh/known_hosts)

Note: Before starting the test, you need to make sure that you have a clean configuration of your infrastructure and that there are no other settings left from tests you did in the past, such as DPDK bound interfaces on a host machine.

2.4.2 Post-installation (SUT and TG)

A Yardstick container itself theoretically is not necessary on the SUT and the TG, but nsb_setup.sh installs the necessary scripts (such as DPDK NIC binding scripts) and applications that need to run on the TG and the SUT.

After completing Yardstick/NSB installation on the TG and the SUT, you can stop the Yardstick container on the TG and the SUT using the following command on each:

```
user@NSB: docker container stop yardstick
user@NSB: docker ps
```

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th>CREATED</th>
<th>STATUS</th>
<th>PORTS</th>
<th>NAMES</th>
</tr>
</thead>
</table>

The VNF running for NFVI characterization (PROX) requires connection to TCP port 8474.

Configure the firewall to accept connection on TCP port 8474, or disable the firewall entirely. In order to disable a Linux firewall, issue the following commands:

```
user@NSB: sudo systemctl status ufw
user@NSB: sudo systemctl stop ufw
user@NSB: sudo systemctl disable ufw
```
2.4.3 NFVI Test Execution
This QSG provides different test cases, from the simplest L2 forwarding to a more complex one.

2.4.3.1 List of NFVI Test Cases
Each NFVI test case has a top-level context file defined; in the Yardstick container, there is a /home/opnfv/repos/yardstick/samples/vnf-samples/nsut/prox directory.

In order to list all available tests and VNFs, use the following commands:

```bash
root@ab12cd34ef56:/home/opnfv/repos/yardstick/yardstick/cmd/NSBperf.py --list-vnf
VNF :
================
1: udp_replay
2: router
3: prox
4: acl
5: vpe
6: cgnapt
7: vfw
8: ping
9: 2trex
```

The order of the VNFs might be different. More VNFs might be available in future releases.

```bash
root@ab12cd34ef56:/home/opnfv/repos/yardstick/yardstick/cmd/NSBperf.py --list-test
VNF : (prox)
tc_prox_baremetal_acl-2.yaml
tc_prox_baremetal_acl-4.yaml
tc_prox_baremetal_binsearch-2.yaml
tc_prox_baremetal_bng-4.yaml
tc_prox_baremetal_bng_qos-4.yaml
tc_prox_baremetal_buffering-1.yaml
tc_prox_baremetal_l2fwd-2.yaml
tc_prox_baremetal_l2fwd-4.yaml
tc_prox_baremetal_l2fwd_multiflow-2.yaml
tc_prox_baremetal_l2fwd_multiflow-4.yaml
```

2.4.3.2 Execute an NFVI Test Case
A test case is executed from inside the Yardstick container using the following Yardstick command:

```bash
root@ab12cd34ef56:/home/opnfv/repos/yardstick/yardstick/cmd/NSBperf.py --list-vnf
VNF :
================
1: udp_replay
2: router
3: prox
4: acl
5: vpe
6: cgnapt
7: vfw
8: ping
9: 2trex
```

For this command, `<contextfile>` is one of the options listed in Section 2.4.3.1.

When this command is executed:

- Debug output will be generated in standard output in the shell session in which the command was run. This output can be used for debugging purposes. A more detailed description of the output is beyond the scope of this QSG. We suggest you run the test and observe the output of the console.
• Test output will be generated in the /tmp/yardstick/yardstick.log file (in the Yardstick container). This output is consistent with the standard output data that the PROX test framework (upon which Yardstick/NSB NFVI testing is based) generates. It can be used to supplement the “official” data stream that Yardstick/NSB generates.
• KPIs will be streamed to the InfluxDB database. This data can be retrieved via manual database queries of the database, or it can be visualized by use of the Grafana tool.

2.4.3.2.1 Example: L2FWD

In order to correctly perform a test, investigate the test file, yardstick/samples/vnf_samples/nsut/prox/tc_prox_baremetal_l2fwd-2.yaml.

The basic configuration file name is listed in the context section:

```
root@ab12cd34ef56:~# vim tc_prox_baremetal_l2fwd-2.yaml
...
context:
  type: Node
  name: yardstick
  nfvi_type: baremetal
  file: prox-baremetal-2.yaml
```

It is necessary to update fields for tg_0 and VNF_0 in the file prox-baremetal-2.yaml with proper values for user, password, vpci, and local_mac.

```
root@ab12cd34ef56:~# vim prox-baremetal-1.yaml
...
nodes:
- name: "tg_0"
  role: TrafficGen
  ip: 1.1.1.1
  user: "root"
  ssh_port: "22"
  password: "r00t"
  interfaces:
    xe0:
      vpci: "0000:08:00.0"
      local_mac: "aa:cc:cc:dd:ee:ff"
      driver: "i40e"
      local_ip: "152.16.100.19"
      netmask: "255.255.255.0"
      dpdk_port_num: 0
```

• The total duration of the test can be changed in yardstick/samples/vnf_samples/nsut/prox/tc_prox_baremetal_l2fwd-2.yaml. This duration should be longer that the expected duration of the test. When the “duration” is elapsed, the test will stop the generator and the SUT, collect results, and update the dashboard.
• The packet sizes are listed in the yardstick/samples/vnf_samples/traffic_profiles/prox_binsearch.yaml file. You can specify a range of packet sizes to use (for instance, [64, 128, 256, 512, 1024, 1280, 1518]).
• In order to start one of the PROX tests, run the following commands:

```
root@ab12cd34ef56:~# cd yardstick/samples/vnf_samples/nsut/prox
root@ab12cd34ef56:~# yardstick task start tc_prox_baremetal_l2fwd-2.yaml
```
Tests might display the following “errors” while succeeding; those errors can be ignored, because they don’t have an impact on the test results.

An example of the output can be found in the Appendixes section.

Results are available for viewing in the Grafana dashboard. A report can also be generated using the following command:

```
root@ab12cd34ef56: #
yardstick report generate eb608a-c6c-4bc-9ca-162fa tc_prox_baremetal_l2fwd-2
```

2.4.4 Viewing Results

Yardstick*/NSB defines a collection of pre-built dashboards for the NFVI tests. Those dashboards are located in the “yardstick/dashboards” directory.

The pre-built dashboard files corresponding to the desired tests should be imported by clicking the Grafana logo, **Dashboards**, and then **Import**, as shown in Figure 9.

The dashboard files must reside on a system reachable by your web browser, as shown in Figure 10.
Indicate the database to use in the dashboard. The database created by Yardstick/NSB is called “yardstick,” as shown in Figure 11.
Assuming that you have recently run an NFVI test case, you can view it by clicking the **time range** icon in the upper-right corner and then selecting an appropriate time range, such as **last 1 hour**.

When the dashboard is loaded, the data display panel will appear, as shown in Figure 12.
The database queries used to generate a display can be viewed and edited by clicking an individual chart’s title (for example, “L2Fwd VNF stats”) and then selecting **Edit**.

The InfluxQL* queries used to generate the display are shown, and they can be edited by use of a form to assist with query syntax, as shown in Figure 13.

---

Figure 12. Grafana* L2FWD_2PORT dashboard
Figure 13. Grafana* with InfluxQL* query syntax
2.5 Standalone Virtualization Environment

2.5.1 SR-IOV Software Requirements
- Bridge for VMs to connect to external network; the bridge name must be set to **br-int**.
- Additional packages to create a guest image.
- A VM image built using Yardstick image-creation tools.

2.5.2 Prepare the System for SR-IOV tests

2.5.2.1 SUT (Host)

```bash
GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on iommu=pt hugepagesz=2M hugepages=8192 hugepagesz=1G hugepages=16"
```

Add a bridge on the host and assign an IP address:

```bash
user@NSB:~# sudo brctl addbr br-int
user@NSB:~# sudo brctl addif br-int <interface_name> && ip addr <ip_address>/<net_mask> dev <interface_name> && dhclient br-int
```

Install additional packages for Yardstick to create an image called **yardstick-image** including the samplevnf project, which is created from an Ubuntu Cloud Server* image. It is necessary to have sudo rights to use this tool.

