APPLICATION NOTE

Intel Corporation
Data Centric Platform Engineering

NFV Demonstration Framework

Intel® Xeon® Scalable Processors

Authors
Sy Jong, Choi
Intel Corporation

Roy, Wang
Intel Corporation

Hu, Eric
Intel Corporation

Chen, Gene
Intel Corporation

Wang, Lifu
Intel Corporation

Nieh, Gorden
Intel Corporation

Liao, Charlie CL
Intel Corporation

Contents

1 Executive Summary

The Network Function Virtualization (NFV) demonstration framework achieves two primary functionalities by combining the console execution of an NFV system with the data visualization from InfluxDB* and Grafana*. This allows for setup to create an NFV demonstration without the use of physical equipment units.

The NFV demonstration setup requires only an Intel® NUC mini Personal Computer (PC) to run the demonstration playback to a client, eliminating the need for additional equipment.

Additionally, this framework can be used as a learning platform that, given its screen recording and publishing capabilities, allows the user to bring up an NFV system to follow a recorded tutorial rather than a document-based user guide.

For more information on NFV which is included in the Intel Container Experience Kits, visit Intel® Network Builders.

1.1 Problem Statement

The ETSI-based NFV reference architecture consists of the following functional blocks:

- OSS and BSS
- EM
- VNF
- Service, VNF, and infrastructure description
- VNF manager(s)
- NFV orchestrator
- VIM(s)
- NFVI that includes hardware and virtual compute, storage, and network resources

Various hardware and software components are required to be integrated to create an NFV system. This creates a high complexity barrier for clients—especially ODMs—who are focused on hardware design only and are not able to demonstrate a boards' full capabilities outside the traditional server functionalities.

The known issues encountered with NFV systems include:

- Setup/integration of an NFV system
- Creation of relevant workload/use cases
- Understanding of the Intel technology
- Inability to extract workload KPIs to show the Intel platform's benefits
- Long turnaround time to optimize the system
- Lack of experience to tune the system's performance for a network transformation

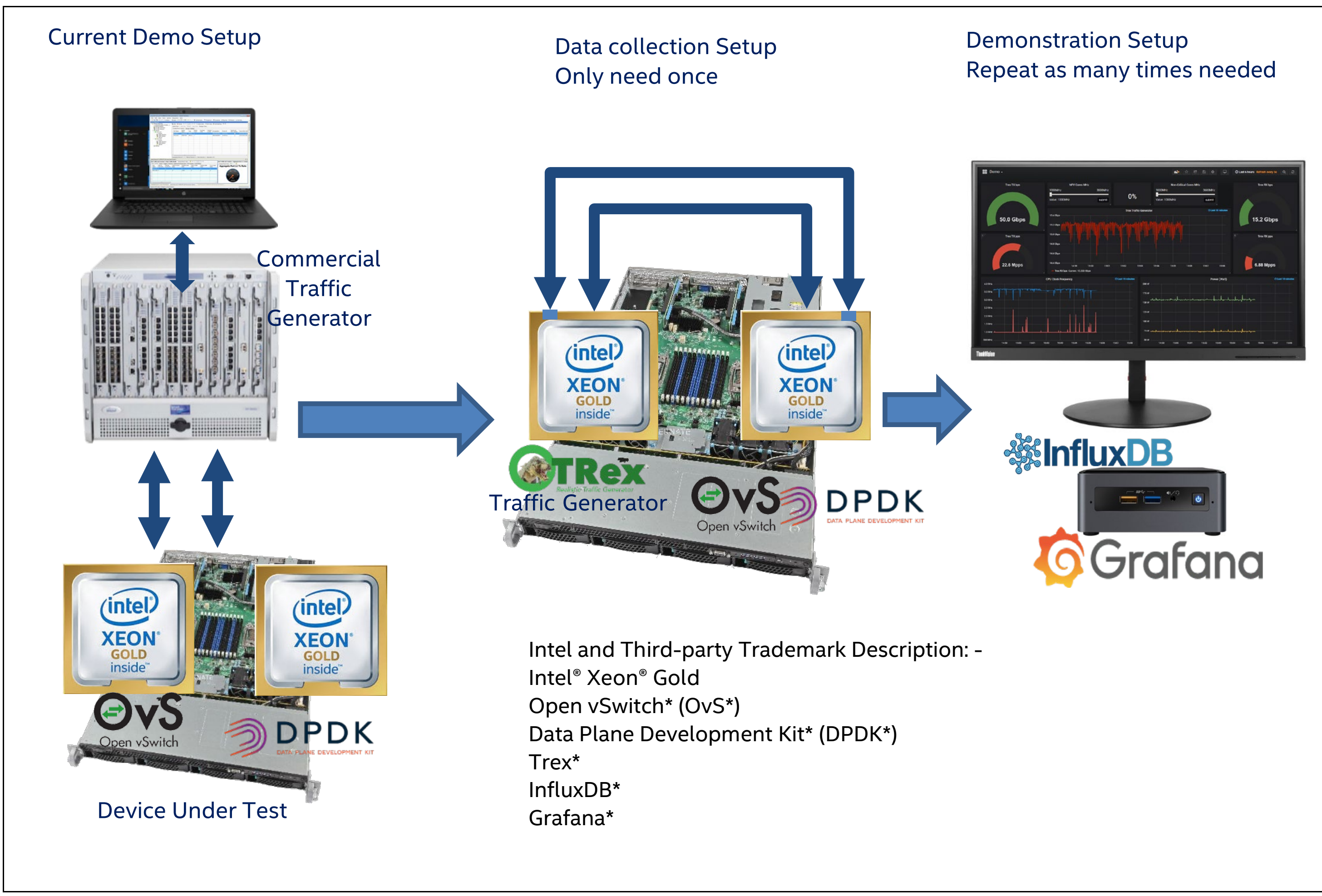
The NFV demonstration framework may provide a solution to solve these problems.

This document outlines how an NFV system is setup and how the KPI data are being collected/recorded with various permutations to Intel technologies, providing a data collection methodology to record the results from an NFV system.

Additionally, the demonstration framework allows the user to save the collected data and play it back later for a demonstration or to present an intuitive data visualization environment for users to discover the performance benefits of Intel processors without additional equipment.

Included in this document:

- A guide to setting up an NFV system with key elements such as:
  - Host system with Linux* OS
  - OvS
  - VM: Ubuntu* 16.04
  - VNF: testpmd application
- The KPI used is the data plane performance of a physical-to-virtual-back-to-physical topology. Highlights from this document include:

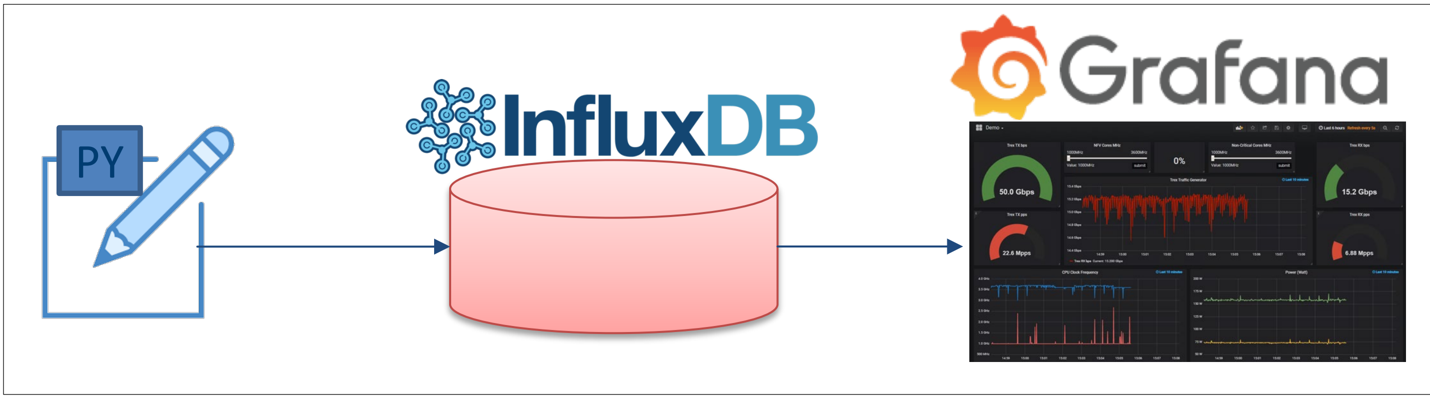


- The demonstration framework provides:
  - A methodology to collect rate from traffic (the data point) to/from the traffic generator to be used for future playback
  - A recording of the console terminal, saving it for future playback to simulate the actual run
  - A playback of the two features at once (the data point collection and the console terminal) and the update to a database for a data visualization tool to present the information to the clients, simulating the actual execution and performance of the NFV system
- The following elements are part of the demonstration framework:
  - asciinema*: a tool to record the console output to play it back later
  - TRex: a tool to generate traffic and to measure the NFV KPI
  - InfluxDB: a tool to store the data point of the NFV KPI
  - Grafana: a tool to display the changes in performance before/after applying the Intel technology

2 Network Function Virtualization (NFV) Demonstration Framework

The NFV demonstration framework is designed to show the playback of a prerecorded terminal console and the data point in a data visualization Graphic User Interface (GUI). The objective is to allow the user to place the NFV Key Performance Indexes (KPIs) into a simple file format to later use Python* to upload the data into an open source time series database which is then picked up by a data visualizing front-end application.

Below is the architecture workflow:



2.1 Summary of the Demonstration Framework

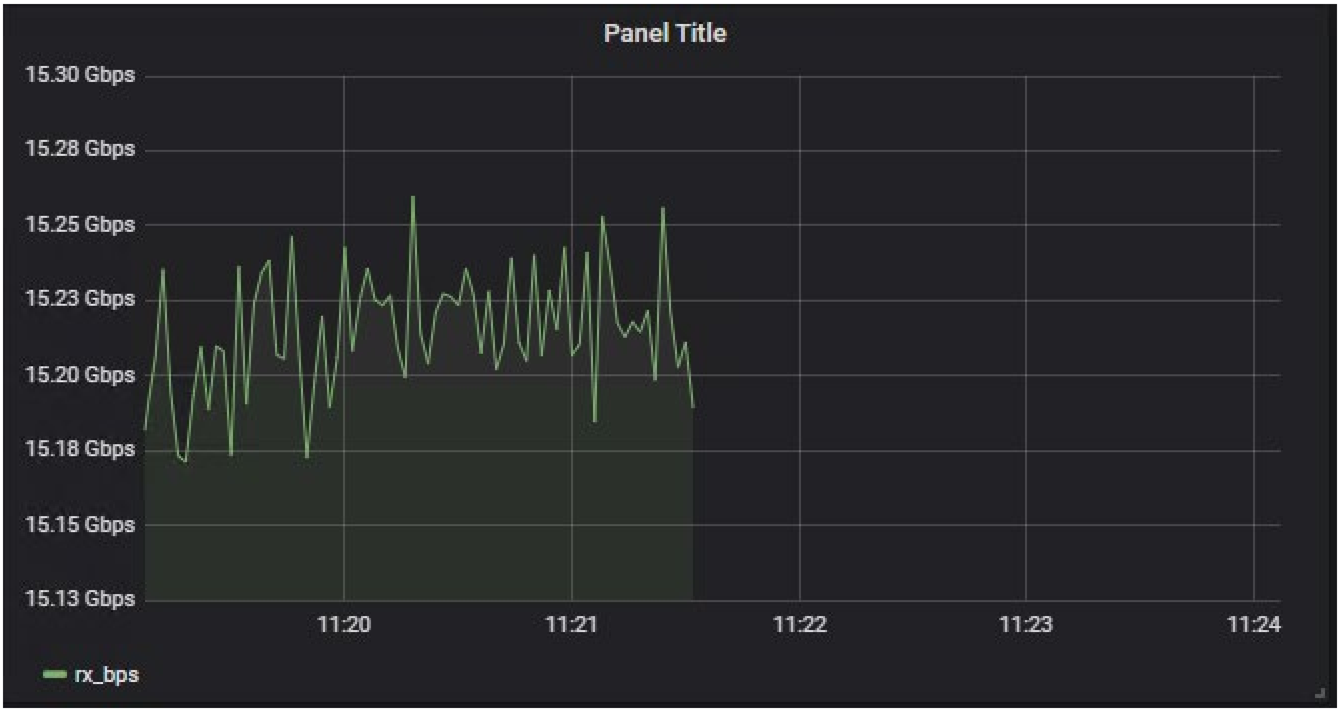
To summarize how the demonstration framework works, the following scenario assumes that the sample data points are viable. In the code sample below, there is a total of five minutes of data collected at 1-Hz frequency:

```
#TX bps        RX bps         TX pps      RX pps
33499588416.0 24600565248.0 49850578.0 36607984.0
33551246400.0 24606240288.0 49927450.0 36616429.0
33651676128.0 24604982976.0 50076899.0 36614558.0
33579966336.0 24619423584.0 49970188.0 36636047.0
33529855296.0 24634791552.0 49895618.0 36658916.0
33513466560.0 24630570720.0 49871230.0 36652635.0
……….
```

