1.0 Executive Summary

Based ETSI NFV Reference Architecture can consist of the following functional blocks:

- Element Management (EM);
- Virtualized Network Function (VNF);
- Service, VNF and Infrastructure Description;
- VNF Manager(s);
- NFV Orchestrator;
- Virtualized Infrastructure Manager(s) (VIM);
- NFV Infrastructure (NFVI), including hardware and virtual compute, storage and network resources.

Various ingredients, hardware and software components are required to be integrated to create an NFV system.

Thus, creating a high complexity barrier for customer as specially ODM who are focusing on hardware design only, and not able to demonstrate their boards full capabilities beside just function as a traditional server.

Here are a list of difficulty faces for Intel’s customers:

- Setup/integration of an NFV system.
- Create relevant workload/use case.
- Understanding of Intel technology.
- Unable extract workload KPI to show its Intel’s platform’s benefit.
- Require long turnaround optimize the system, no experience to tune system performance for a network transformation.

This is where NFV Demonstration framework will provide a solution to solve these problems. This document outlines an NFV system is setup, and KPI data are being collected/recorded with various permutation to Intel Technologies. providing data collection methodology to record KPI data collected from NFV system. Finally, the demo framework allow you save the collected data and play it back later for demonstration, present an intuitive a data visualization environment for user to understand the performance benefit of Intel processor, 24/7 anytime, anywhere without a full investment of expensive traffic generator and full hardware of NFV system.
Highlights from this document include:

- A guide to setup an NFV system, with key elements such as OpenvSwitch and a VNF.
  - Host system with Linux OS
  - OpenvSwitch
  - VM: ubuntu 16.04
  - VNF: Testpmd

- The KPI used is the dataplane performance of physical to virtual back to physical topology. The traffic will flow from a physical port to a virtual port on the Virtual Machine (VM), and then back to the physical port which can measure the NFV system, virtual switch and VM VNF dataplane performance in one go.

- The demo framework will provide:
  - The methodology to collect rate from traffic (our data point) to/from the traffic generator to be use for future playback
  - Recording of the console terminal, saving it for future playback to simulate actual running.
  - Playback 2 things at once, the console terminal, and the collect data point, and update a database for data visualization tool to present the information to the user, simulating actual execution and running of the NFV system.

- The following elements is part of the demo framework
  - A tool to record console output, and then play it back later, asciinema
  - A tool to generate traffic and measuring the NFV KPI, T-Rex
  - A tool to store data point of the NFV KPI, InfluxDB
  - A tool to display the changes in performance before/after applying Intel Technology.
2.0 Introduction

Intel NFV Demonstration Framework is designed to show playback pre recorded terminal console, and datapoint in to a data visualization GUI. The objective is to allow user to place NFV KPI into a simple file format, later using python upload the data into an open source time series database, which is then pick up by a data visualizing front end.

Below is the architecture:

Let look at how to present the pre-recorded data to the user.

Assuming the following sample data points are a viable, e.g. this is a snippet of a total of 5 minutes of data collected at 1 Hz frequency.

<table>
<thead>
<tr>
<th>TX bps</th>
<th>RX bps</th>
<th>TX pps</th>
<th>RX pps</th>
</tr>
</thead>
<tbody>
<tr>
<td>3349958841.0</td>
<td>24600565248.0</td>
<td>49850578.0</td>
<td>36607984.0</td>
</tr>
<tr>
<td>3351246400.0</td>
<td>24606240288.0</td>
<td>49927450.0</td>
<td>36616429.0</td>
</tr>
<tr>
<td>33651676128.0</td>
<td>24604982976.0</td>
<td>50076899.0</td>
<td>36614558.0</td>
</tr>
<tr>
<td>33579966336.0</td>
<td>24619423584.0</td>
<td>49970188.0</td>
<td>36636047.0</td>
</tr>
<tr>
<td>33529855296.0</td>
<td>24634791552.0</td>
<td>49895618.0</td>
<td>36658916.0</td>
</tr>
<tr>
<td>33513466560.0</td>
<td>24630570720.0</td>
<td>49871230.0</td>
<td>36652635.0</td>
</tr>
</tbody>
</table>

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

```python
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
while (infinite):
    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
```
And then by configuring Grafana’s Dashboard and Panel, you can create a real time graph of data rate forwarded by an NFV system.

The remaining of the document will guide you through the entire process from creating a NFV system to creating an offline demonstration.