Install several additional packages by using the command:

```bash
user@NSB:~# sudo apt-get update && sudo apt-get install -y qemu-utils kpartx libguestfs-tools
```

In order to build a Yardstick image, go to the Yardstick directory and use following commands:

```bash
user@NSB:~# export YARD_IMG_ARCH="amd64"
user@NSB:~# echo "Defaults env_keep += '\"YARD_IMG_ARCH\"' | sudo tee --append /etc/sudoers > /dev/null
user@NSB:~# sudo -EH tools/yardstick-img-modify tools/ubuntu-server-cloudimg-modify.sh
```

After building the image, use virt to disable unnecessary services that can cause boot-up errors. Disable the cloud-init service, which can cause huge delays during boot up, by editing the following file and setting the data sources to “none”:

```bash
user@NSB:~# virt-edit /tmp/workspace/yardstick/yardstick-image.img
/etc/cloud/cloud.cfg.d/90_dpkg.cfg
```

The next step is to allow logging on to the VM with a root account. By default, this is disabled for security reasons. Use virt-edit to edit the SSH configuration file and change PermitRootLogin to **yes**.

```bash
user@NSB:~# virt-edit /tmp/workspace/yardstick/yardstick-image.img /etc/ssh/sshd_config
```

By default, Ubuntu cloud does not have a configured root account password, which is necessary for the Yardstick test context file. It is possible to configure this with virsh-sysprep by using the following command:

```bash
user@NSB:~# virsh-sysprep --root-password password:new_password -a /tmp/workspace/yardstick/yardstick-image.img
```

Move the created image from /tmp/workspace to /var/lib/libvirt/images/Ubuntu.qcow2. You might also want to increase its size, which you can do with the following command:

```bash
qemu-img resize /var/lib/libvirt/images/Ubuntu.qcow2 +2G
```
Configure or disable a firewall on the SUT because it would prevent Dynamic Host Configuration Protocol (DHCP).

```
sudo iptables -X
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -P PREROUTING ACCEPT
iptables -t mangle -P FORWARD ACCEPT
iptables -t mangle -P OUTPUT ACCEPT
iptables -t mangle -F
iptables -t mangle -X
iptables -t filter -P INPUT ACCEPT
iptables -t filter -P FORWARD ACCEPT
iptables -t filter -P OUTPUT ACCEPT
iptables -t filter -F
iptables -t filter -X
```

2.5.2.2 Yardstick Container on Jump

Before executing the test, you can verify the VM template which is available at the following path: `/home/opnfv/repos/yardstick/yardstick/benchmark/contexts/standalone/model.py`

Copy the `trex_bm.yaml.sample` and `host_sriov.yaml` files to the location mentioned in the test-configuration file, and rename them accordingly.

For the `trex_bm.yaml.sample` file:

- The default location is `<yardstick>/etc/yardstick/nodes/standalone/trex_bm.yaml.sample`.
- Copy the file to `/etc/yardstick/nodes/standalone/`.
- Rename the file as `pod_trex.yaml`.

For the `host_sriov.yaml` file:

- The default location is `<yardstick>/etc/yardstick/nodes/standalone/host_sriov.yaml`.
- Copy the file to `/etc/yardstick/nodes/standalone`.

Ensure that the `pod_trex.yaml` and `host_sriov.yaml` files contain the correct information—such as IP, user, password, vpci, driver, and local_mac. Examples of content of such files are provided below, with key fields in bold.

**pod_trex.yaml:**
host_sriov.yaml:

nodes:

- name: sriov
  role: Srvio
  ip: 192.168.100.101
  user: ""
  password: ""

interfaces:

xe0:
  vpci: "0000:07:00.0"
  driver: i40e # default kernel driver
dpk_port_num: 0
  local_ip: "152.16.100.20"
  netmask: "255.255.255.0"
  local_mac: "00:00:00:00:00:01"

xe1:
  vpci: "0000:07:00.1"
  driver: i40e # default kernel driver
dpk_port_num: 1
  local_ip: "152.16.40.20"
  netmask: "255.255.255.0"
  local_mac: "00:00:00:00:00:02"

Check and update the context section in the chosen test case. For instance, properly configure the PCI addresses of the NIC on the host SUT, in addition to the user and password (as set by virt-sysprep) from the VM. By default, the IP address for the VM will be obtained by DHCP.
password: "" # update password
servers:
vnf:
    network_ports:
        mgmt:
            cidr: '1.1.1.61/24' # Update VM IP address, if static, <ip>/<mask> or if dynamic, <start of ip>/<mask>
            xe0:
                - uplink_0
            xe1:
                - downlink_0
networks:
    uplink_0:
        phy_port: "0000:05:00.0"
        vpci: "0000:00:07.0"
        cidr: '152.16.100.10/24'
        gateway_ip: '152.16.100.20'
    downlink_0:
        phy_port: "0000:05:00.1"
        vpci: "0000:00:08.0"
        cidr: '152.16.40.10/24'
        gateway_ip: '152.16.100.20'

2.5.3 Manual Preparation of a VM Disk Image
NSB needs a VM disk image to create a new VM, as indicated in the config file. One way to prepare a VM disk image was already described in section 2.5.2. As an alternative, this section describes the steps to create a VM disk image from scratch using the Ubuntu cloud image, which can be downloaded from the following location: [http://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img](http://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img)

In preparing the VM image for NSB, you’ll complete the following actions:

- Resize the partition and file system
- Change the root password
- Enable root logon through SSH
- Disable the cloud-init part trying to fetch user data from different cloud providers
- Install NSB
- Disable unattended upgrades for the kernel and modules

As the first operation, it is recommended that you resize the file system of the image because 2 GB might not be sufficient. Once that is done, set the root password.

```bash
qemu-img resize ubuntu.img 20G
modprobe nbd max_part=8
gemu-nbd -c /dev/nbd0 ubuntu.img
parted /dev/nbd0 rm 1
parted /dev/nbd0 mkpart -a optimal p ext4 0% 100%
e2fsck -fy /dev/nbd0p1
resize2fs /dev/nbd0p1
mount /dev/nbd0p1 /mnt
echo root:new_root_password | chpass -R /mnt
umount /mnt
gemu-nbd -d /dev/nbd0
```

Once the image is resized, it can be modified. You can continue modifications on a mounted image in the /mnt directory. Usually, the most convenient place to start is with the VM using this image. Perform
modifications on the live system, then store it for future use. You can use following command to create the VM:

```
virt-install --name nsb_VM --vcpu 4 --memory 8192 --network bridge=br-int,model=virtio --disk ubuntu.img --import --graphics vnc,listen="0.0.0.0" --noautoconsole
```

Once the VM starts, you need to log on to it as root and make further modifications. First, enable SSH root logon by changing the following line in the `/etc/ssh/sshd_config` file:

```
PermitRootLogin yes
PasswordAuthentication yes
```

Disable the cloud-init data source by changing the following line in the `/etc/cloud/cloud.cfg.d/90_dpkg.cfg` file:

```
datasource_list: [ None ]
```

Disable unattended upgrades by changing following line in the `/etc/apt/apt.conf.d/` file and replacing 1 with 0:

```
APT::Periodic::Unattended-Upgrade 0;
```

Install NSB by referring to the Yardstick/NSB installation chapter (2.1) in this QSG.

Once VM modification is done, you can keep the VM for your custom adjustments later, or you can delete it, leaving only the disk image with the following commands:

```
virsh shutdown nsb_VM
virsh undefine nsb_VM
```

### 2.5.4 Collecting Data from Tests

Test execution and data collection are described in sections 2.4.3 and 2.4.4 of this QSG.
3 Managed Virtualization Environment: OpenStack*

3.1 Principles of Operation

Yardstick/NSB is a test framework. It implements a TS, which drives traffic to an SUT and measures the SUT’s response. The response is measured both by consuming and measuring traffic received from the SUT (for example, measuring throughput, latency, jitter, dropped packets). The response is also measured by collecting measurements from instrumentation installed in the SUT (for example, measuring CPU, memory, and network utilization).

The following subsections describe the logical architecture of Yardstick/NSB and the NFVI in which it is deployed.

3.2 Deployment Architecture

“Deployment architecture” refers to the procedures and technology employed to realize an actual TS and SUT. Figure 14 depicts the deployment architecture of Yardstick/NSB, as used to perform NFVI characterization.

Figure 14. Yardstick*/NSB deployment architecture for NFVI

The deployment architecture depicts an OpenStack deployment on a control server and a group of one or more compute hosts. The traffic generator and SUT run as VMs running on the compute host(s), orchestrated via a Docker container running on the jump host. Docker and OpenStack interact with each other via the Kolla* service, one of the 100+ services currently included in the OpenStack project. Parts of the test system—the data-collection database, based on InfluxDB, and the dashboard display, based on Grafana—run in separate Docker containers on the jump host.