To playback the data point, add an entry into the InfluxDB* at a 1-Hz rate:

```
cl = InfluxDBClient(host='127.0.0.1', port=8086)
cl.switch_database('mydb')

f = open("/home/cs2019/"+file_name, "r")
lines = f.readlines()
f.close()

i = 0

# run this forever
while True:
    line = lines[i]
    i += 1

    if i == len(lines):
        # restart from beginning again, when reach last line
        i = 0

    # breaks line into list
    line = line.split('\n')
    data_points = line[0].split(' ')

    entry = [{"measurement" : 'performance', "fields" :
            {file_name+"_tx_bps" : float(data_points[0]),
             file_name+"_rx_bps" : float(data_points[1]),
             file_name+"_tx_pps" : float(data_points[2]),
             file_name+"_rx_pps" : float(data_points[3])}}]
    # write data point into database
    cl.write_points(entry)
    # wait for 1 second for 1Hz frequency
    sleep(1)
```

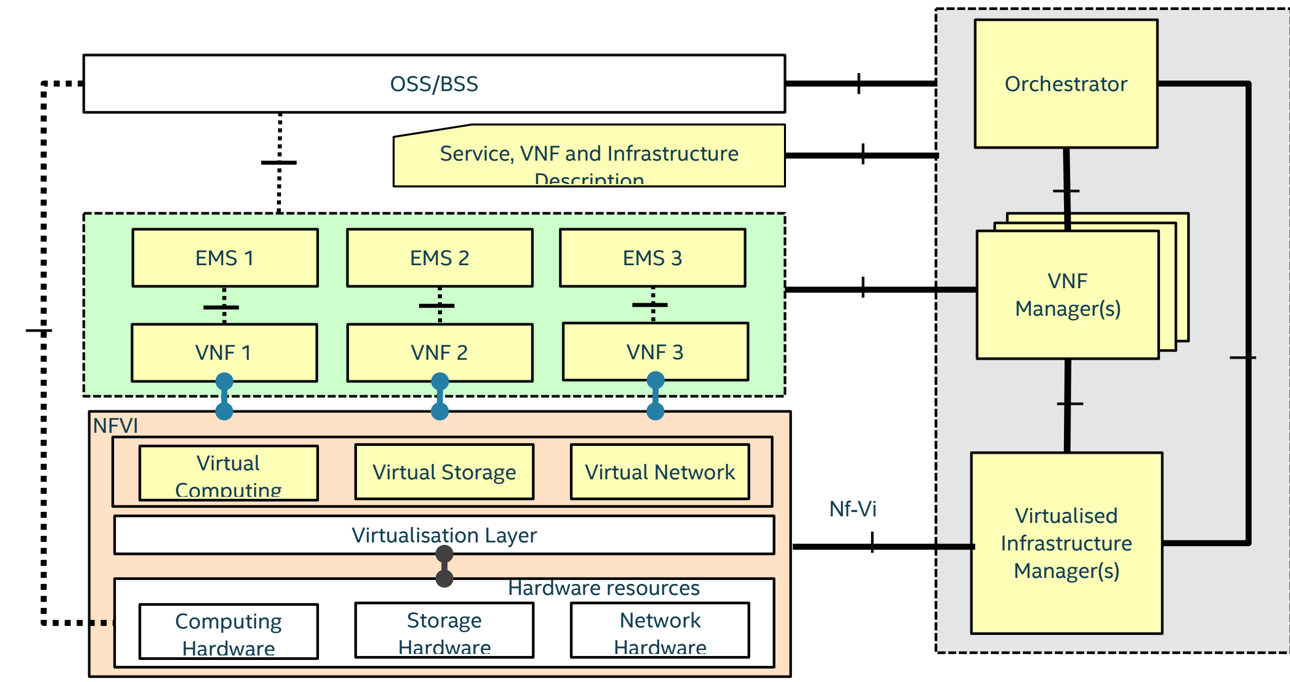
Then, by configuring Grafana*'s dashboard and panel, the user can create a real time graph of data rate forwarded by an NFV system.



The next sections and subchapters in this document explain the entire process from creating an NFV system to creating an offline demonstration.

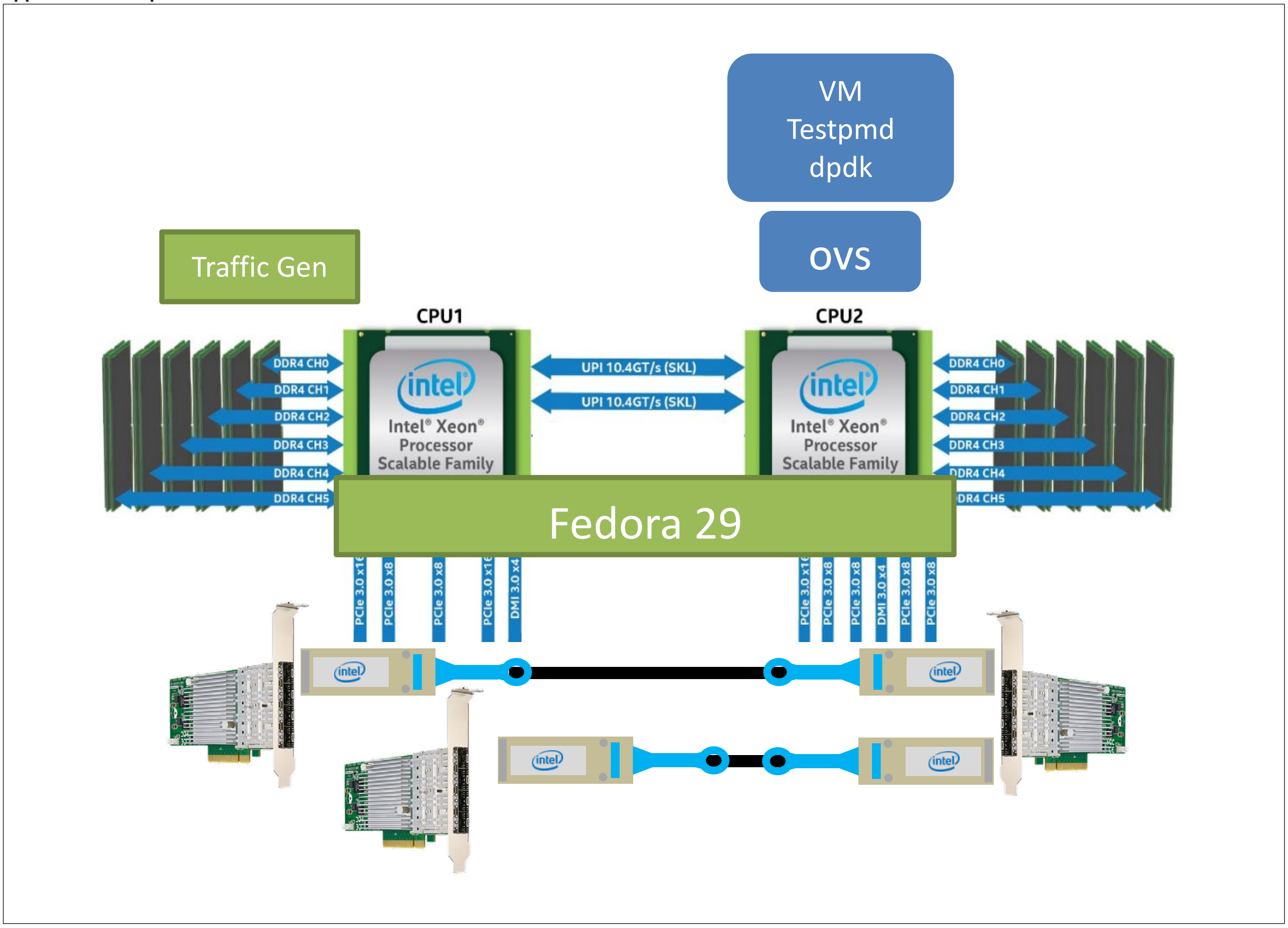
2.2 Building an NFV system

An NFV system as defined by ETSI are specify by the diagram below:-



Acronym Descriptions:
Operations Support System (OSS)
Business Support System (BSS)
Virtualize Network Function (VNF)
Element Management (EM)
NFV Infrastructure (NFVI)

An simplify NFV test environment used by this document specified by the diagram below:-



This simplest form of an NFV environment consists of preparing a Linux* system (refer to the Ubuntu* or Fedora* websites to prepare a Linux host system).

2.2.1 Preparation of Linux Environment

For the Red Hat Enterprise Linux* (RHEL*)/Fedora systems, install the following required software packages by executing the following commands:

```
$ dnf groupinstall "Development Tools"
$ yum install "kernel-devel-uname-r == $(uname -r)"
$ yum install python-six autoconf automake
```

For the Ubuntu/Debian* systems, install the following required software packages by executing the following commands:

```
$ apt install build-essential
$ apt install linux-headers-$(uname -r)
$ apt-get install python-six autoconf automake
```

Software Versions Needed for an NFV Environment:-

Software Needed for NFV Environment	Version
DPDK	dpdk-stable-17.11.4
OpenvSwitch	openvswitch-2.10.1
Qemu	qemu-2.12.1
Trex	Trex v2.35

The following commands are used to compile each software:

- Compile DPDK

```
$ cd /<dpdk_source_code>
$ make install T=x86_64-native-linuxapp-gcc DESTDIR=install
$ cd x86_64-native-linuxapp-gcc
$ EXTRA_CFLAGS="-Ofast" make -j3
```

- Compile OpenvSwitch

```
$ cd /<path_of_openvswitch_source_code>
$ ./boot.sh
$ ./configure --with-dpdk=/<dpdk_source_code>/x86_64-native-linuxapp-gcc CFLAGS="-Ofast" --disable-ssl
$ make CFLAGS="-Ofast -march=native" -j3
```

- Compile Qemu

```
cd /<qemu_source_code>
apt-get install -y  libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev
./configure --target-list=x86_64-softmmu
make -j10
```

2.3 Performance Test Scenarios

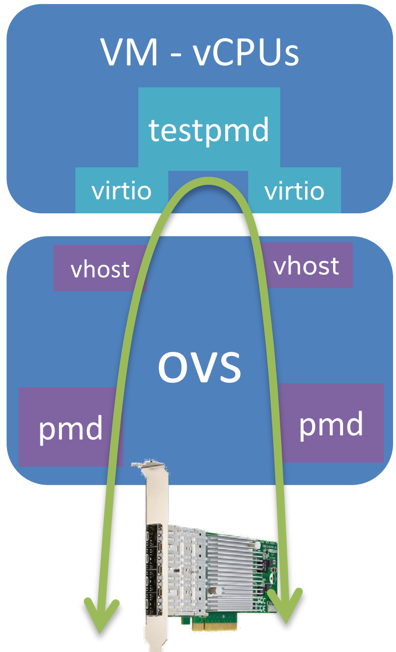
The NFV KPI performance test scenarios are designed to demonstrate the data plane forwarding capability of the host virtual switch moving data packets from the host physical ports to the Virtual Machine (VM).

The VM will also be running a simplified Virtualized Network Function (VNF) that forwards data packets, back and forth, from the virtio interface to the virtual switch.

The data path of this test goes from the physical port to the vswitch; after which, the virtio interface in the VM is forwarded by the VNF from another virtio interface. The vswitch then forwards the virtio interface to the physical port to complete the route.