2.1 Building an NFV system

An NFV system as defined by ETSI are specify by the diagram below:-
An simplify NFV test environment used by this document specified by the diagram below:-

This simplest form of NFV environment consist of:-

1) First is to prepare a Linux system
2) Host: Linux system ubuntu or Fedora
   b.  https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0
Preparation of Linux Environment

For RHEL/Fedora systems these can be installed required software packages using

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

For Ubuntu/Debian systems these can be installed using

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

Download software

<table>
<thead>
<tr>
<th>Software Needed for NFV Environment</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPDK</td>
<td>dpdk-stable-17.11.4</td>
</tr>
<tr>
<td>OpenvSwitch</td>
<td>openvswitch-2.10.1</td>
</tr>
<tr>
<td>Qemu</td>
<td>qemu-2.12.1</td>
</tr>
<tr>
<td>Trex</td>
<td>Trex v2.35</td>
</tr>
</tbody>
</table>

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 libglvnd2.0-dev libfdt-dev libpixman1-dev zlib1g-dev`
- `./configure --target-list=x86_64-softmmu`
- `make -j10`

2.2 Performance Test Scenarios

The NFV KPI performance test scenarios were designed in order to demonstrate the data plane forwarding capability of the host virtual switch moving data packet from the host physical ports, to the VM. The VM will also be running a simplify VNF that is forwarding data packets from virtio interface from/back to the virtual switch. The data path of this test is physical port to vswitch, then virtio in VM, forwarded by VNF to another virtio interface, vswitch will forward virtio to physical port to completed the route.

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

- `testpmd DPDK user-mode application. DPDK is a set of libraries providing a programming framework to enable high-speed data packet networking applications. Applications using DPDK libraries and interfaces run in user mode and directly interface with NIC functions, skipping slow, kernel layer components to boost packet processing performance and throughput. These applications process raw network packets without relying on protocol stack functionality provided by kernel. For more information on DPDK go to http://www.dpdk.org.`
openvswitch is an open-source implementation of a distributed virtual multilayer switch. The main purpose of Open vSwitch 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 improved performance in comparison with kernel-based dataplane.

Once all required software components are compiled, it is time to launch the apps.

Setting/Preparation for OpenvSwitch:

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

Setting up, first initialize new OpenvSwitch database

Before we can start the OvS daemon "ovs-vswitchd", we need to initialize the OvS DB and start ovsdb-server. The following commands show how to clear/create a new OvS DB and 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

Starting OpenvSwitch portion using 1GB:-

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"

The OpenvSwitch log file is located 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

The following steps will create a bridge with 2 physical ports and 2 vhost backend to support 2 virtio interface in the VM.

./utilities/ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev
ifconfig br0 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
After OpenvSwith is running, we can start VM:

```
./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
```

After OpenvSwith is running, we can start 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, ifname=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,
```

With VM powered on, we can start testpmd to forward packets between 2 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/ustertools/dpdk-devbind.py -b igb_uio 00:04.0
$DPDK_DIR/ustertools/dpdk-devbind.py -b igb_uio 00:05.0
$DPDK_DIR/x86_64-native-linuxapp-gcc/app/testpmd -c 0x6 -n 4 --txd=2048 --rxd=2048 --txqflags=0xf00 --disable-hw-vlan
```

With the final steps a NFV system with a simple VNF is complete. But we have created a script to automatic all these steps to get every component up and running.

### 2.3 Traffic Generator

In this configuration, we will setup the traffic generator running on another socket cpu so that it wouldn't interfere with NFV system and VNF running across the UPI on the opposing socket. To get our traffic generator T-Rex up and running, follow the instruction here:

```
wget -n -c http://trex-tgn.cisco.com/trex/release/v2.35.tar.gz
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
```

With VM powered on, we can start testpmd to forward packets between 2 virtio interfaces:
Another key component to this framework is we utilize a traffic generator with python api which is convenient for controlling the traffic and packets rates can be collected from the traffic generator at a fixed interval. By executing from steps A to H, traffic will be flowing across the virtual switch to VNF in the VM, and backout through another port.
2.4 Data Collection

There are 2 parts in the data collection to complete the framework:

1) Screen capture of execution of traffic generator and VM’s VNF
2) Data need to be collected here are packets per second to measure the data forwarding rate of NFV system and VNF.

Diagram after integration of data collection components:

We are using InfluxDB to store our NFV KPI. InfluxDB is the database and storage engine purpose-built to handle time series data. A perfect metric store for multiple data sources to help you avoid a siloed approach.

```
Fedora 29

--- Intel NFV 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
```

Fedora 29

```
Fedora 29

--- Intel NFV 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
```
dnf -y install InfluxDB

Ubuntu:

curl -sL https://repos.influxdata.com/InfluxDB.key | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/InfluxDB.list
sudo apt-get update && sudo apt-get install InfluxDB

to display the data points we can use Grafana, it is a open source analytics and monitoring solution for every database, Grafana allows you to query, visualize, alert on and understand our NFV KPI metrics no matter where they are stored.