The Yardstick/NSB control component runs as an application in a Docker container. The use of Docker is a design pattern adopted by recent releases of the OpenStack project. Orchestrating the operation of an OpenStack system, as an NFV orchestrator or Yardstick/NSB do, is a complex process, and implementations employ dozens (or hundreds) of packages and libraries. Packaging a complex system in a
container allows a system such as NSB/Yardstick to be delivered in a robust manner. In the Ocata* release of OpenStack, the Kolla service allows the Yardstick/NSB control application to interact with OpenStack services to orchestrate performance tests. For more information, visit the following websites: Ocata and Kolla.

The Yardstick/NSB control component functions as a special-purpose NFV orchestrator whose job is to configure a test. The test consists of a VNF under test, a traffic generator, and a description of a system configuration and test plan. The system configuration describes how to configure the VNF, and the test plan describes how to execute a test (for example, ramp traffic intensity over the running time of the test until maximum performance is found, changing traffic intensity in a binary search procedure to find maximum performance). This information is described as a declarative “VNF model,” expressed in a collection of YAML documents, and a traffic model. Similarly describing the traffic type and test plan, Figure 15 represents the current hierarchy of YAML documents used to express a Yardstick/NSB test.

![YAML file hierarchy for a simple NFVI test case](image)

On one hand, when Yardstick/NSB is used to characterize commercial VNFs, the VNF model is shipped as part of a VNF deployment package and modified to align with the NFVI where it is executed. On the other hand, when Yardstick/NSB is used to characterize an NFVI, the VNF model and traffic model are provided, because they are part of the implementation of the NFVI test VNF.

### 3.3 Yardstick/NSB Network Architecture

This section presents considerations for configuring a server cluster to support Yardstick/NSB.

#### 3.3.1 OpenStack Network Architecture

There are many ways to configure an OpenStack network. This subsection will describe a configuration for small-scale Yardstick/NSB testing, and it will explain the design considerations leading to the choice of the small-scale configuration.

Figure 16 depicts an example small-scale network architecture. The physical hosts depicted include an OpenStack jump host, an OpenStack controller, and two OpenStack compute hosts.
The OpenStack jump host is used to stage installation of all the other software packages in the cluster, and it is the server from which the Yardstick/NSB controller is executed. Furthermore, a separate jump host insulates the system from changes in the OpenStack cluster that might require reinstallation of OpenStack, and therefore a reinstallation of other system components, including Yardstick/NSB.

Three subnets are shown connecting to the servers. The provider network is an OpenStack external network, through which traffic from the outside world can be routed to VMs running in OpenStack, and vice-versa. When a provider network is provisioned, OpenStack requires a pool of “floating” IP addresses that can be assigned to VMs. In the depicted network, OpenStack expects the gateway to provide the pool of addresses, which, in practical terms, requires that the OpenStack cluster be provisioned cooperatively with the IT managers of the infrastructure in which it is located.

The management network is the subnet over which the controller manages the OpenStack system. It carries OpenStack API traffic.

The tenant network is an optional network that can be provisioned to support services running on the cluster. An individual tenant network is owned by a single user (that is, “tenant”). For Yardstick/NSB testing, tenant networks are used to carry packet traffic between the traffic generator and the test VNF. In this network, the tenant network is a physical network, connected to NIC interfaces on the servers.

The access switch is a private physical switch used by OpenStack to provision its various networks. It can be configured in a number of ways, ranging from a “flat network” (that is, no virtual LANs [VLANs], with routing accomplished via IP addresses) or managed to provide segregated traffic for performance or security. In this network, for example, the provider network could be on “VLAN 0” and the other networks on multiple additional VLANs.

This network architecture is a common choice among OpenStack clusters, but for the purposes of this QSG, we chose to simplify it. We expect the Yardstick/NSB clusters constructed from this guide to be used often for educational purposes, and it would be preferable to reduce the dependency on the IT infrastructure in order to construct the cluster.

---

1 OpenStack documentation sometimes refers to OpenStack jump hosts as “deployment hosts.”
The first simplification is to isolate the provider network, as shown in Figure 17. Isolating the provider network. The major innovation is to no longer rely on the gateway for providing floating IP addresses—instead, this is carried out by the Neutron* (OpenStack networking) service. The gateway is still used to provide connectivity to the physical hosts for the end user, but this is a separate network.

In this configuration, a user logging on to the controller or compute hosts can “ping” VMs via the provider network. If the jump host is connected and routed, a user logging on to it can also ping VMs on the provider network.
If this cluster is to be used for educational purposes, it can be simplified further, as shown in Figure 18. If the compute server is sufficiently powerful, only one compute server is required. Similarly, if the OpenStack configuration is deemed stable, then it can be used as the jump host. Finally, no physical tenant networks are required, because the traffic generation and test VNF VMs will run on a single compute server. The final bill of materials is two servers (with two NICs each) and a (recommended 10 Gbps) switch.

![Reduced "educational" OpenStack* cluster](image)

**Figure 18. Reduced "educational" OpenStack* cluster**

### 3.3.2 NSB Network Architecture

Yardstick/NSB builds on the OpenStack architecture. Yardstick/NSB is required to support performance characterization in three contexts: bare metal, standalone virtualized, and managed virtualized. It is also required to support VNFs that require complex or stateful traffic. In order to support these additional requirements, the architecture requires additional physical hosts:

- **Yardstick jump host:** This host runs the Yardstick/NSB container, from which the traffic generator VM and the SUT VM are launched. The TS and the SUT VMs run on the compute host.
- **Replay host:** This host works with the traffic generator to generate complex or stateful traffic (for example, traffic in which end users interact with each other by sending traffic to each other through the SUT). Not all VNFs require a replay host; in particular, for NFVI testing, the replay host is not required.
3.3.3 Reduced NSB Architecture for NFVI Testing

While many of the procedures (such as physical networking and installation of Yardstick, InfluxDB, and Grafana) are valid for all of Yardstick/NSB operation, the primary purpose of this version of the QSG is training in NFVI testing. The reduced architecture for this purpose is shown in Figure 20.

3.4 Test Execution Architecture

Figure 21. OpenStack* network topology depicts an NFV network service (in this case, a test VNF and a traffic generator), as created by Yardstick/NSB. On the left side of the figure is a representation of the networks and NFV elements in the OpenStack dashboard. The code on the right side is the template, represented in YAML notation corresponding to the test VNF and networks.
The network service, in general, will carry a variety of traffic types, varying in source, destination, packet-size distribution, and packet arrival-time distribution. Examples of different traffic types are control protocols, voice packets, and HTTP packets. Traffic types produced and consumed by a VNF depend on the function that the VNF implements.

Prior to instantiating the VNFs, subnetworks are created and associated with different traffic types. There is no absolute rule concerning how to associate traffic with subnets; it is done either algorithmically by the orchestrator or manually by a human operator, based on considerations of bandwidth, latency, and redundancy.

In the example of Figure 21, the two VNFs are a four-port multi-protocol label switching (MPLS) VNF and a traffic generator that produces and consumes traffic to and from the MPLS VNF. For maximum throughput, each port of each VNF is connected to a separate subnet (the green, red, purple, and brown networks), which are configured as 10 Gbps data networks. The VNFs are also connected to a management subnet (the orange network), which is used by OpenStack to perform VM-orchestration actions.

In the figure, the blue network is a public network, connected to the external Internet. Traffic from the outside world reaches the NFV network service via this network. When the network service is instantiated, a router is added between the public network and the NFV management network.

These networks and VNFs are configured (at a high level) as follows:

- The network-service model describes the network requirements and VNF processing requirements. These are prepared by the developer of the VNF.
- The network service model is created algorithmically or manually, describing the logical connectivity of the network service (in this case, the topology is rather simple).
- An NFV model interpreter in Yardstick/NSB translates these models into an OpenStack template (as shown on the right side of the figure).
- OpenStack interprets the template and creates each of the networks.
- OpenStack further interprets the template and instantiates each of the VNFs, connecting them accordingly to the subnets.
- OpenStack creates a router between the public network and the management network.
- Yardstick/NSB interprets the test plan using the control interfaces to the VNFs to start their execution and cause traffic to be sourced and sunk according to the test plan.

---

2 As of this writing, the public network in Yardstick*/NSB must be named "yardstick-public." This will be reiterated in the detailed installation instructions.

3 Note: In the current instantiation of Yardstick*/NSB, the translation from the network service model to the OpenStack* template does not happen automatically; this functionality is a work in progress. Each VNF that is “onboarded” into Yardstick/NSB requires a manual step to create the OpenStack template.