The following software applications are used as the applications under test in the scenario above:

- The testpmd DPDK user-mode application: the DPDK is a set of libraries providing a programming framework to enable high speed data packet networking applications. The applications using DPDK libraries and interfaces run in user mode and directly interface with the Network Interface Card (NIC) functions, skipping slow, and kernel layer components to boost the packet processing performance and throughput. These applications process raw network packets without relying on the protocol stack functionality provided by kernel. For more information on the DPDK, go to http://www.dpdk.org.
- The OvS is an open-source implementation of a distributed virtual multilayer switch. The main purpose of the OvS is to provide a switching stack for hardware virtualization environments while supporting multiple protocols and standards used in computer networks. It is optimized with DPDK libraries to deliver an improved performance in comparison with the kernel-based data plane.



Acronym Descriptions:
Virtual Machine (VM)
Virtual Central Processing Unit (vCPU)
Virtual Host (vHost)
Poll Mode Driver (PMD)

Third-party Trademark Description:
Open VSwitch * (OvS*)

When the required software components are compiled, launch the applications.

2.4 Preparation for OpenvSwitch:-

```
export DPDK_DIR=/<dpdk_source_code>
export DPDK_BUILD=$DPDK_DIR/x86_64-native-linuxapp-gcc
export OVS_DIR=/<openvsiwtch_source_code>

echo 32 > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr_hugepages
echo 32 > /sys/devices/system/node/node1/hugepages/hugepages-1048576kB/nr_hugepages

umount /dev/hugepages
mount -t hugetlbfs nodev /dev/hugepages -o pagesize=1GB

rmmod i40e
rmmod igb_uio
rmmod cuse
rmmod fuse
rmmod openvswitch
rmmod uio
rmmod eventfd_link
rmmod ioeventfd
rm -rf /dev/vhost-net


modprobe uio
insmod $DPDK_BUILD/kmod/igb_uio.ko

python $DPDK_DIR/usertools/dpdk-devbind.py --bind=igb_uio <NIC1_B:D:F>
python $DPDK_DIR/usertools/dpdk-devbind.py --bind=igb_uio <NIC2_B:D:F>
# terminate OVS

pkill -9 ovs
rm -rf /usr/local/var/run/openvswitch
```

```
rm -rf /usr/local/etc/openvswitch/
rm -rf /usr/local/var/log/openvswitch
rm -f /tmp/conf.db

mkdir -p /usr/local/etc/openvswitch
mkdir -p /usr/local/var/run/openvswitch
mkdir -p  /usr/local/var/log/openvswitch
```

2.4.1 Steps to Initialize a New OvS Database

1. Before launching the OvS daemon "ovs-vswitchd," it is necessary to initialize the OvS database and start the ovsdb-server. The following commands show how to clear and create a new OvS database and an ovsdb-server instance:

```
cd $OVS_DIR
./ovsdb/ovsdb-tool create /usr/local/etc/openvswitch/conf.db ./vswitchd/vswitch.ovsschema

# Starting OpenvSwitch database server:-
./ovsdb/ovsdb-server --remote=punix:/usr/local/var/run/openvswitch/db.sock \
                 --remote=db:Open_vSwitch,Open_vSwitch,manager_options \
                 --pidfile --detach

# Initialize OpenvSwitch database:-
./utilities/ovs-vsctl --no-wait init
```

2. Start the OvS portion using 1 GB:

```
export DB_SOCK=/usr/local/var/run/openvswitch/db.sock

#Edit this section for the ovs-vswitchd to be on Core 1 (in socket0) or core X (in remote socket).
By default it is set to core 1 (local socket0)
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true other_config:dpdk-
lcore-mask=<Core_Mask> other_config:dpdk-socket-mem="2048,2048"
```

3. Locate the OvS log file at /usr/local/var/log/openvswitch/ovs-vswitchd.log:

```
./vswitchd/ovs-vswitchd unix:$DB_SOCK --pidfile --detach  --log-
file=/usr/local/var/log/openvswitch/ovs-vswitchd.log

$ OVS_DIR/utilities/ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=<Core Mask>
$ OVS_DIR/utilities/ovs-vsctl set Open_vSwitch . other_config:max-idle=30000
```

4. The following steps create a bridge with two physical ports and two back-end Virtual Hosts (vHosts) to support two virtio interfaces in the VM:

```
./utilities/ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev
ifconfig br0 0 up
./utilities/ovs-vsctl add-port br0 dpdk0 -- set Interface dpdk0 type=dpdk options:dpdk-
devargs=<NIC1_B:D:F> ofport_request=1
sleep 8
./utilities/ovs-vsctl add-port br0 dpdk1 -- set Interface dpdk1 type=dpdk options:dpdk-
devargs=<NIC2_B:D:F> ofport_request=2
sleep 8

#Create vhost-user interfaces
./utilities/ovs-vsctl add-port br0 vhost-user0 -- set Interface vhost-user0 type=dpdkvhostuser
ofport_request=3
./utilities/ovs-vsctl add-port br0 vhost-user1 -- set Interface vhost-user1 type=dpdkvhostuser
```

```
ofport_request=4

./utilities/ovs-vsctl show
```

5. Once OvS is running, the next step is to start the VM:

```
<qemu_source_code>/x86_64-softmmu/qemu-system-x86_64 -m 4G -smp 3,cores=3,threads=1,sockets=1 -cpu 
host -drive format=raw,file="+main_path+"vm-images/ubuntu-16.04-testpmd.img -boot c -enable-kvm -
name VNF -object memory-backend-file,id=mem,size=4G,mem-path=/dev/hugepages,share=on -numa 
node,memdev=mem -mem-prealloc -netdev user,id=nttsip,hostfwd=tcp::2024-:22 -device 
e1000,netdev=nttsip -chardev socket,id=char1,path=/usr/local/var/run/openvswitch/vhost-user0 -
netdev type=vhost-user,id=net1,chardev=char1,vhostforce -device virtio-net-
pci,netdev=net1,mac=00:01:00:00:00:01,csum=off,gso=off,guest_tso4=off,guest_tso6=off,guest_ecn=off,
mrg_rxbuf=off -chardev socket,id=char2,path=/usr/local/var/run/openvswitch/vhost-user1 -netdev 
type=vhost-user,id=net2,chardev=char2,vhostforce -device virtio-net-
pci,netdev=net2,mac=00:02:00:00:00:02,csum=off,gso=off,guest_tso4=off,guest_tso6=off,guest_ecn=off,
mrg_rxbuf=off -vnc :1 -daemonize
```

6. With the VM powered on, the last step is to start the testpmd application to forward packets between two virtio interfaces:

```
# First logging to the VM via ssh root@localhost -p 2024
#export DPDK_DIR=/root/dpdk-stable-17.05.1; rmmod igb_uio; modprobe uio; insmod $DPDK_DIR/x86_64-
native-linuxapp-gcc/kmod/igb_uio.ko

$DPDK_DIR/usertools/dpdk-devbind.py -b igb_uio 00:04.0
$DPDK_DIR/usertools/dpdk-devbind.py -b igb_uio 00:05.0
#DPDK_DIR/x86_64-native-linuxapp-gcc/app/testpmd -c 0x6 -n 4 -- --burst=64 --txd=2048 --rxd=2048 --
txqflags=0xf00 --disable-hw-vlan
```

With the steps completed, an NFV system with a simple VNF is now ready.

2.5 Traffic Generator

In this configuration, the traffic generator runs on another CPU socket to avoid it from interfering with the NFV system, and the VNF runs across the Intel® Ultra Path Interconnect (Intel® UPI) on the opposing socket. To get the TRex traffic generator up and running, follow the next instructions:

- Download Trex software package

```
wget --no-check-certificate http://trex-tgn.cisco.com/trex/release/v2.35.tar.gz
```

- Extract and setup Trex

```
tar xzvf v2.35.tar.gz
cd v2.35
vi config.yaml
```

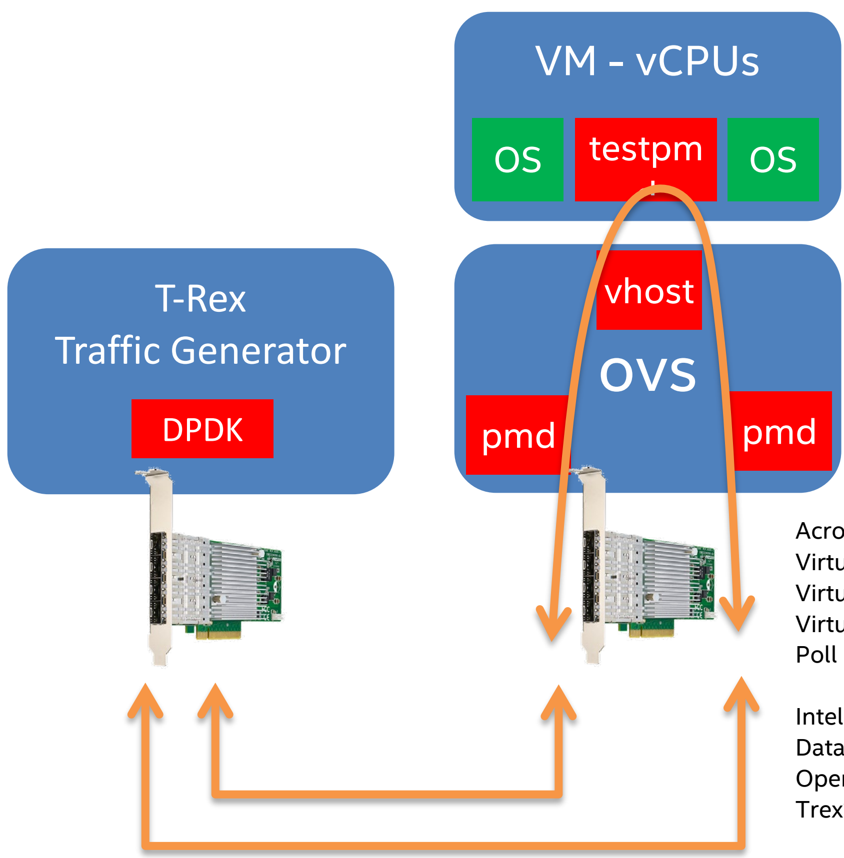
```
### Config file generated by dpdk_setup_ports.py ###

- port_limit: 2
  version: 2
  interfaces: ['<NIC1_B:D:F>', '<NIC2_B:D:F>']
  limit_memory    : 2048
  rx_desc         : 4096
  tx_desc         : 4096
  prefix: trex
  port_info:
      - dest_mac: 00:01:02:03:04:05 # MAC OF LOOPBACK TO IT'S DUAL INTERFACE
        src_mac:  00:11:22:33:44:55
      - dest_mac: 00:11:22:33:44:55 # MAC OF LOOPBACK TO IT'S DUAL INTERFACE
        src_mac:  00:01:02:03:04:05
```

```
platform:
    master_thread_id: 1
    latency_thread_id: 2
    dual_if:
      - socket: 0
        threads: [<Provide 8 CORES ID>]
```

- To run TRex: -

`./t-rex-64 --cfg config.yaml -i -c 8`

Traffic Generator Workflow:-



Acronym Descriptions:
Virtual Machine (VM)
Virtual Central Processing Unit (vCPU)
Virtual Host (vHost)
Poll Mode Driver (PMD)

Intel and Third-party Trademark Description:
Data Plane Development Kit* (DPDK*)
Open VSwitch * (OvS*)
Trex*

Another key component to this framework is to use a traffic generator with a Python Application Programming Interface (API) which is convenient for controlling the traffic and packet rates that can be collected from the traffic generator at a fixed interval.

By executing the steps from A to H, as shown in the following figure, the traffic will be flowing across the virtual switch to the VNF in the VM and backout through another port.