To install Grafana

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

we create a console providing the live data feed of core utilization, traffic generator and testpmd's packet rates

Main control windows
Application Note | NFV Demonstration Framework

--- Intel NFV Enabling Kit ---

(a) Select Cores
(b) Select NICs
(c) Start OVS
(d) Create FVF
(e) Start VM
(f) Start VM fwd
(g) Start Trex backend
(h) Start Traffic
(i) Start data collection
(j) Add full cores traffic

---

(r) Set max to Turbo (POn)
(s) Set SST Prioritized Base Frequency
(t) Set max to Base (Pl)
(u) Absolute Minimum Frequency
(v) Set SST Core Power Frequency
(p) Print Core Frequency info

[?] Show Help Text
(q) Exit Script

---

WARNING, Using core list as defined in
an array embedded in this script
Not to be used with PBF enabled BIOS!

---

Option: 

Trex statistic windows:

```
-Per port stats table
  ports | 0 | 1

|         | 733062437422 | 733052438089
| packets | 1876639033576704 | 187663903357669
| bytes   | 224868023114 | 224868026533
| packets | 575662136084002 | 57566214481094
| errors  | 0 | 0
| errors  | 0 | 0
| Rx Bw   | 23.08 Gbps | 23.08 Gbps

-Global stats enabled
  Cpu Utilization : 8.5 % 135.0 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 : 68799.4 sec
test duration : 0.0 sec
```
In term of screen capture, we use Acinema a tools to record and share your terminal sessions, it is a lightweight, purely text-based approach to terminal recording, this allow our console execution to be playback as if it was the real execution of the application.

The method is to split the recording into 2 parts, start by recording setup until traffic start flowing, then we can stop the recording here.
Then the 2nd part, we will record the console screen of continuous flowing of packets across the openvswitch and VM's testpmd for a fix duration for example 5 minutes.

We should also collect the same amount of data points to match the screen recording e.g. of 5 minutes. Start collecting NFV KPI data, e.g. packet per second, for every second interval, at 1Hz frequency.

Ensure that trex backend is running in the background, to create traffic streams, we uses Trex's python api to configure and start transmission of the traffic.

```python
# 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)
```

Sample code to collect statistic from Trex and save it as data point:

```python
# 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 = float(fs["flow_stats'][fs_pgids[1]]['tx_pps']["total"]
    rx_pps = float(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)
    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) + '

# write data point into a file.
f = open(path_download+"/result.txt", "w")
f.write(data_points)
f.close()
```
Assuming the following sample data points are viable, e.g. this is a snippet of a total of 5 minutes of data collected at 1 Hz frequency.

<table>
<thead>
<tr>
<th>TX bps</th>
<th>RX bps</th>
<th>TX pps</th>
<th>RX pps</th>
</tr>
</thead>
<tbody>
<tr>
<td>33499588416.0</td>
<td>24600565248.0</td>
<td>49850578.0</td>
<td>36607984.0</td>
</tr>
<tr>
<td>3355124600.0</td>
<td>24606202488.0</td>
<td>49927450.0</td>
<td>36616429.0</td>
</tr>
<tr>
<td>33651676128.0</td>
<td>24604982976.0</td>
<td>50076899.0</td>
<td>36614558.0</td>
</tr>
<tr>
<td>33579966366.0</td>
<td>24619243584.0</td>
<td>49970188.0</td>
<td>36636047.0</td>
</tr>
<tr>
<td>3352985296.0</td>
<td>24634791552.0</td>
<td>49895618.0</td>
<td>36658916.0</td>
</tr>
<tr>
<td>33513466560.0</td>
<td>24630570720.0</td>
<td>49871230.0</td>
<td>36652635.0</td>
</tr>
</tbody>
</table>

To playback screen playback:
```
os.system("asciinema play -s 1 /<your_recording_path>/setup.cast")
while infinite:
    os.system('clear')
    os.system("asciinema play -s 1 /<your_recording_path>/traffic.cast")
time.sleep(1)
```

To playback data point:
```
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
while (infinite):
    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 you can combine the playback screen recording, and data point update to database by creating a function based on the above code e.g. 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
```
2.5 Grafana for Data Visualization

Grafana is used to query, visualize, data points store in InfluxDB, and with build in function we will create a time based graph visualize the in coming NFV KPI as it automatically query new entries in InfluxDB.

The following step guide you to create a graph to show the latest performance of the NFV system.