4 It is common to refer to the template as the "Heat template" because the OpenStack Heat* service implements the OpenStack* orchestrator.
3.5 Installation Procedures

3.5.1 Server Capacity

As explained in Section 3.3, the minimum required servers are the following: an OpenStack controller, an OpenStack jump host, one or two OpenStack compute hosts, a Yardstick jump host, and a Yardstick replay host. The current scope of this QSG covers only NFVI characterization, which does not use a replay host; the replay host will not be referred to further in this current document.

Recent processors such as Intel® Xeon® Processor E5 family or Intel® Xeon® Scalable Processors are required for Yardstick/NSB, due to the use of DPDK software components compiled for these architectures. This is true for the compute host(s), and the Yardstick jump host.
For the controller, we recommend at least 30 GB of main memory. For the compute host, we recommend and at least 60 GB of main memory. For both hosts, we recommend two separate 10 Gbps NICs.

The capacity of a host can be queried via the following commands:

```
cat /proc/cpuinfo
```
```
cat /proc/meminfo
```

Sample output of these commands is shown in Appendix 5.1 and 5.2.

### 3.5.2 Software Tool Installation

Yardstick/NSB depends on many open source tools. It is recommended that you install these Linux packages. In general, if a package is required, but not installed, an error message will appear at some point in the installation procedure that complains about the missing package. The error can be corrected by installing the package.

```
sudo apt-get install -y python-dev python-pip python-mock tcpreplay libpcap-dev python-virtualenv python-dev libssl1-dev git build-essential tree ssh openssh-server
```

### 3.5.3 OpenStack and Docker* Installation

There are many ways to install OpenStack. However, the recommended way for this guide is kolla-ansible, which requires a deployment node on which the Ansible* framework is installed. It can be the same machine as the Yardstick jump host or a separate machine.

The release “docker-ce” is the current (as of this writing) release of the Docker engine. However, we have seen cases in which Kolla installation failed. As a fallback, try to use the release “docker-engine” (also installed by kolla-ansible if Docker is not present on the system), but first remove all Docker components.

```
sudo apt-get remove docker docker-engine docker.io
```

For Ubuntu distribution, it is better to use apt for Ansible installation instead of pip*, because pip does not create some directories needed for pre-deployment preparation. Please refer to the Ansible web page ([https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#latest-releases-via-apt-ubuntu](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#latest-releases-via-apt-ubuntu)) for the installation method on Ubuntu; as a shortcut, you can follow these steps for Ansible installation on Ubuntu:

```
$ sudo apt-get update
$ sudo apt-get -y install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get -y install ansible
```

After the installation of Ansible, follow the kolla-ansible installation procedure available in [https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html](https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html) or use the steps below:

```
$ sudo apt-get -y install python-pip
$ sudo pip install -U pip
$ sudo apt-get install python-dev libffi-dev gcc libssl1-dev python-selinux
```

It is recommended that you set the following values in the /etc/ansible/ansible.cfg file:

```
[defaults]
host_key_checking=False
pipelining=True
forks=100
```
Please note that this guide for OpenStack installation might become outdated after some time, so it is recommended that you refer to the latest installation instructions on the kolla-ansible web page (https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html).

Currently, you can use the following steps for OpenStack deployment. First, install Kolla Ansible using following command:

```
$ pip install kolla-ansible
```

Copy example configuration files, and then modify them to reflect your OpenStack infrastructure.

```
$ cp -r /usr/local/share/kolla-ansible/etc_examples/kolla /etc/kolla/
$ cp /usr/local/share/kolla-ansible/ansible/inventory/* .
$ vim multinode
$ vim /etc/kolla/globals.yml
```

Make sure that all nodes are available via IP addresses you set in a multimode inventory file:

```
$ ping controller
$ ping compute
```

Then copy SSH keys to enable password-less authentication:

```
$ ssh-copy-id controller
$ ssh-copy-id compute
```

Make sure that Python* is installed on all nodes:

```
$ ansible -m raw -a "apt-get -y install python-dev" -i multinode all
```

Check if all OpenStack nodes are accessible for Ansible:

```
$ ansible -m ping -i multinode all
```

Generate passwords for the OpenStack services:

```
$ kolla-genpwd
```

After password generation, you can modify the password file and set your own passwords (in particular for keystone_admin) to simplify access:

```
$ vim /etc/kolla/passwords.yml
```

When all configuration jobs are done, you can start installation of OpenStack services with the following commands:

```
$ kolla-ansible -i multinode bootstrap-servers
$ kolla-ansible -i multinode prechecks
$ kolla-ansible -i multinode deploy
```

Please note that correct configuration reflecting your real infrastructure is crucial for deploying OpenStack services successfully. In case of errors, please refer to kolla-ansible and OpenStack documentation, in addition to Ansible log files.

In some cases (depending on kolla-ansible development status), it might be necessary to re-run the deployment process several times because it cannot get information about correctly working some of service Docker containers.

```
$ kolla-ansible -i multinode deploy
```
After the successful installation of OpenStack, you can verify it by inspecting Docker images. Then start post-deployment tasks:

```
# kolla-ansible post-deploy
```

Next, set required OpenStack variables and install a command-line interface (CLI) to manage OpenStack services:

```
. /etc/kolla/admin-openrc.sh
pip install python-openstackclient python-glanceclient python-neutronclient
```

The last step is to create an external network and router on OpenStack enabling external access to the instances. Before starting the initialization script, it is recommended that you check external network parameters and adjust them if needed.

```
vim /usr/local/share/kolla-ansible/init-runonce
. /usr/local/share/kolla-ansible/init-runonce
```

### 3.5.3.1 Inspecting OpenStack and Docker Images

Documentation of Docker commands is beyond the scope of this QSG. We recommend reading the Docker site’s information for more information. As a shortcut, all Docker commands are prefaced with the keyword Docker, and help pages can be displayed via the “docker -h” or “docker <subcommand> -h” commands. In the procedures that follow, a few Docker commands (for example, “docker images”) will be shown, wherein the purpose should be straightforward.

At the end of this step, a large number of Docker images will be present. Use the following command to show the Docker images:

```
root@sg-nsb-tester:~# docker images
```

Appendix 5.3 shows such a list for the controller node.

### 3.5.3.2 Setting OpenStack Environmental Variables

In many of the steps that follow, you will invoke many OpenStack APIs, either directly via shell commands or indirectly via Bash* or Python scripts. OpenStack APIs require a set of credentials to be provided for authentication. The most convenient way to provide these credentials is to define environmental variables. We recommend that you create a script that defines them and run “source <script>” to define them upon logon.

These variables must be defined in any session in which Yardstick/NSB is operated—both in the bare-metal control server and in the Yardstick Docker container (which will be created and used in subsequent steps).

The following list provides environment variables; note that they might differ in your system:

- The value of OS_PASSWORD is the password that must be entered when logging on to the OpenStack dashboard. By default, the OpenStack installation procedures generate a long hash string, which is strong but inconvenient. Because Yardstick/NSB is used for performance testing, we recommend that a friendlier password be defined, as shown below.

```
# Yardstick/NSB is used for performance testing, so we recommend using a friendlier password
OS_PASSWORD=my_password
```

- The OS_AUTH_URL environmental variable refers to the IP address of the control host.

---

5 A Docker* image is analogous to an OpenStack* VM—a set of file objects that can be instantiated into a Docker container. In turn, a Docker container is analogous to an OpenStack VM instantiation.
Yardstick*/NSB Quick-Start Guide (QSG)

- The EXTERNAL_NETWORK environmental variable refers to the name of the public network (see Section 3.3 for more information). As of this writing, the value of the variable must be yardstick-public, as shown below.

```bash
root@sg-nsb-tester:~# cat /etc/kolla/admin-openrc.sh
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
#export OS_USERNAME=admin
export OS_PASSWORD=yYjAKcBe5nv1qObgpEXra6xMTH6HIBevhIYwwv3B
export OS_PASSWORD=admin
export OS_AUTH_URL=http://10.23.95.83:35357/v3
export OS_INTERFACE=internal
export OS_IDENTITY_API_VERSION=3
export EXTERNAL_NETWORK=yardstick-public
root@sg-nsb-tester:~#
```

3.5.3.3 Verify Access to the OpenStack Dashboard

OpenStack can be operated entirely from a CLI. Documenting this CLI is outside the scope of this QSG. Instead, we recommend that the user operate OpenStack from the dashboard (implemented by the OpenStack Horizon* service).