Traffic Flow Steps:-

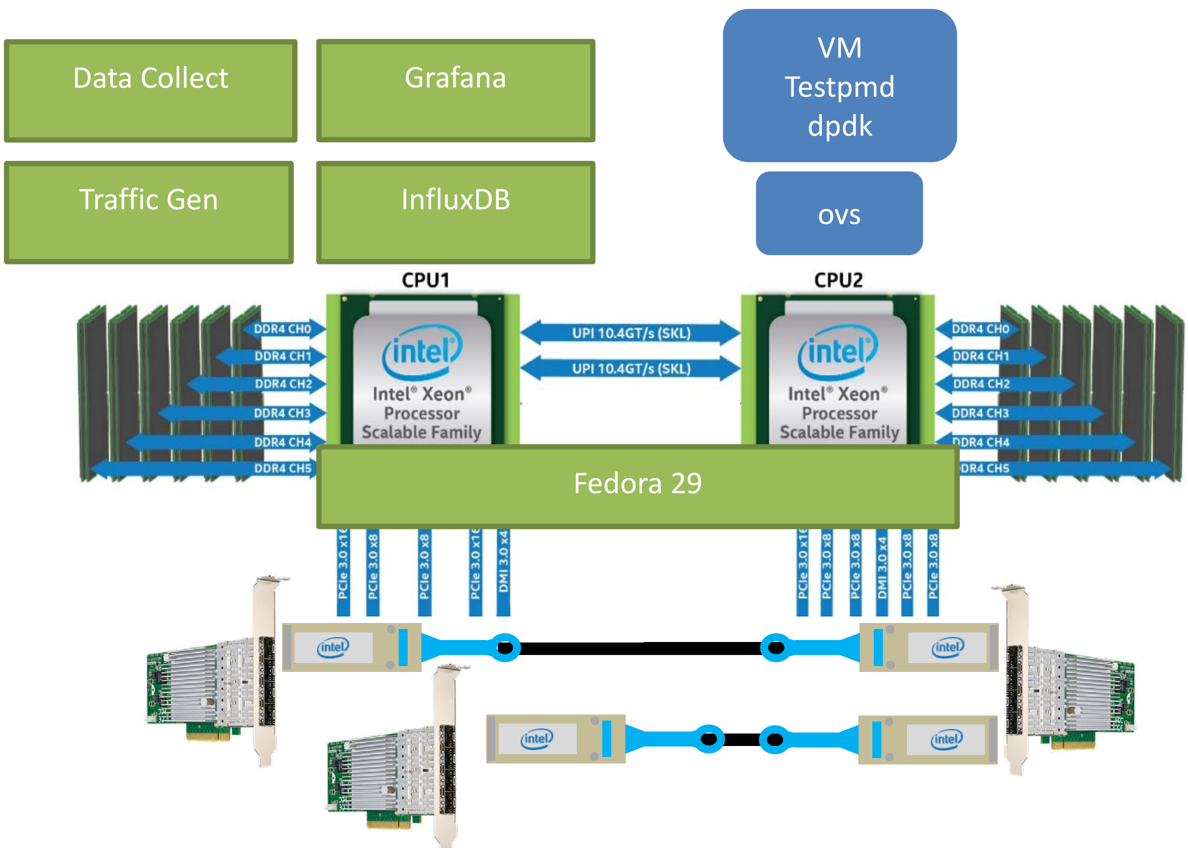
```
------Intel NFVi Enabling Kit-----------
[a] Select Cores
[b] Select NICs
[c] Start OVS
[d] Create PVP
[e] Start VM
[f] Start VM fwd
[g] Start Trex backend
[h] Start Traffic
```

2.6 Data Collection

There are two parts in the data collection to complete the framework:

1. The screen capture of the execution of the traffic generator and VM VNF.
2. The data needed to be collected here are Packets per Second (PPS) to measure the data forwarding rate of an NFV system and a VNF.

Integration of Data Collection Components: -



Acronym Descriptions:
Virtual Machine (VM)
Central Processing Unit (CPU)
Double Data Rate (DDR)
Channel (CH)

Intel and Third-party Trademark Descriptions:
Data Plane Development Kit* (DPDk*)
Open vSwitch* (OvS*)
PCIe*
Intel® Xeon® Processor Scalable Family
Intel® Ultra Path Interconnect (Intel® UPI)

This document uses InfluxDB to store the NFV KPI. InfluxDB is the database and purpose-built storage engine to handle time series data; it is a metric store for multiple data sources to help you avoid a siloed approach.

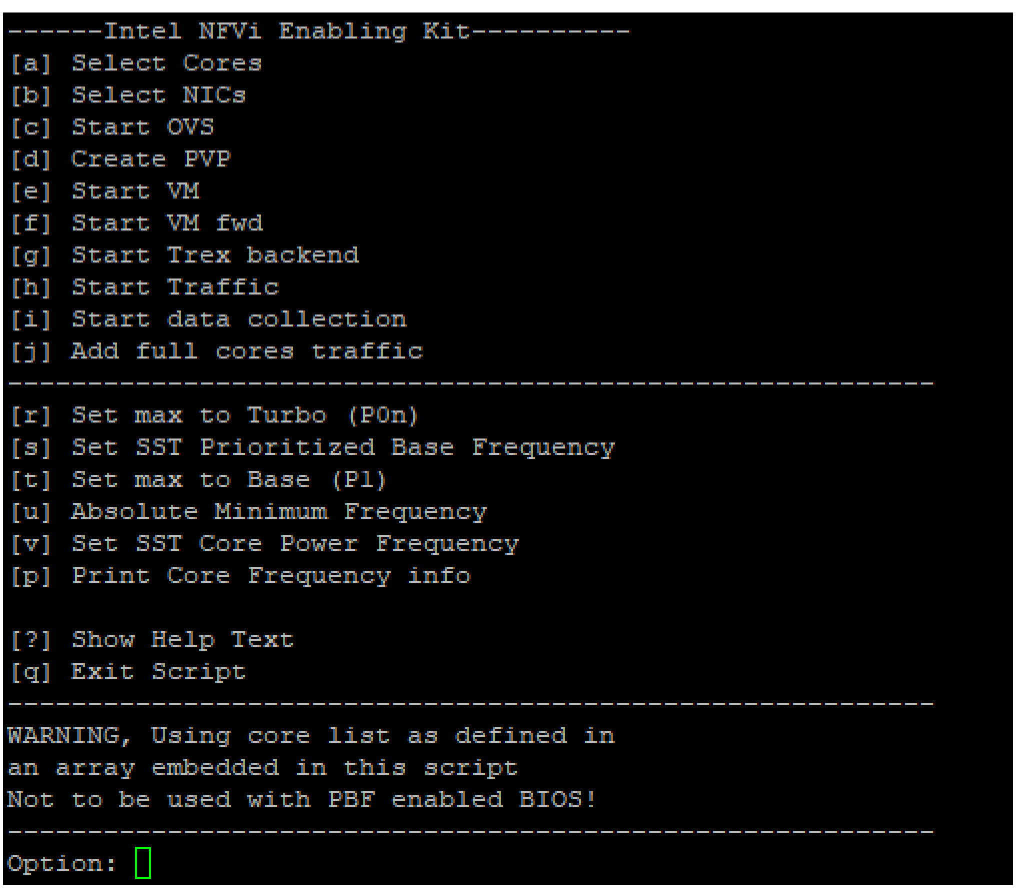
```
#Fedora: -
cat <<EOF | sudo tee /etc/yum.repos.d/InfluxDB.repo
[InfluxDB]
name = InfluxDB Repository - RHEL
baseurl = https://repos.influxdata.com/rhel/7/x86_64/stable/
enabled = 1
gpgcheck = 1
gpgkey = https://repos.influxdata.com/InfluxDB.key
EOF
dnf -y install InfluxDB

#Ubuntu: -
curl -sL https://repos.influxdata.com/InfluxDB.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" | sudo tee
/etc/apt/sources.list.d/InfluxDB.list
sudo apt-get update && sudo apt-get install InfluxDB
```

To display the data points, this document uses Grafana; it is an open source analytics and monitoring solution for every database. Grafana allows the user to query, visualize, alert on and understand the NFV KPI metrics wherever they are stored. To install Grafana, run the following commands:

```
#Ferdora:-
yum install initscripts urw-fonts
wget https://dl.grafana.com/oss/release/grafana-5.4.2-1.x86_64.rpm
sudo yum localinstall grafana-5.4.2-1.x86_64.rpm

#Ubuntu: -
wget https://dl.grafana.com/oss/release/grafana_5.4.2_amd64.deb
sudo dpkg -i grafana_5.4.2_amd64.deb
```

The following figures show a created console providing the live data feed of core utilization, traffic generator, and testpmd's packet rates:

Main control windows



Trex statistic window:-

```
-Per port stats table
      ports |               0 |               1
 -----------------------------------------------------------------------------------------
   opackets |    733062437422 |    733062435069
     obytes | 187663983976704 | 187663983373568
   ipackets |    224868023114 |    224868026533
     ibytes |  57566213608902 |  57566214481094
    ierrors |               0 |               0
    oerrors |               0 |               0
      Tx Bw |      23.08 Gbps |      23.08 Gbps

-Global stats enabled
 Cpu Utilization : 8.5  %  135.8 Gb/core
 Platform_factor : 1.0
 Total-Tx        :      46.16 Gbps
 Total-Rx        :      14.15 Gbps
 Total-PPS       :      22.54 Mpps
 Total-CPS       :       0.00  cps

 Expected-PPS    :       0.00  pps
 Expected-CPS    :       0.00  cps
 Expected-BPS    :       0.00  bps

 Active-flows    :        0  Clients :        0   Socket-util : 0.0000 %
 Open-flows      :        0  Servers :        0   Socket :        0 Socket/Clients :  -nan
 Total_queue_full : 431473
 drop-rate       :      32.01 Gbps
 current time    : 64799.4 sec
 test duration   : 0.0 sec
```

Testpmd PPS window

```
  ######################## NIC statistics for port 1  ########################
  RX-packets: 584646509600 RX-missed: 0          RX-bytes:  0
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 225018461518 TX-errors: 0          TX-bytes:  56704652302536

  Throughput (since last show)
  Rx-pps:      9010814
  Tx-pps:      3452120
  ############################################################################
testpmd> show port stats all

  ######################## NIC statistics for port 0  ########################
  RX-packets: 598747997504 RX-missed: 0          RX-bytes:  0
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 225028849007 TX-errors: 0          TX-bytes:  56707269949764

  Throughput (since last show)
  Rx-pps:      9215548
  Tx-pps:      3450352
  ############################################################################

  ######################## NIC statistics for port 1  ########################
  RX-packets: 584673640048 RX-missed: 0          RX-bytes:  0
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 225028852694 TX-errors: 0          TX-bytes:  56707270878888

  Throughput (since last show)
  Rx-pps:      9008610
  Tx-pps:      3450368
  ############################################################################
testpmd>
```