First access Grafana via browser [http://<your system ip address>:3000](http://<your system ip address>:3000)

First time access to the system, user is require to add a data source, in this case our InfluxDB.
User will need to update the URL of InfluxDB to allow Grafana to connect to it.

In our case here:-

The URL is [http://localhost:8086](http://localhost:8086) because InfluxDB and Grafana are both reside on the same system, and the name of your database.
Press Save and Test, make sure that Grafana will prompt back as Data source is working
Click at the “+” and create a new Dashboard

At the new dashboard, select add a graph as new panel
Instantly you are provided with a mock up graph panel.
Click on the panel title, then select Edit.

Update the graph to read InfluxDB database “testing”, the measurement as “performance” and finally the datapoint “rx_bps”, you can also provide a name for “rx_bps” at the Alias field. Change time interval to 2s.
Now give a unit to “rx_bps”, as data_rate, bits/sec

Finally the graph will look like this, and users are allowed to create other panels, please follow Grafana documentation for more detail setting:-
At the upper right corner, user can change the time frame and the refresh rate of Grafana.
4.0 Platform Specifications
Table 4-1, Table 4-2, Table 4-3, Table 4-4 list the hardware and software components used by the NFV System and Demonstration System.

4.1 Hardware ingredients for NFV System
Table 4-1 Hardware ingredients used in performance tests

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

4.2 Software ingredients for NFV System
Table 4-2 Software ingredients used in performance tests

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
</table>
| Host Operating System | Ubuntu 18.04 x86_64 (Server) Fedora 29          | https://www.ubuntu.com/download/server
| 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 |
4.3 Hardware ingredients for Demonstration System

Table 4-3 Hardware ingredients used in performance tests

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

4.4 Software ingredients for Demonstration System

Table 4-3 Software ingredients used in performance tests

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Operating System</td>
<td>Linux OS Distribution: Fedora or Ubuntu Kernel: 4.4.0-62-generic</td>
<td><a href="https://www.ubuntu.com/download/server">https://www.ubuntu.com/download/server</a></td>
</tr>
<tr>
<td>Database</td>
<td>InfluxDB</td>
<td>From OS repository</td>
</tr>
<tr>
<td>Data Visualization</td>
<td>Grafana</td>
<td><a href="https://dl.grafana.com/oss/release/grafana_5_4.2_amd64.deb">https://dl.grafana.com/oss/release/grafana_5_4.2_amd64.deb</a></td>
</tr>
</tbody>
</table>
5.0 Summary

The NFV Demonstration framework achieve 2 primary functionalities, by combining console execution of NFV system with data visualization using InfluxDB and Grafana.

Firstly, it allows user to use this setup to create an NFV Demonstration without require any physical equipment's. This demonstration setup only requires a NUC mini pc to run the demo, do not need any cumbersome and costly duplication of equipment just to playback a demo to customer.

Secondly, this framework can be used as a learning platform, where by the screen recording can be published, allowing anyone who is interested to bring up an NFV system can follow the recording, because it is more intuitive than a document based user guide.

To access more information that is part of the Intel Container Experience Kits (user guides, application notes, feature briefs and other collateral) go to: https://networkbuilders.intel.com/network-technologies/container-experience-kits.
## Appendix A: Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DPDK</td>
<td>Data Plane Development Kit</td>
</tr>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Index</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Functions Virtualization</td>
</tr>
<tr>
<td>NUC</td>
<td>Next Unit Computing</td>
</tr>
<tr>
<td>OVS</td>
<td>OpenvSwitch</td>
</tr>
<tr>
<td>PMD</td>
<td>DPDK Poll Mode Driver</td>
</tr>
<tr>
<td>SKU</td>
<td>Stock Keeping Unit</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtual Network Function</td>
</tr>
</tbody>
</table>
## Appendix B: Reference Documents

<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>TRex</td>
<td><a href="https://trex-tgn.cisco.com/">https://trex-tgn.cisco.com/</a></td>
</tr>
<tr>
<td>4</td>
<td>InfluxDB</td>
<td><a href="https://www.influxdata.com/">https://www.influxdata.com/</a></td>
</tr>
<tr>
<td>5</td>
<td>Grafana</td>
<td><a href="https://grafana.com/">https://grafana.com/</a></td>
</tr>
<tr>
<td>6</td>
<td>ETSI</td>
<td><a href="https://www.etsi.org/technologies/nfv/nfv">https://www.etsi.org/technologies/nfv/nfv</a></td>
</tr>
</tbody>
</table>
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 specifications. 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.

© 2018 Intel Corporation. All rights reserved.