After a full installation of OpenStack (for example, following [https://releases.openstack.org/ocata/index.html](https://releases.openstack.org/ocata/index.html), the OpenStack dashboard is available via a web interface, at the URL [http://10.23.95.83](http://10.23.95.83) (controller host).

![OpenStack dashboard](image)

**Figure 22. OpenStack* logon dashboard**

If you have edited your `/etc/kolla/admin-openrc.sh` configuration file as described above, to change the password to “admin,” this page allows you to log on with the credentials admin and admin (for the username and password fields).
Once in the dashboard, as in Figure 23, the left margin of the page provides links with which to navigate to pages where the cluster can be configured and where networks and VMs can be instantiated and executed.

![Figure 23. OpenStack* dashboard Overview page](image)

### 3.5.3.4 Define Security Groups

OpenStack security groups are a mechanism for regulating the types of traffic allowed to pass through networks. A full treatment of security groups is beyond the scope of this QSG. For the purposes of NFVI test VNF execution, the goal is to define “pass-through” rules for the packet types required by the test.

In order to create and edit security groups, navigate to Project > Network > Security Groups, as shown in Figure 24. In your installation, there might be only a “default” security group. Click Manage Rules.
Figure 24. OpenStack* Security Groups page

Figure 25 shows the rules that are recommended. Several traffic types must be permitted for proper operation of the test VNF, such as ICMP (that is, “ping”), TCP, UDP, and traffic on port 22 (for SSH) and 8474 (because Yardstick and the test VNF use it). These rules are defined for both ingress and egress traffic.
3.5.3.5 Create VM Flavors

A “flavor” is a specification of virtual resources used in the creation of a VM instance. It is convenient to have standard flavors defined in an OpenStack cluster for use in launching VM instances manually. A typical set is shown in Figure 26.

Flavors are not necessary for Yardstick/NSB testing, because the appropriate VM specification is created dynamically when the VNF is launched.
3.5.3.6 Create the Provider Network

When OpenStack is installed, a “ghost” external network, 10.2.85.0/24, is “defined” associated with the external bridge, br-ex (which serves as the gateway, 10.2.85.1, of the network), and it is given a physical network name, “physnet1.” This is not a true external network, because there is no real external network 10.2.85.0/24; as long as these definitions are in effect, OpenStack will behave as though it is an external network, and it will assign floating IP addresses from it for newly created VMs; however, only the controller, compute(s), and VMs will be able to reach it.

For Yardstick/NSB, a provider network, capable of serving IP addresses to VMs, must be created. In the dashboard, navigate to Admin > Networks, and then click Create Network. Fill in the fields as shown in Figure 27. In particular, the network should be named exactly yardstick-public. As of this writing, this network name is a hard-coded value.

---

6 Note: You should navigate to this page from Admin, not from Project. The Project path appears to provide a page with which you can create a network, but it will be lacking in some of the options required for the provider network.
When the command completes, the Networks page will indicate that a new network has been added.

External network “public-network” is created manually. It is on physical network “physnet1,” which goes to br-ex. It is 10.2.85.0/24, and the gateway is 10.2.85.1, with 100 floating IPs.

The correct values are:

```
Name: yardstick-public
ID: <long hash string>
Project ID: <long hash string>
Status: Active
Admin State: UP
Shared: No
External Network: Yes
MTU: 1500
Provider Network: Network Type: flat
    Physical Network: physnet1
    Segmentation ID: -
    Project: admin
```

Within this network, we establish one subnet, in which we also specify the pool of floating IP addresses to be handed out to VMs. Click Subnet > Create Subnet, and then fill out the fields as shown in Figure 28 and Figure 29.
The values for the subnet should be:

- Subnet name: public-subnet
- ID: <long hash string>
- Network Name: yardstick-public
- Network ID: <long hash string>
3.5.4 Yardstick Image/Container Installation for OpenStack

Yardstick/NSB test runs are executed by logging on to the Yardstick container and running commands in the container. In this step, the Yardstick container is downloaded and installed.

3.5.4.1 Obtaining a Yardstick Code Drop from a Git* Repository

Downloading is done from the Yardstick jump host, pulling the image from a repository at OPNFV. We recommend creating a directory into which to place this image. Issue the following command on the Yardstick jump host:

```
mkdir /home/nsb/mydirectory
cd /home/nsb/mydirectory
git clone https://gerrit.opnfv.org/gerrit/yardstick
```

3.5.4.2 Build a Yardstick Image from a Code Drop

Yardstick/NSB ships with an installation script, ../yardstick/nsb_setup.sh. When this script executes, it scans the admin-openrc.sh file, created by the OpenStack installation, and the /boot/grub/grub.cfg file, in order to determine how to configure the container with respect to the OpenStack cluster to which it is attached and how to configure VNFs that it builds (for example, determining whether Huge Pages are employed). The script downloads any Linux packages required for the installation, builds a Docker image of Yardstick, and then runs the Yardstick container.

In this step, we assume that the admin-openrc.sh file has been created in the /root directory. The command is:

```
./nsb_setup /root/admin-openrc.sh
```

If your cluster is inside a firewall, you must define HTTP proxies as with the following commands:

```
export http_proxy=http://myproxyserver.com:myproxyport
export https_proxy=https://mysshproxyserver.com:mysshproxyport
```

The command will run for more than a dozen minutes, and it will generate voluminous output, including a number of apparent “error” and “warning” messages. If the command succeeds, it will create an image such as:

```
root@ysjump:/home/ysjump# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
opnfv/yardstick     latest              90e0a2686f54        25 minutes ago      1.85 GB
root@ysjump:/home/ysjump#
```

It will also create the container:

```
root@ysjump:/home/ysjump# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
```

---

Subnet Pool: None
IP Version: IPv4
CIDR: 10.2.85.0/24
IP Allocation Pools: Start 10.2.85.100 – End 10.2.85.200
Gateway IP: 10.2.85.1
DHCP Enabled: No
Additional Routes: None
DNS Name Servers: 8.8.8.8
Network Address: 10.2.85.1/24
3.5.4.3 The Test VNF VMs

As part of its behavior, the nsb_setup.sh script previously executed builds a VM image for test VNFs and loads it into the OpenStack VM image repository.

For NFVI characterization, a software package called PROX, implements both the traffic generator and test VNF. PROX runs as a Linux application, and it takes on its traffic generator or test VNF “personality” from command-line arguments and configuration files.

An explanation of how to configure PROX for these usages is outside of the scope of this QSG. A user who only wants to execute the pre-implemented test cases need only download the VM image file(s) and install them in the OpenStack storage system.

The VNF VM image can be inspected by navigating to Admin > System > Images in the dashboard:

![Figure 30. Inspecting the sample VNF VM image](image)

The properties of the VM image can be inspected by clicking the name hotlink. The image has a number of custom properties, defined in the image metadata. This can be further inspected by clicking the arrow next to the Launch field in the upper right, and by then clicking Update Metadata, as shown in Figure 31.
Image metadata is used by OpenStack to carry out special-purpose steps when launching a VM from the image. The metadata defined for the image is shown in the “Existing Metadata” column (as shown in Figure 32).

The metadata defined for the test VNF is configured automatically as part of the nsb_setup.sh script. It consists of the following values:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>nsb</td>
</tr>
<tr>
<td>Visibility</td>
<td>Public</td>
</tr>
<tr>
<td>Protected</td>
<td>No</td>
</tr>
<tr>
<td>Checksum</td>
<td>1c5a47f36a40b418c3b15810300f5e5de6100f</td>
</tr>
</tbody>
</table>
**4 NFVI Characterization**

Characterization of NFVI is done by collecting the KPIs on the infrastructure with the set of selected VNFs, as depicted in Figure 33. This process can be done using commercial tools or open source tools like VSPERF* or NSB.

With the NSB framework, characterization can be performed on each environment (bare metal, standalone virtualized, and managed virtualized environments) mentioned in this document; however, the number of available VNFs for characterization can be limited or different for specific environments.