CPU utilization window:-

```
  1  [|                3.9%]   17 [                 0.0%]   33 [                 0.0%]   49 [                 0.0%]
  2  [||               6.0%]   18 [                 0.0%]   34 [                 0.0%]   50 [|                0.7%]
  3  [||||||||||||||100.0%]   19 [                 0.0%]   35 [                 0.0%]   51 [                 0.0%]
  4  [||||||||||||||100.0%]   20 [                 0.0%]   36 [                 0.0%]   52 [                 0.0%]
  5  [||||||||||||||100.0%]   21 [                 0.0%]   37 [                 0.0%]   53 [                 0.0%]
  6  [||||||||||||||100.0%]   22 [                 0.0%]   38 [                 0.0%]   54 [                 0.0%]
  7  [||||||||||||||100.0%]   23 [                 0.0%]   39 [                 0.0%]   55 [                 0.0%]
  8  [||||||||||||||100.0%]   24 [||||||||||||||100.0%]   40 [                 0.0%]   56 [                 0.0%]
  9  [||||||||||||||100.0%]   25 [||||||||||||||100.0%]   41 [                 0.0%]   57 [                 0.0%]
  10 [||||||||||||||100.0%]   26 [||||||||||||||100.0%]   42 [                 0.0%]   58 [                 0.0%]
  11 [||||||||||||||100.0%]   27 [                 0.0%]   43 [                 0.0%]   59 [                 0.0%]
  12 [                 0.0%]   28 [|                0.7%]   44 [                 0.0%]   60 [                 0.0%]
  13 [                 0.0%]   29 [                 0.0%]   45 [                 0.0%]   61 [                 0.0%]
  14 [                 0.0%]   30 [|                1.3%]   46 [                 0.0%]   62 [                 0.0%]
  15 [                 0.0%]   31 [||||||||||||||100.0%]   47 [                 0.0%]   63 [                 0.0%]
  16 [                 0.0%]   32 [|                0.7%]   48 [                 0.0%]   64 [                 0.0%]
  Mem[||||||||||||||                         39.8G/187G]   Tasks: 66, 556 thr; 14 running
  Swp[                                        0K/4.00G]   Load average: 13.02 13.05 13.07
                                                          Uptime: 9 days, 17:31:10

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
51980 root       20   0 5050M 16292  7316 S 904.  0.0    163h ./_t-rex-64 --cfg config.yaml -i -c 8
51358 root       20   0 18.7G 43940  6360 S 299.  0.0 54h20:20 ./vswitchd/ovs-vswitchd unix:/usr/local/var/run/openvsw
51564 root       20   0 4931M 19936  7132 S 100.  0.0 18h08:13 /home/ovs-dpdk/qemu-2.12.1/x86_64-softmmu/qemu-system-x
51540 root       20   0 18.7G 43940  6360 R 99.7  0.0 18h06:40 ./vswitchd/ovs-vswitchd unix:/usr/local/var/run/openvsw
51541 root       20   0 18.7G 43940  6360 R 99.7  0.0 18h06:30 ./vswitchd/ovs-vswitchd unix:/usr/local/var/run/openvsw
51539 root       20   0 18.7G 43940  6360 R 99.7  0.0 18h05:48 ./vswitchd/ovs-vswitchd unix:/usr/local/var/run/openvsw
51578 root       20   0 4931M 19936  7132 R 99.7  0.0 18h04:17 /home/ovs-dpdk/qemu-2.12.1/x86_64-softmmu/qemu-system-x
51987 root       20   0 5050M 16292  7316 R 99.7  0.0 18h01:16 ./_t-rex-64 --cfg config.yaml -i -c 8
51991 root       20   0 5050M 16292  7316 R 99.7  0.0 18h01:16 ./_t-rex-64 --cfg config.yaml -i -c 8
51989 root       20   0 5050M 16292  7316 R 100.  0.0 18h01:15 ./_t-rex-64 --cfg config.yaml -i -c 8
51990 root       20   0 5050M 16292  7316 R 99.7  0.0 18h01:15 ./_t-rex-64 --cfg config.yaml -i -c 8
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

For screen capture, this process uses asciinema* as the tool to record and share the terminal sessions; it is lightweight, text-based approach to terminal recording and allows the console execution to be played back as if it was the real execution of the application.

The method consists of splitting the recording into two parts; start by recording the setup until the traffic starts flowing and then stop the recording.

Next, record the console screen for the continuous flow of packets across the OvS and the VM's testpmd for a fix duration (for example, five minutes).

The recording should also collect the same amount of data points to match the screen recording (for example, five minutes). As a result, start collecting the NFV KPI data (for example, packets per second) for every second interval at 1-Hz frequency.

Ensure that the backend TRex is running in the background to create traffic streams; this demonstration uses the TRex Python API to configure and start the transmission of the traffic.

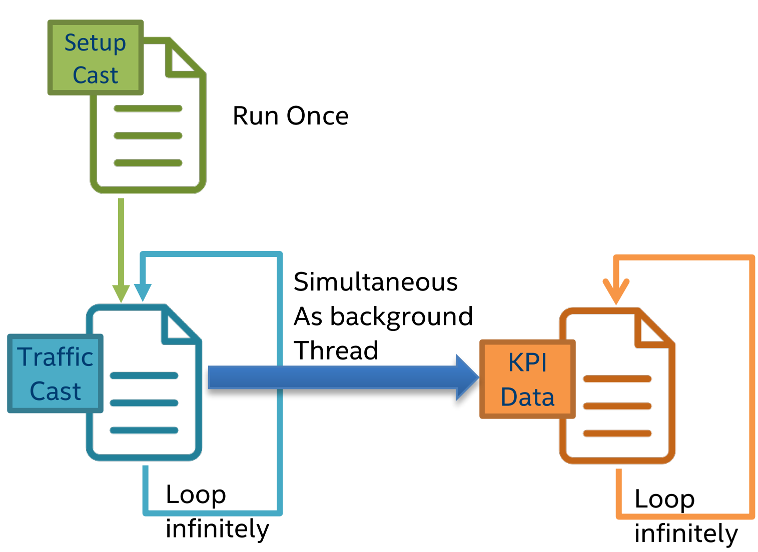
```
# connect to trex backend
trex_client = STLClient(username = "root", server="127.0.0.1")
trex_client.connect()

# prepare
trex_client.reset(ports=[0,1])
trex_client.clear_stats()
trex_client.set_port_attr(ports = [0,1], promiscuous=True)

# Create a packet contents, based on scapy python api
base_pkt =
Ether(dst='00:02:00:00:00:02')/IP(src="10.2.2.22",dst="10.1.1.11")/UDP(dport=5201,sport=1025)
        pad = max(0, pkt_size - len(base_pkt)) * 'x'

# Create a stream, with attribute of base_pkt, inter stream gap, and statistic collection
s0 = STLStream( isg = 0.0, name='S0', packet = STLPktBuilder(pkt = base_pkt/pad), mode = STLTXCont(
percentage = 100), flow_stats = STLFlowStats(pg_id = 0))

# add both streams to the desire port
trex_client.add_streams(s0, ports = [0])

# clear the stats before injecting
trex_client.clear_stats()
```

```
# start transmission of the traffic
trex_client.start(ports = [0], mult = "100%", duration = -1, core_mask = STLClient.CORE_MASK_PIN)
```

The next sample code is to collect the statistics from TRex and save the results as data points:

```
# Connect to trex backend
trex_client = STLClient(username = "root", server="127.0.0.1")
trex_client.connect()

# the flow stats is state at pgids = 1
fs = trex_client.get_pgid_stats()
tx_bps_l1 = fs['flow_stats'][fs_pgids[1]]['tx_bps_l1']['total']
tx_pps     = fs['flow_stats'][fs_pgids[1]]['tx_pps']['total']
rx_pps    = fs['flow_stats'][fs_pgids[1]]['rx_pps']['total']

# final formatting the data
tx_bps_l1 = float(tx_bps_l1)
tx_pps    = float(tx_pps)
rx_pps    = float(rx_pps)
# need to calculate this value due do trex value not correct
rx_bps_l1 = float(rx_pps * (pkt_size + 24) * 8)

# a single line in the text file is 1 second of data.
data_points = str(tx_bps_l1) + ' ' + str(rx_bps_l1) + ' ' + str(tx_pps) + ' ' + str(rx_pps) + '\n'

# write data point into a file.
f = open(path_download+"/result.txt", "w")
f.write(data_points)
f.close()
```

Interaction between the asciinema Recording and the Data Point Playback :-



This scenario assumes that the following sample data points are viable. In the following code sample, there is a total of five minutes of data collected at 1-Hz frequency:

```
#TX bps       RX bps        TX pps     RX pps
33499588416.0 24600565248.0 49850578.0 36607984.0
33551246400.0 24606240288.0 49927450.0 36616429.0
33651676128.0 24604982976.0 50076899.0 36614558.0
33579966336.0 24619423584.0 49970188.0 36636047.0
33529855296.0 24634791552.0 49895618.0 36658916.0
33513466560.0 24630570720.0 49871230.0 36652635.0
……….
```

To play the screen playback, run the python code below:

```
os.system("asciinema play -s 1 /<your_recording_path>/setup.cast")
# run this forever
while True:
    os.system('clear')
    os.system("asciinema play -s 1 /<your_recording_path>/traffic.cast")
    time.sleep(1)
```

To playback the data point, , run the python code below:-

```
cl = InfluxDBClient(host='127.0.0.1', port=8086)
cl.switch_database('mydb')

f = open("/home/cs2019/"+file_name, "r")
lines = f.readlines()
f.close()

i = 0
# run this forever
while True:
    line = lines[i]
    i += 1

    if i == len(lines):
        # restart from beginning again, when reach last line
        i = 0

    # breaks line into list
    line = line.split('\n')
    data_points = line[0].split(' ')

    entry = [{"measurement" : 'performance', "fields" :
            {file_name+"_tx_bps" : float(data_points[0]),
             file_name+"_rx_bps" : float(data_points[1]),
             file_name+"_tx_pps" : float(data_points[2]),
             file_name+"_rx_pps" : float(data_points[3])}}]
    # write data point into database
    cl.write_points(entry)
    # wait for 1 second for 1Hz frequency
    sleep(1)
```

Or to combine the play back screen recording and the data point update to the database, create a function based on the code above (for example, a function name as “start_update_db”):