---

<table>
<thead>
<tr>
<th>owner_specified.shade.md5</th>
<th>&lt;message digest value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner_specified.shade.object</td>
<td>images/yardstick-samplevnfs (i.e., the name of the VM)</td>
</tr>
<tr>
<td>owner_specified.shade.sha256</td>
<td>&lt;secure hash value, for security&gt;</td>
</tr>
</tbody>
</table>

“Shade” refers to the Shade* library, used to simplify the task of manipulating VMs in Yardstick/NSB. (For more information, see Shade).
4.1 Prerequisites and Requirements

Components needed to perform NFVI characterization:

- Traffic generator—depending on the test case it can be either external or internal, acting as a part of the NFVI
- Set of VNFs that will be used for characterization; a list of available VNFs is described in section 4.5.1
- NFVI as SUT
- Set of KPIs to collect
- Storage for results and KPI data collection; for example, InfluxDB
- (Optional) Data visualization tool; for example, Grafana
- (Optional) Packet replay system
Configuration and system requirements:

- Validation of the hardware for failures and bottleneck issues
- Validation of the infrastructure configuration for failures and bottleneck issues
- Type of traffic selection corresponding to a set of VNFs; workloads should be simple to explain and should highlight specific aspects of NFV
- Manual configuration of pod.yaml in the /etc/yardstick/nodes directory
- Manual configuration of prox-baremetal-*.yaml in the test case (TC) directory
- Manual adjustment of test-case parameters in the TC directory
- SSH access from the NSB jump host to the TG and the SUT or compute node

All tests performed for NFVI characterization should also meet following requirements:

- Be realistic—the set of VNFs chosen for characterization should reflect real traffic and a real environment
- Repeatable by others—using commercial or open source components
- Reproducible—run tests multiple times to ensure that the same tests produce the same results; also, different traffic generators should give similar performance numbers
- External factors considered—because external factors can have some influence to the results (for example, noise from the CPU scheduler, noise from neighbors, temperature, time since boot and VNF startup, and so on)

4.2 KPI Collection

KPI collection is the process of sampling KPIs at multiple intervals to allow for investigation into anomalies during runtime. Some KPI intervals are adjustable. KPIs are collected from traffic generators and NFVI for the SUT. There is already some reporting in NSB available, but NSB collects all KPIs for analytics to process.

Below is an example list of basic KPIs:

- Throughput
- Latency
• Packet delay variation
• Maximum establishment rate
• Maximum tear-down rate
• Maximum simultaneous number of sessions

Of course, there can be many other KPIs that will be relevant for specific a NFVI, but in most cases these KPIs are enough to give you a basic picture of the SUT.

NSB also uses collectd plug-ins in order to collect the KPIs. Currently the following collectd plug-ins are available:

• Libvirt
• DPDK stats
• DPDK events
• OVS events
• vSwitch stats
• Huge Pages
• RAM
• Intel® PMU plug-in

4.3 VNF Selection

VNF selection should be done with respect to expected traffic and the final working environment. There are several VNFs available in NSB defined in separate TC files. You can use all of them or create your own TC files, in addition to defining your own VNFs to reflect your individual needs. In any case, the VNFs chosen for characterization should be realistic as much as possible so that you get valuable results showing how your NFVI will behave in the real world.

VNFs currently available in the NSB test suite include the following:

• Packet forwarding (no touch), “no-op” VNF, baseline—might not be applicable for a switched network
• L3 forwarding—packet forwarding modifying MACs
• Multi-flow L3 forwarding—200K flows, softswitch stress test
• MPLS tagging—protocol conversion, adding/removing MPLS tags, and packet length variation
• Access Control List (ACL)—flow matching ACL, complex packet filtering
• Load balancing (LB)/5-tuple lookup—5-tuple-based flow matching table lookups for load balancing
• Buffering—packet flow buffering for at least 125 ms, stresses cache and memory
• Border network gateway (BNG)—ARP, QinQ, LB, routing, generic route encapsulation (GRE), and MPLS
• BNG with quality of service (QoS)
• Virtualized Provider Edge (vPE)—ACL filtering, flow classification, routing (longest prefix match [LPM] lookup), metering, policing, and marking
• Lightweight Address Family Translation Router (lw-AFTR)—IPv4 and IPv6 conversion
4.4 Configuration Files

Each test case consists of a couple of configuration files defining many aspects of the test; the most crucial definitions are:

- Topology
- Nodes
- Traffic for the TG
- VNF

The following shows a sample configuration file (tc_file):

```yaml
---
schema: "yardstick:task:0.1"

scenarios:
  -
    type: NSPerf
    traffic_profile: ../../traffic_profiles/prox_binsearch.yaml
    topology: prox-tg-topology-2.yaml

nodes:
  tg_0: tg_0.yardstick
  vnf__0: vnf_0.yardstick

options:
  vnf__0:
    prox_path: /opt/nsb_bin/prox
```

Figure 35. Sample global profile of NFV performance with aggregated KPIs
The highlighted lines represent key components of each test; they can be adjusted in the files referred to in the configuration shown above. However, in most cases, it is not necessary to change anything to get the test running except the node-definition file, in which you need to set IP addresses for each node, in addition to providing access credentials. In this particular case, this is the prox-baremetal-2.yaml file:
interfaces:
  xe0:
    vpci: "0000:00:05.0"
    local_mac: "52:54:00:cb:82:25"
    driver: "virtio-pci"
    local_ip: "152.16.100.21"
    netmask: "255.255.255.0"
    dpdk_port_num: 0
  xe1:
    vpci: "0000:00:06.0"
    local_mac: "52:54:00:0c:7d:58"
    driver: "virtio-pci"
    local_ip: "152.16.40.21"
    netmask: "255.255.255.0"
    dpdk_port_num: 1
routing_table:
  - network: "152.16.100.20"
    netmask: "255.255.255.0"
    gateway: "152.16.100.20"
    if: "xe0"
  - network: "152.16.40.20"
    netmask: "255.255.255.0"
    gateway: "152.16.40.20"
    if: "xe1"
nd_route_tbl:
  - network: "0064:ff9b:0:0:0:0:9810:6414"
    netmask: "112"
    gateway: "0064:ff9b:0:0:0:0:9810:6414"
    if: "xe0"
  - network: "0064:ff9b:0:0:0:0:9810:2814"
    netmask: "112"
    gateway: "0064:ff9b:0:0:0:0:9810:2814"
    if: "xe1"

Also, depending on hardware configuration, you might need to change additional parameters (highlighted in the code above), such as the following, to reflect your configuration:

- vpci
- local_mac
- driver

After such modifications, the configuration files should be ready to run the test case.

4.5 Execute NFVI Test Cases

4.5.1 List of NFVI Test Cases

Each NFVI test case has a top-level context file defined in the Yardstick container, in the /home/opnfv/repos/yardstick/samples/vnf_samples/nsut/prox directory. In the following listing, only the context files and directories corresponding to the managed virtualization test cases are shown.

```
root@ef1e3c99799e:/home/opnfv/repos/yardstick/samples/vnf_samples/nsut/prox# pwd
/home/opnfv/repos/yardstick/samples/vnf_samples/nsut/prox
root@ef1e3c99799e:/home/opnfv/repos/yardstick/samples/vnf_samples/nsut/prox# ls -l
 total 152
//(only the templates relevant to the QSG are shown)
-rw-r--r-- 1 root root 1693 Aug 25 20:53 prox-tg-topology-1.yaml
```
You can also view a list of NSB PROX test cases with the following command:

```
root@f04c7932172f:/home/opnfv/repos/yardstick/yardstick/cmd# ./NSBperf.py -list
...
BareMetal Testcase:
===================
tc_prox_baremetal_l2fwd-2.yaml
tc_prox_baremetal_l2fwd-4.yaml
tc_prox_baremetal_mpls_tagging-2.yaml
tc_prox_baremetal_mpls_tagging-4.yaml
tc_prox_baremetal_lw_aftr-4.yaml
tc_prox_baremetal_l2fwd-2.yaml
tc_prox_baremetal_bng_qos-4.yaml
tc_prox_baremetal_bng_qos-2.yaml
tc_prox_baremetal_l3fwd-4.yaml
tc_prox_baremetal_acl-2.yaml
tc_prox_baremetal_l2fwd_pktTouch-4.yaml
tc_prox_baremetal_l2fwd_pktTouch-2.yaml
tc_prox_baremetal_l2fwd_multiflow-2.yaml
tc_prox_baremetal_vpe-4.yaml
tc_prox_baremetal_mpls_tagging-2.yaml
tc_prox_baremetal_l2fwd_multiflow-4.yaml
tc_prox_baremetal_l2fwd-2.yaml
tc_prox_baremetal_ramp-2.yaml
tc_prox_baremetal_bng-4.yaml
tc_prox_baremetal_acl-4.yaml
tc_prox_baremetal_buffering-1.yaml

Openstack Testcase:
===================
tc_prox_heat_context_mpls_tagging-4.yaml
tc_prox_heat_context_acl-2.yaml
tc_prox_heat_context_l2fwd_multiflow-4.yaml
tc_prox_heat_context_lw_aftr-4.yaml
tc_prox_heat_context_acl-4.yaml
tc_prox_heat_context_l2fwd-4.yaml
tc_prox_heat_context_l2fwd-13fwd-4.yaml
tc_prox_heat_context_l2fwd_multiflow-2.yaml
tc_prox_heat_context_mpls_tagging-2.yaml
tc_prox_heat_context_l2fwd-2.yaml
tc_prox_heat_context_l2fwd_pktTouch-2.yaml
tc_prox_heat_context_bng_qos-4.yaml
tc_prox_heat_context_l2fwd_pktTouch-4.yaml
```
4.5.2 Execute an NFVI Test Case

A test case is executed from inside the Yardstick container with the following command:

```bash
cd /home/opnfv/repos
yardstick -d task start yardstick/samples/vnf_samples/nsut/prox/<tc_file>
```

For this command, `<tc_file>` is one of the options listed in Section 2.4.3.1.

When this command is executed:

- Debug output will be generated in standard output in the shell session in which the command was run. This output can be used for debugging purposes. A more detailed description of the output is beyond the scope of this QSG.
- Test output will be generated in the /tmp/yardstick/yardstick.log file (in the Yardstick container). This output is consistent with the standard output data that the PROX test framework (upon which Yardstick/NSB NFVI testing is based) generates. It can be used to supplement the “official” data stream that Yardstick/NSB generates.
- KPIs will be streamed to the InfluxDB database. This data can be retrieved via manual database queries of the database, or it can be visualized by use of the Grafana tool.

4.6 Miscellaneous Procedures

4.6.1.1 Manual Inspection of InfluxDB Data

This information is meant to be used only for debugging if you suspect that data is not being stored in the database.

The InfluxDB database runs in a Docker container. This container was started via the procedure in Section 2.3.2. You can find the name of the running container via “docker ps,” as in the following example:

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
... 1d09a38041d2 tutum/influxdb:0.13 ""/run.sh" 7 s ago Up 7 days 0.0.0.0:8083->8083/tcp, 0.0.0.0:8086->8086/tcp zen_blackwell ...
```

In this case, the running InfluxDB container happens to have the name “zen_blackwell.”

You can log on to the container via the following command:

```
docker exec -it zen_blackwell /bin/bash
```

Once in the container, you can log on to the database via the following command:

```
$ influxdb
> use yardstick
Using database yardstick
```

---

Yardstick*/NSB Quick-Start Guide (QSG)
For this command, “yardstick” is the name of the database created by Yardstick/NSB. For more information, read InfluxDB.

4.6.1.2 Configuration of Package Management with the Advanced Package Tool (APT)

Correct installation of Yardstick depends on the ability to download Linux and Python packages from the Internet. This, in turn, depends on the proper configuration of the Yardstick jump host package-management system.

In the current version of this QSG, the host operating system is Ubuntu, so the package-management system of interest is the Debian package manager, dpkg, in addition to the front-end Advanced Package Tool (APT). At a high level, this means you must:

- Make sure that the appropriate libraries are configured
- Make sure that certificates are installed and that the Ubuntu cloud keyring is installed
- Make sure that the /boot partition does not run out of space
- Make sure that CPU architectures irrelevant to your configuration will not be requested when running package updates

This section is not a complete APT reference manual, or even a full tutorial. It is a suggested procedure to run in case you encounter difficulties in running the nsb_setup.sh script. Problems with package management can manifest themselves in a number of different fatal errors in that script, including certificate errors, repositories-not-available errors, fetch errors, and so on. Some of the errors caused by a package manager misconfiguration can be obscure.

If you are starting with a fresh install of the host operating system, you might not encounter these problems. If you do, consider the following procedure.

First, make sure that APT configuration files are in a safe state. Check the /etc/apt/apt.conf file for the presence of proxy definitions. An example of this file is:

```
Acquire::http::Proxy "http://proxy.domain.com:911";
Acquire::https::Proxy "https://proxy.domain.com:911";
```

Second, make sure your installation will obtain the correct Docker packages. In the default version of the /etc/apt/sources.list file, it is possible that the following lines will appear:

```
# deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial edge
# deb-src [arch=amd64] https://download.docker.com/linux/ubuntu xenial edge
```

These lines should be commented out, as shown above, to make sure that the packages are obtained from Ubuntu, not Docker.

If your installation requires proxies, make sure that the apt.conf file defines them. Make sure that the values are correct; incorrect proxy values will lead to “Proxy CONNECT aborted” errors when running APT commands.

Third, run the following commands to ensure that all of the required repositories and certificates are installed:

Assuming you are running on x8086 you don’t need ARM packages, this removes an error message:

```
user@nsb:~ dpkg --force-architecture -remove-architecture arm64
```
Remove unneeded packages – this will also help to clear out the /boot partition

```
user@nsb:# apt-get autoremove --purge
```

You need to store certificates

```
user@nsb:# apt-get install ubuntu-cloud-keyring
```

Make sure you have a key for docker

```
user@nsb: # apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 8118E89F3A912897C070ADBF76221572C52609D
```

nsb_setup.sh uses ansible

```
user@nsb: # apt-add-repository ppa:ansible/ansible
```

Update/upgrade ensures that packages are up-to-date

```
user@nsb: # apt-get update -y
user@nsb: # apt-get upgrade
```

You should reboot if you think that packages that you installed or upgraded may have changed the kernel

```
user@nsb: # reboot
```
4.7 PROX Application

PROX is a DPDK application that is included in the NSB framework. It is used by NSB to create VNFs using already defined configuration files, but you can also use it to create your own configuration files for PROX and use it as standalone application. PROX as a standalone application can be used to diagnose problems rather than characterizing NFV.

By default, the PROX application is in the /opt/nsb_bin directory of the Yardstick container. Sample configuration files are in the /home/opnfv/repos/yardstick/samples/vnf_samples/nsut/prox/configs directory.

In the configuration files, you can define jobs that should be completed by the PROX application. Depending on the content of the configuration files, the PROX application can act as a traffic generator or VNF.

To run PROX as a standalone application, some requirements must be met, similar to running it as a DPDK application. At a minimum, you need to have free Huge Pages and two interfaces bound to a DPDK driver. For further details about the requirements, please refer to the current DPDK documentation available at https://dpdk.org/doc.

4.7.1 Execute PROX

If you want to use PROX as both a TG and a SUT, you need to run it on both machines. On the TG, this can be done with following commands:

```
cd /opt/nsb_bin
./prox -f configs/gen_l2fwd-2.cfg
```

On the SUT you can complete this in a similar way, but you need to use a config file matching the one used on the TG:

```
cd /opt/nsb_bin
./prox -f configs/handle_l2fwd-2.cfg
```

After successful initialization, you should be to see a screen like the one shown in Figure 36.
Figure 36. Sample screenshot of the PROX application
5 Appendixes

5.1 /proc/cpuinfo Example

```
root@ubuntu:~# cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 63
model name : Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz
stepping : 2
microcode : 0x1e
cpu MHz : 1200.000000

cache size : 15360 KB
physical id : 0
siblings : 12
core id : 0
cpu cores : 6
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 15
wp : yes
flags : fpu vme de pse tsc msr pae mce cmov pat pse36 clflush
dtsc acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc
arch_perfmon pebs bts rep_good nopl apic ida mcm vsm tsc_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust
cpu name : Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz
```
5.3 Controller Docker Images

You can find the list of docker images using the following command:

root@sg-nsb-sut:~# docker images

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATED  SIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-nova-compute 4.0.2 1.112 GB</td>
<td>20 hours ago</td>
<td>b4dbe0456c9a</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-neutron-openvswitch-agent 4.0.2 893.4 MB</td>
<td>20 hours ago</td>
<td>1f34e3ee8dbe</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-nova-ssh 4.0.2 997.1 MB</td>
<td>20 hours ago</td>
<td>46694e271bb9</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-kolla-toolbox 4.0.2 763.5 MB</td>
<td>20 hours ago</td>
<td>92119d48c91e</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-nova-libvirt 4.0.2 809.2 MB</td>
<td>20 hours ago</td>
<td>142cb60eb01d</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-fluentd 4.0.2 644.2 MB</td>
<td>20 hours ago</td>
<td>f421ceedc974</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-openvswitch-vswitchd 4.0.2 338.2 MB</td>
<td>20 hours ago</td>
<td>671d7d4cc723</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-openvswitch-db-server 4.0.2 338.2 MB</td>
<td>20 hours ago</td>
<td>633f3ae4e536</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-cron 4.0.2 309 MB</td>
<td>20 hours ago</td>
<td>5371ac3a37d7</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-nova-compute 4.0.2 1.112 GB</td>
<td>42 hours ago</td>
<td>7436007f338e</td>
</tr>
<tr>
<td>10.23.95.83:4000/kolla/ubuntu-source-neutron-openvswitch-agent 4.0.2 893.4 MB</td>
<td>42 hours ago</td>
<td>e97086f33d1e</td>
</tr>
</tbody>
</table>
5.4 Yardstick Command-Line Output Example