```
os.system("asciinema play -s 1 /<your_recording_path>/setup.cast")
# right after setup playback is done, start updating database
t = threading.Thread(target=start_update_db,args=[<your_data_point_file>])
t.start()
threads.append(t)
# run this forever
while True:
    os.system('clear')
    os.system("asciinema play -s 1 /home/cs2019/traffic_"+sys.argv[2]+".cast")
    time.sleep(1)
```

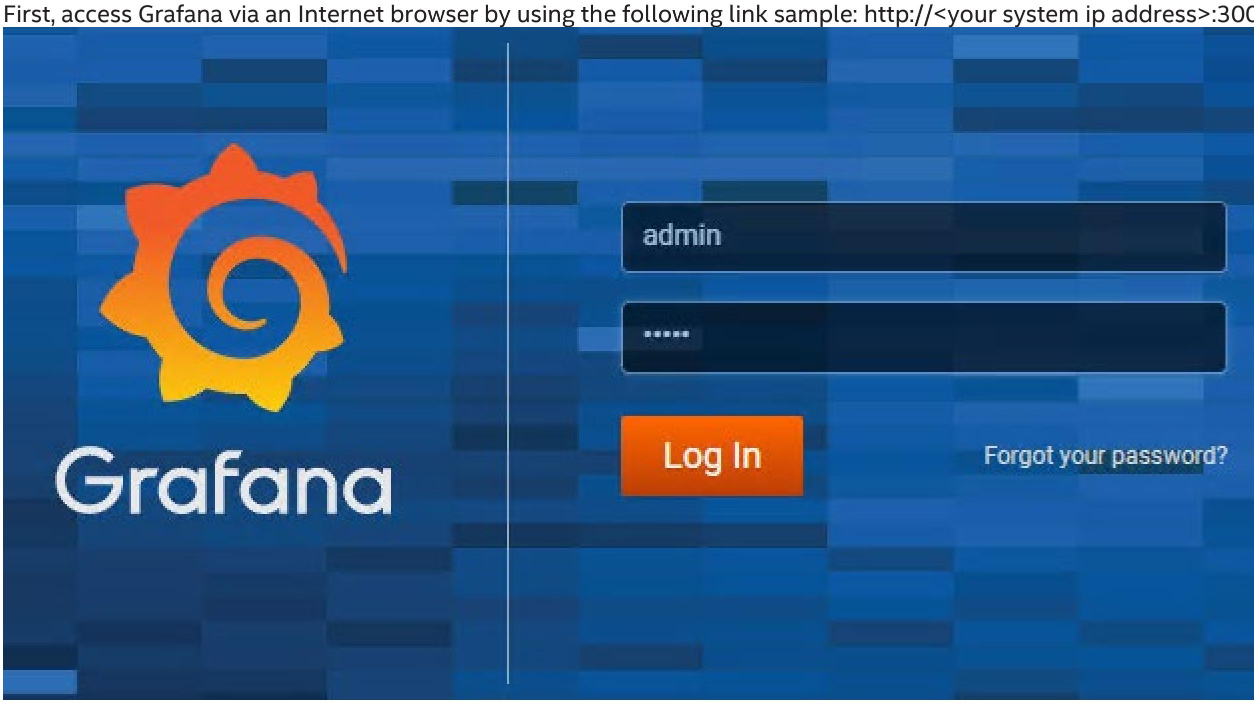
2.7 Grafana for Data Visualization

Grafana is used to query, to visualize, and to store data points in InfluxDB.

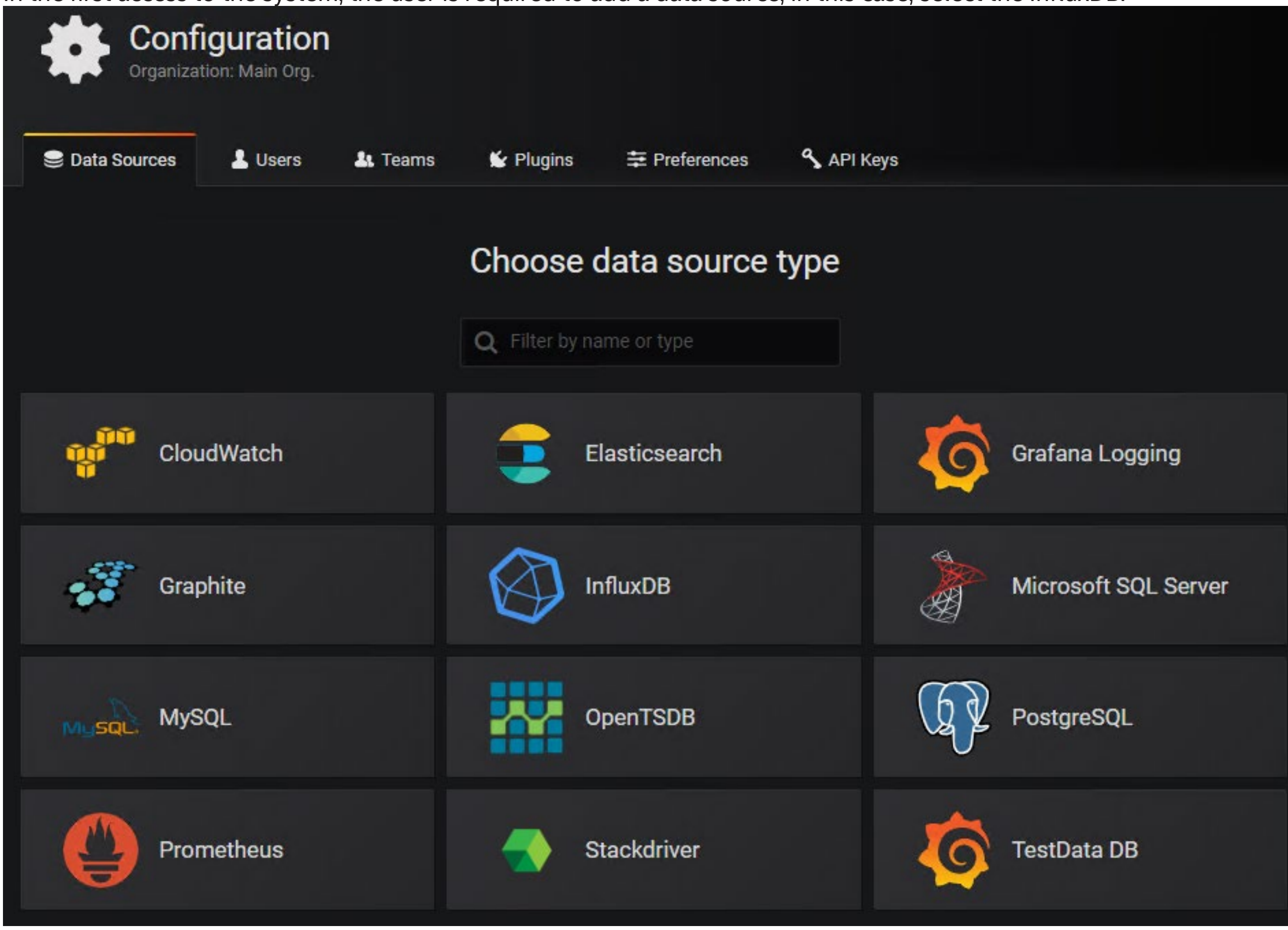
With a built-in function, this demonstration creates a time-based graph to visualize the incoming NFV KPIs since the function automatically queries the new entries in the InfluxDB.

The following guide shows how to create a graph to show the latest performance of the NFV system.

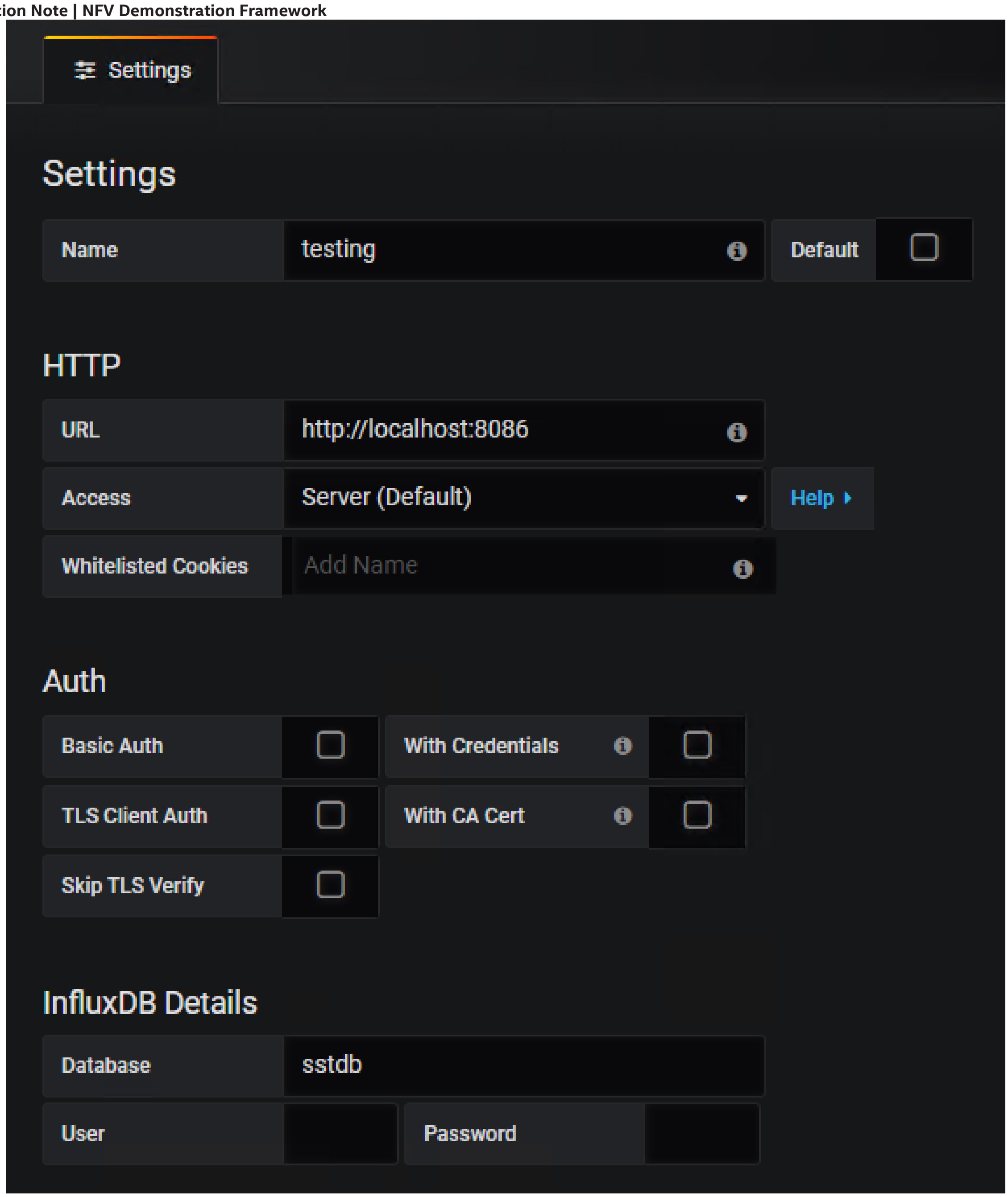
1. First, access Grafana via an Internet browser by using the following link sample: http://<your system ip address>:3000



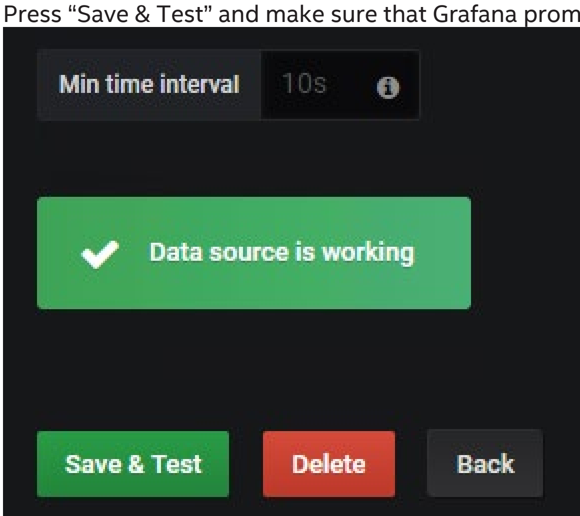
2. In the first access to the system, the user is required to add a data source; in this case, select the InfluxDB.



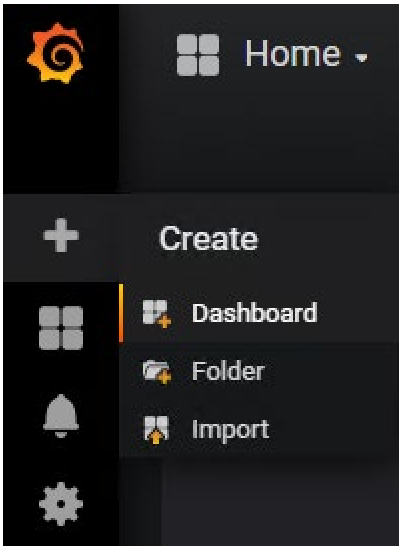
3. The user must update the Uniform Resource Locator (URL) of InfluxDB (this document uses http://localhost:8086 because both InfluxDB and Grafana reside on the same system) so that Grafana is able to connect to it and to provide the database name.



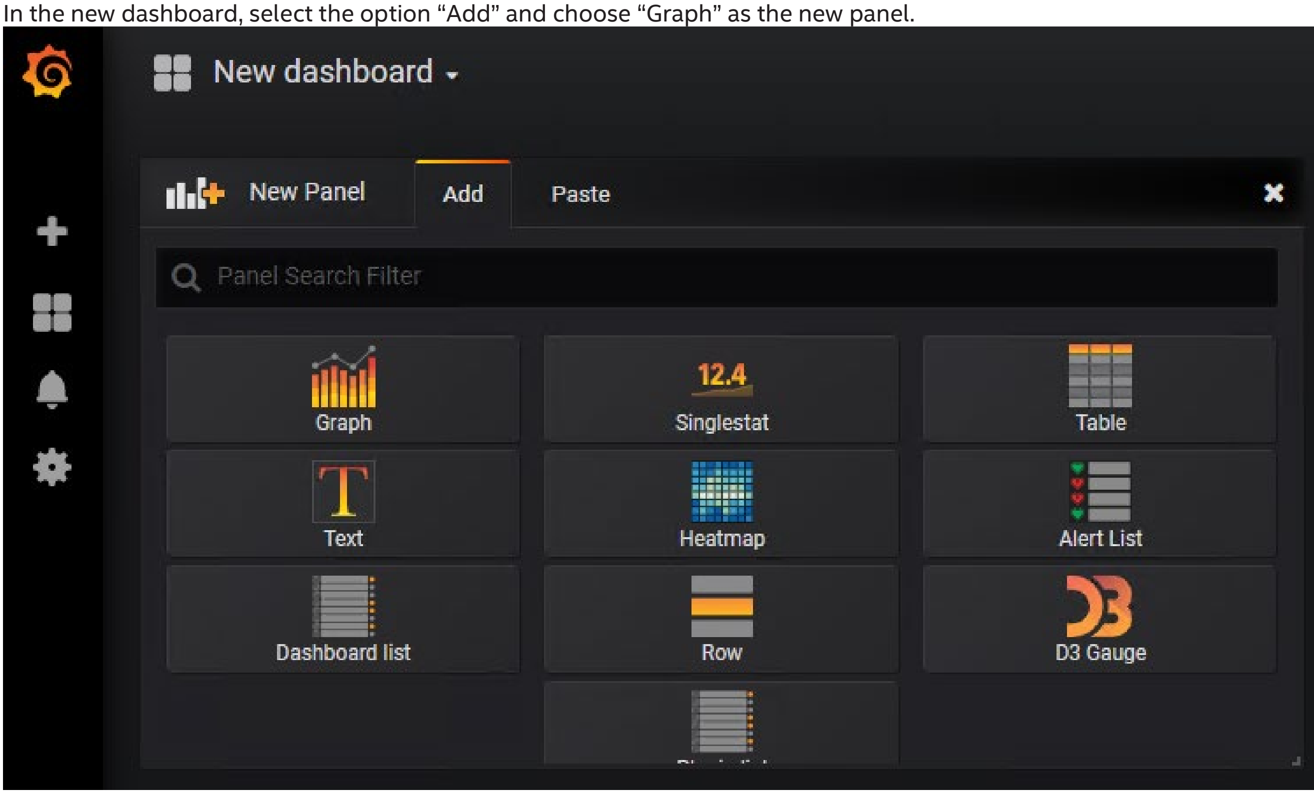
4. Press “Save & Test” and make sure that Grafana prompts back the message “Data source is working.”



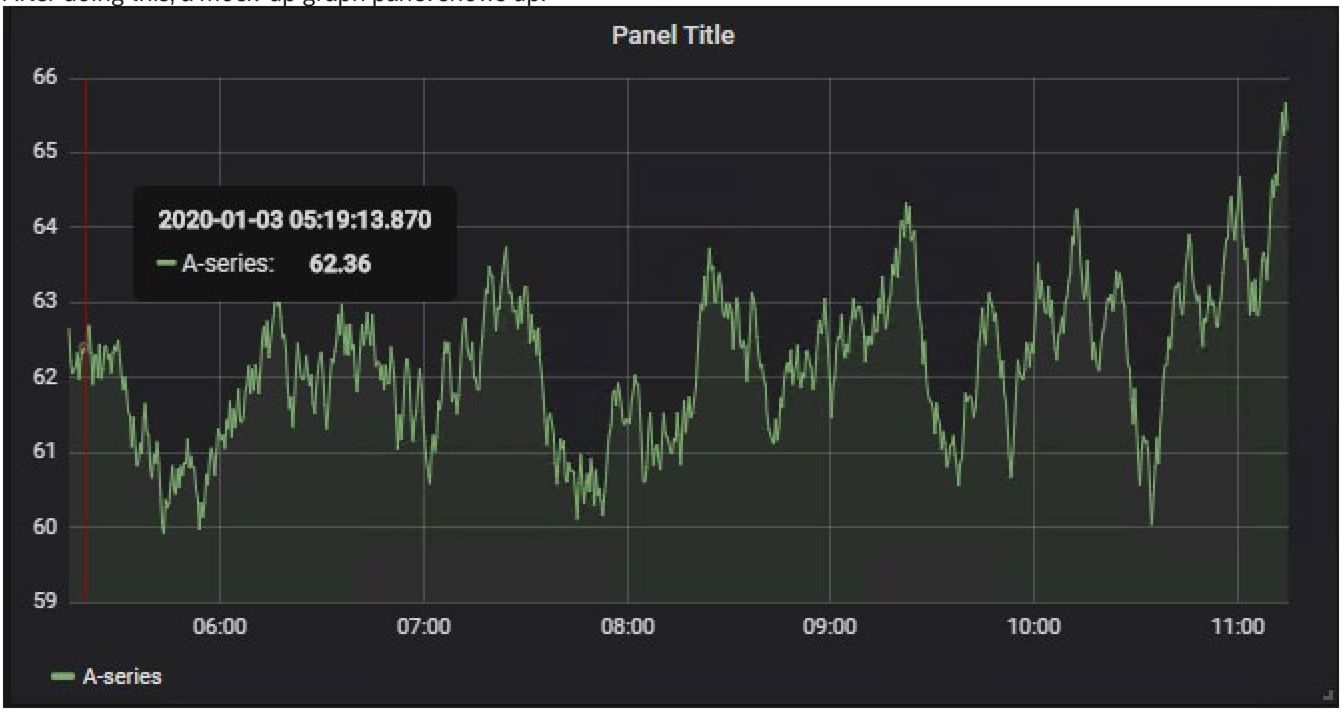
5. Click the "+" button and create a new dashboard.

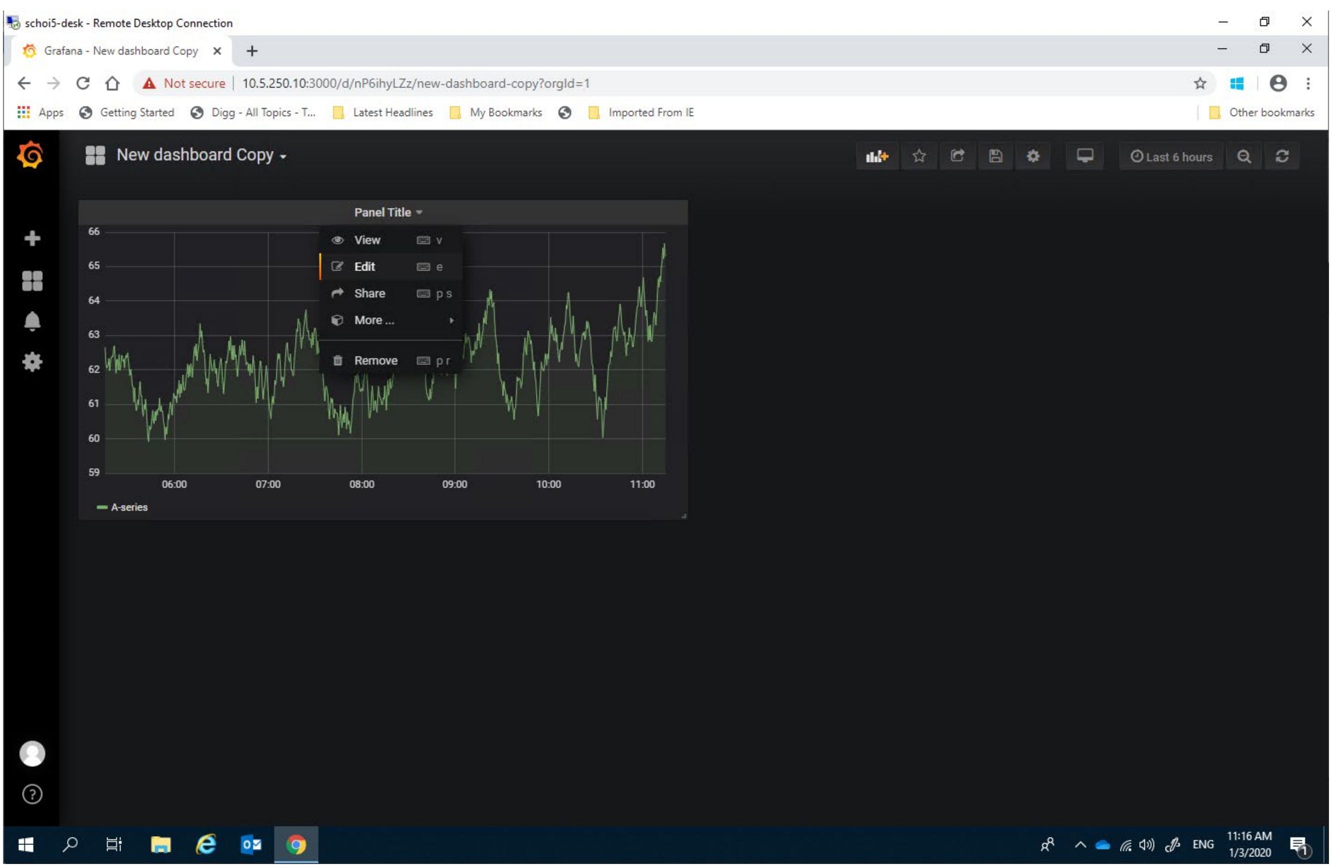


6. In the new dashboard, select the option "Add" and choose "Graph" as the new panel.

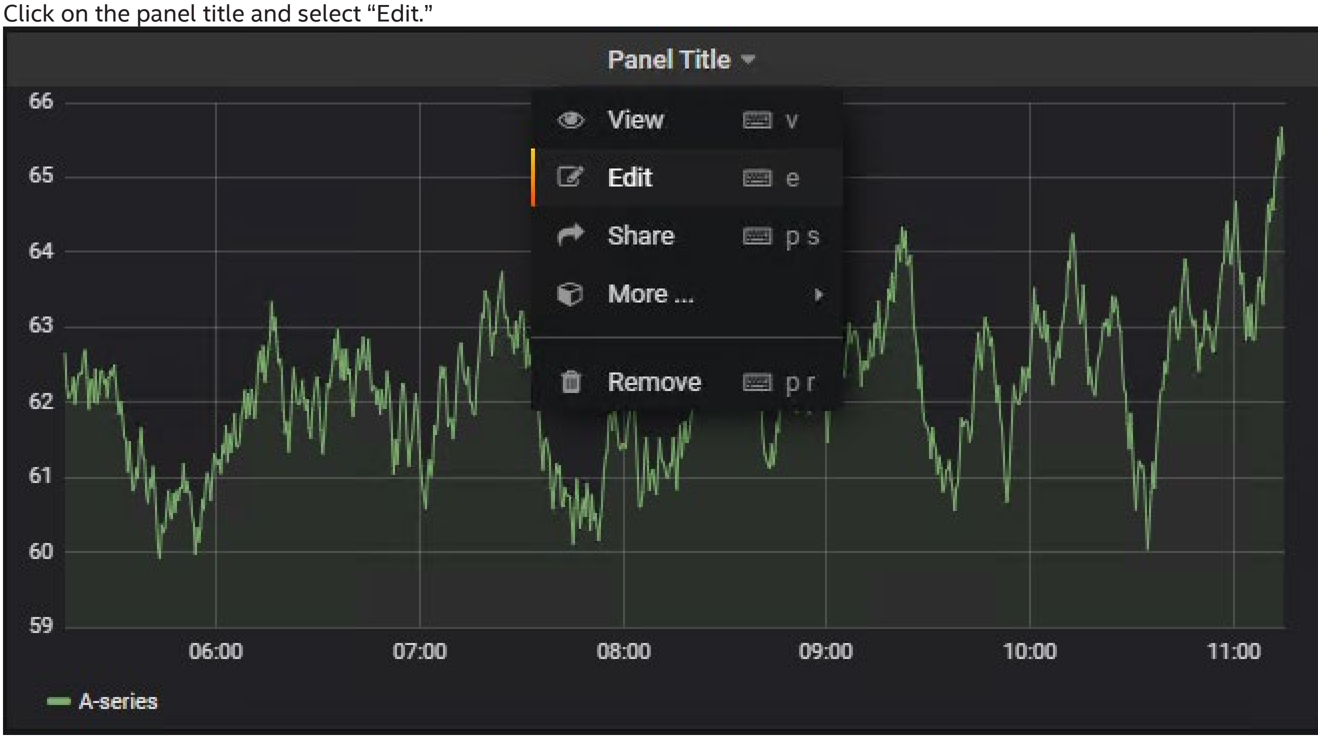


7. After doing this, a mock-up graph panel shows up.

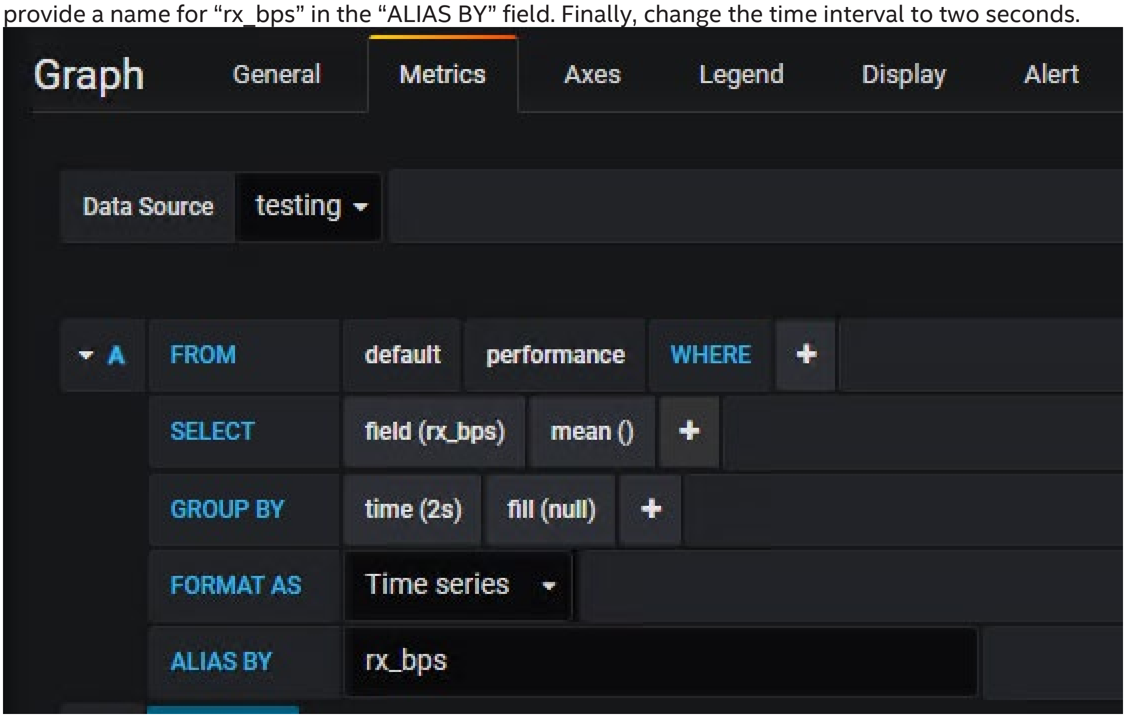




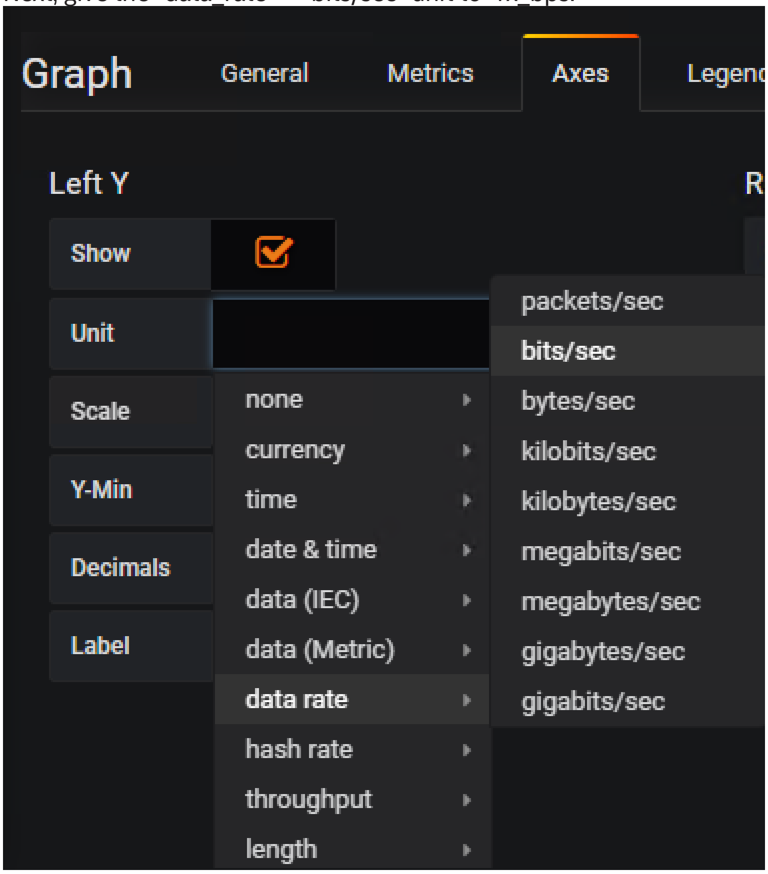
8. Click on the panel title and select "Edit."



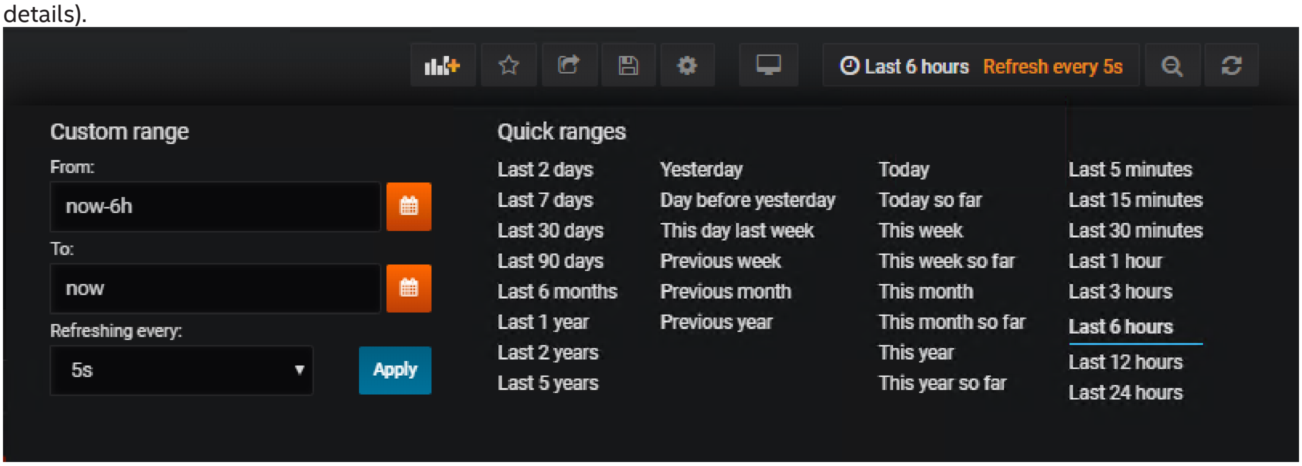
9. Update the graph to read the InfluxDB database by setting the following values: in "Data Source," choose "testing;" in "FROM," set the measurement to "performance;" and in "SELECT," enter the data point "rx_bps." The user can also provide a name for "rx_bps" in the "ALIAS BY" field. Finally, change the time interval to two seconds.



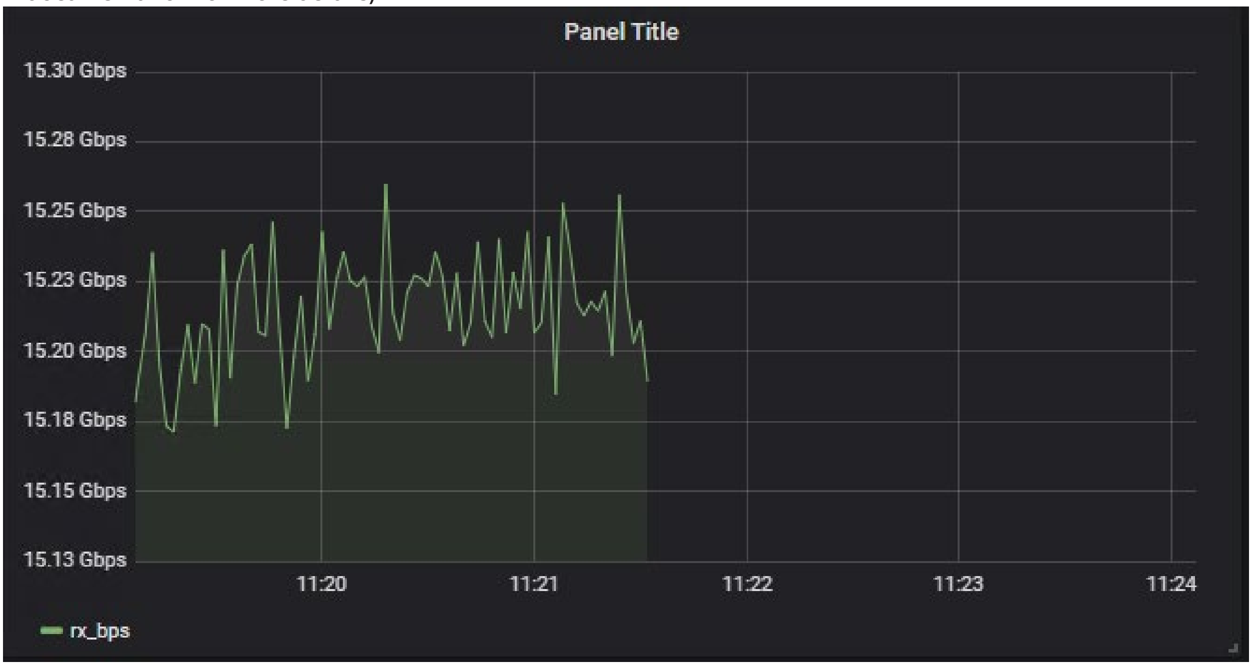
10. Next, give the “data_rate” > “bits/sec” unit to “rx_bps.”



11. At the upper right corner, the user can change the time frame and the refresh rate of Grafana. Finally, the graph should look as in the figure below; the user is allowed to create other panels (follow Grafana’s documentation for more details).



12. Finally, the graph should look as in the figure below; the user is allowed to create other panels (follow Grafana's documentation for more details).



3 Platform Specifications

The following tables list the hardware and software components used by the Network Function Virtualization (NFV) system and by the demonstration system.

Table 3-1 Hardware Ingredients for the NFV System Used in the Performance Tests

Item	Description	Notes
Platform	Intel Server Board S2600WFQ	Intel Xeon processor-based dual- processor server board with 2 x 10 GbE integrated LAN ports
Processor	2 x Intel Xeon Gold Processor	At least 10 cores are require, with 2 processors and hyperthread, 20 cores with 40 threads
Memory	192GB Total; Micron* MTA36ASF2G72PZ	12x16GB DDR4 2133MHz 16GB per channel, 6 Channels per socket
NIC	3 x Intel Ethernet Network Adapter XXV710-DA2 (2x25G)	6 x 1/10/25 GbE ports, only 4 will be use. Firmware version 5.50
Storage	Intel DC P3700 SSDPE2MD800G4	SSDPE2MD800G4 800 GB SSD 2.5in NVMe/PCIe
BIOS	Intel Corporation SE5C620.86B.0X.01.0007.060920171037 Release Date: 06/09/2017	Hyper-Threading - Enable Boot performance Mode – Max Performance Energy Efficient Turbo – Disabled Turbo Mode - Disabled C State - Disabled P State - Disabled Intel VT-x Enabled Intel VT-d Enabled

Table 3-2 Software Ingredients for the NFV System Used in the Performance Tests

Software Component	Description	References
Host Operating System	Ubuntu 18.04 x86_64 (Server) Fedora 29	https://www.ubuntu.com/download/server https://getfedora.org/en/server/download/
DPDK	dpdk-stable-17.11.4	https://fast.dpdk.org/rel/dpdk-17.11.4.tar.xz
OpenvSwitch	openvswitch-2.10.1	https://www.openvswitch.org/releases/openvswitch-2.10.1.tar.gz
Qemu	qemu-2.12.1	https://download.qemu.org/qemu-2.12.1.tar.xz
Trex	V2.35	https://github.com/cisco-system-traffic-generator/trex-core/releases/tag/v2.35

Table 3-3 Hardware Ingredients for the Demonstration System Used in the Performance Tests