```
2017-12-12 17:40:23,063 [INFO] yardstick.cmd.commands.task task.py:54 Task START
2017-12-12 17:40:23,065 [INFO] yardstick.benchmark.core.task task.py:483 Parsing task config:
tc_prox_baremetal_12fwd-4.yaml
2017-12-12 17:40:23,073 [INFO] yardstick.benchmark.contexts.node node.py:57 Parsing pod file:
prox-baremetal-4.yaml
2017-12-12 17:40:23,089 [INFO] yardstick.benchmark.core.task task.py:373 Starting runner of type
'Duration'
2017-12-12 17:40:23,096 [INFO] yardstick.benchmark.runners.duration duration.py:46 Worker START, duration is 600s
2017-12-12 17:40:23,298 [INFO] yardstick.benchmark.scenarios.networking.vnf_generic
vnf_generic.py:599 Instantiating tg__0
2017-12-12 17:40:27,388 [INFO] yardstick.benchmark.scenarios.networking.vnf_generic
vnf_generic.py:601 Waiting for tg__0 to instantiate
2017-12-12 17:40:27,684 [INFO] yardstick.network_services.vnf_generic.vnf.prox_helpers
prox_helpers.py:842 Provision and start the prox
2017-12-12 17:40:28,392 [INFO] yardstick.network_services.vnf_generic.vnf.sample_vnf
sample_vnf.py:769 Waiting for PROX VNF to start..
2017-12-12 17:40:38,404 [INFO] yardstick.network_services.vnf_generic.vnf.sample_vnf
sample_vnf.py:759 PROX VNF is up and running.
2017-12-12 17:40:58,348 [INFO] yardstick.benchmark.scenarios.networking.vnf_generic
vnf_generic.py:599 Instantiating vnf__0
2017-12-12 17:41:02,344 [INFO] yardstick.benchmark.scenarios.networking.vnf_generic
vnf_generic.py:601 Waiting for vnf__0 to instantiate
2017-12-12 17:41:03,346 [INFO] yardstick.network_services.vnf_generic.vnf.prox_helpers
prox_helpers.py:842 Provision and start the prox
2017-12-12 17:41:03,346 [INFO] yardstick.network_services.vnf_generic.vnf.sample_vnf
sample_vnf.py:769 Waiting for PROX VNF to start..
2017-12-12 17:41:13,359 [INFO] yardstick.network_services.vnf_generic.vnf.sample_vnf
sample_vnf.py:759 PROX VNF is up and running.
2017-12-12 17:41:31,023 [INFO] yardstick.benchmark.scenarios.networking.vnf_generic
vnf_generic.py:620 Starting traffic on tg__0
2017-12-12 17:41:31,032 [INFO] yardstick.network_services.vnf_generic.vnf.sample_vnf
sample_vnf.py:916 Starting ProxTG client...
2017-12-12 17:41:31,034 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:69 Testing with packet size 64
2017-12-12 17:41:31,035 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 100.0
2017-12-12 17:42:00,421 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 50.0
2017-12-12 17:42:29,476 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 75.0
2017-12-12 17:42:58,532 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 62.5
2017-12-12 17:43:27,584 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 56.25
2017-12-12 17:43:56,640 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 59.375
2017-12-12 17:44:25,684 [INFO] yardstick.network_services.traffic_profile.prox_binsearch
prox_binsearch.py:52 Testing with value 57.8125
```
5.5 Yardstick Installation Commands on Clean Ubuntu 16.04*

Add repositories to the sources.list file:

```bash
user@NSB:~$ sudo vi /etc/apt/sources.list

deb http://archive.ubuntu.com/ubuntu xenial main restricted universe multiverse
```

---

5.5 Yardstick Installation Commands on Clean Ubuntu 16.04*

Add repositories to the sources.list file:
Install additional packages:

```
user@NSB:~$ sudo apt-get install \
openssh-server 
xfce4 
xfce4-goodies 
tightvncserver 
tig 
apt-transport-https ca-certificates
```

Permit SSH root logon by editing the `sshd_config` file:

```
user@NSB:~$ sudo vi /etc/ssh/sshd_config
PermitRootLogin yes
user@NSB:~$ sudo service ssh restart
```

Add a user as sudo without a password:

```
user@NSB:~$ sudo visudo
add at the end: username ALL=(ALL) NOPASSWD:ALL
```

Install Yardstick:

```
user@NSB:~$ git clone https://gerrit.opnfv.org/gerrit/yardstick
user@NSB:~$ cd yardstick
user@NSB:~$ git checkout origin/stable/euphrates
user@NSB:~$ sudo ./nsb_setup.sh
```

Check the Yardstick installation:

```
user@NSB:~$ docker images
<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>opnfv/yardstick</td>
<td>latest</td>
<td>05012143ffcb</td>
<td>7 hours ago</td>
<td>1.96 GB</td>
</tr>
</tbody>
</table>
```

```
user@NSB:~$ docker ps
<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th>CREATED</th>
<th>NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>9b014d119462</td>
<td>opnfv/yardstick:latest</td>
<td>&quot;/usr/bin/supervisord&quot;</td>
<td>31 minutes ago</td>
<td>Up 31 minutes</td>
</tr>
<tr>
<td>5000/tcp</td>
<td>yardstick</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Finalize Grafana and InfluxDB installation:

```
user@NSB:~$ sudo docker exec -it yardstick /bin/bash
root@ab12cd34ef56:$ yardstick env influxdb
root@ab12cd34ef56:$ yardstick env grafana
```
6 Troubleshooting and Known Issues

6.1 Version: stable/euphrates

• VMs cannot be created in a standalone virtualization environment.
  Cause:
  The generated XML file used for VM creation contains the wrong data.
  Solution:
  Replace the /home/opnfv/repos/yardstick/yardstick/benchmark/contexts/standalone/model.py file with the same file from master branch.

• Missing /opt/nsb_bin/nsb_setup.sh on the TG.
  Solution:
  Copy this file from the Yardstick container to the TG.

• Hostnames cannot be resolved.
  Solution:
  Update the /etc/hosts files with the correct entries for the TG and VNF image.

• Typo in the config files, lists “.” Instead of “;” in MAC address, which can be difficult to spot.
  Solution: Just find it in the /etc/yardstick/nodes/standalone/pod_trex.yaml file and replace the “.” character with the “;” character.

• DPDK is not starting because no correct igb_uio module can be found.
  Cause:
  The kernel version has been upgraded by unattended Ubuntu upgrades, so the igb_uio module compiled during installation no longer matches the kernel version.
  Solution:
  Disable unattended upgrades, and then downgrade the kernel version to the one you prepared the igb_uio module for. Reboot the TG and VM used as the image template, and then recreate the VM image.

• At the time of writing the standalone virtualized mode is not fully supported in NSB NFVi characterization, causing the following issues:
  o Unable to build an image using yardstick-img-dpdk-modify because the script tries to download an Ubuntu version that does not exist in the Ubuntu FTP. [OVS-DPDK]
  o “PMD: i40evf_dev_configure(): VF can't disable HW CRC Strip” error during the UDP_REPLAY test on an Intel® Ethernet Converged Network Adapter X710 card in a standalone virtualization environment. For an Intel Ethernet Converged Network Adapter X520 and an Intel Ethernet Converged Network Adapter X540, the UDP_REPLAY test works fine.
- No PROX tests for a standalone virtualization environment in the NSBperf script. The Yardstick test run returns an error.
7 References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Docker2017]</td>
<td>Docker. “Docker documentation.” <a href="https://docs.docker.com/">https://docs.docker.com/</a></td>
</tr>
<tr>
<td>[Shade2017]</td>
<td>Python Software Foundation. “shade 0.11.0.” <a href="https://pypi.python.org/pypi/shade/0.11.0">https://pypi.python.org/pypi/shade/0.11.0</a></td>
</tr>
</tbody>
</table>

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to intel.com/benchmarks.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2018 Intel Corporation.  "Printed in USA       0618/BY/PRW/PDF Please Recycle  337755-001US"