Item	Description	Notes
Platform	INTEL® NUC KIT NUC6i7KYK	Intel® NUC 7 Mini PC
Processor	Core i7 6770HQ Skylake	Base Frequency 2.60Ghz, 4 cores 8 threads
Memory	16 GB	2x8GB DDR4 2133MHz 8GB per channel, 2 Channels
NIC	Intel® Ethernet Connection I219-LM	10/100/1G Ethernet
Storage	M.2 SSD	Intel SSDSCKKW 512GB SSD 2.5in

Table 3-4 Software Ingredients for the Demonstration System Used in the Performance Tests

Software Component	Description	References
Host Operating System	Linux OS Distribution: Fedora or Ubuntu Kernel: 4.4.0-62-generic	https://www.ubuntu.com/download/server
Database	InfluxDB	From OS repository
Data Visualization	Grafana	https://dl.grafana.com/oss/release/grafana_5.4.2_amd64.deb

4 Appendix A: Abbreviations

Abbreviation	Description
CPU	Central Processing Unit
DPDK	Data Plane Development Kit
DUT	Device Under Test
KPI	Key Performance Index
NFV	Network Functions Virtualization
NUC	Next Unit Computing
OVS	OpenvSwitch
PMD	DPDK Poll Mode Driver
SKU	Stock Keeping Unit
SLA	Service Level Agreement
SUT	System Under Test
VM	Virtual Machine
VNF	Virtual Network Function

5 Appendix B: Reference Documents

#	Title	Reference
1	Intel Ethernet Converged Network Adapter X710-DA2	http://ark.intel.com/products/83964/Intel-Ethernet-Converged-Network-Adapter-X710-DA2
2	RFC 2544 Benchmarking Methodology	https://tools.ietf.org/html/rfc2544
3	TRex	https://trex-tgn.cisco.com/
4	InfluxDB	https://www.influxdata.com/
5	Grafana	https://grafana.com/
6	ETSI	https://www.etsi.org/technologies/nfv/nfv

6 Legal Information

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifi cations. Current characterized errata are available on request. Contact your local Intel sales once or your distributor to obtain the latest specifications and before placing your product order.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer. Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

All products, computer systems, dates and gestures specified are preliminary based on current expectations, and are subject to change without notice. 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.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

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

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Intel, the Intel logo, Intel vPro, 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.

Printed in USA Please Recycle 0319/SJC//PDF 342792 -001US