EPA Hands-On Lab #3

Version 1.0

Date 30th Nov 2018

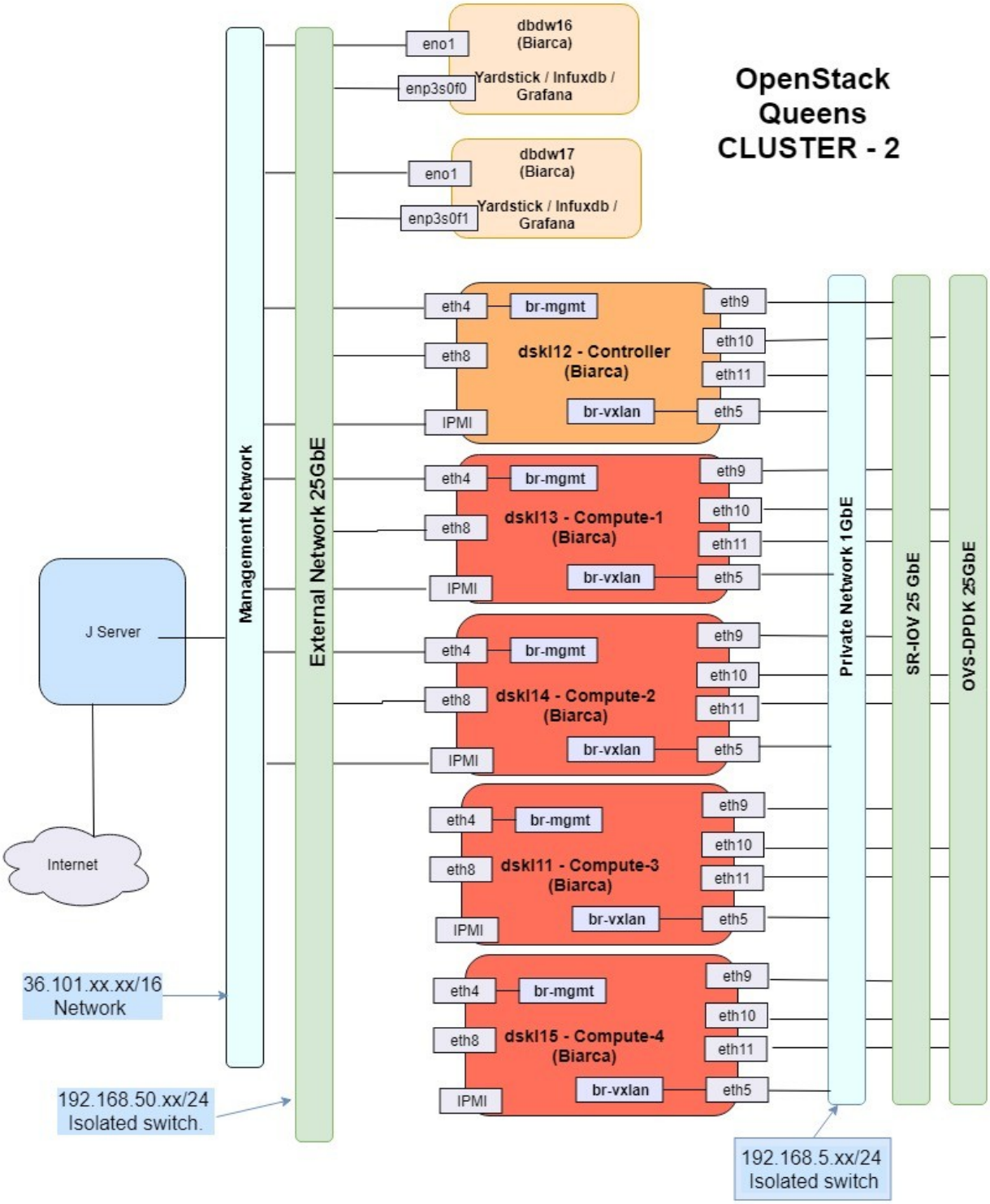
1 Objective

Using an OpenStack Queens based Multi-Node cluster, help get an understanding of Data Plane Development Kit (DPDK) as a simple option for network performance enhancement. Plan is to use iperf3 as a tool of choice running on Ubuntu 16.04 Instances using DPDK based network ports and validate East-West communication. Throughput will be measured from a CPUs pinned instance and an instance whose CPUs are shared.

2 Prerequisites

- Networking fundamentals
- Basic knowledge of DPDK and familiarity with OpenStack environment
- Hands-On EPA Lab – Introduction
- Overview of current Infrastructure
- Host to be used for connecting to DPDK capable OpenStack Instances
- Infra-access details available (**Did you receive a note/slip? - If NO! - please STOP here and ASK!!!**)
    - Credentials to login into jumpserer
    - Host Access details
    - OpenStack Dashboard Credentials
    - DPDK Capable Instances details for East-West
        - External/Floating IP
        - Guest Net IP
        - DPDK Network IP - East - West
        - SSH Key pair Host Info
        - User Name: ‘ubuntu’
    - DPDK Capable Compute Nodes Availability Zone
- Commands in BLUE
- Dark Orange 2 text needs to be **updated before you attempt the** command
- Dark Magenta - “Cambria” font - **sample/console output - DO NOT USE for COPY/PASTE**
- Highlighted information has yellow background

3 Network Diagram



4 DPDK Instance(s) Basic Details

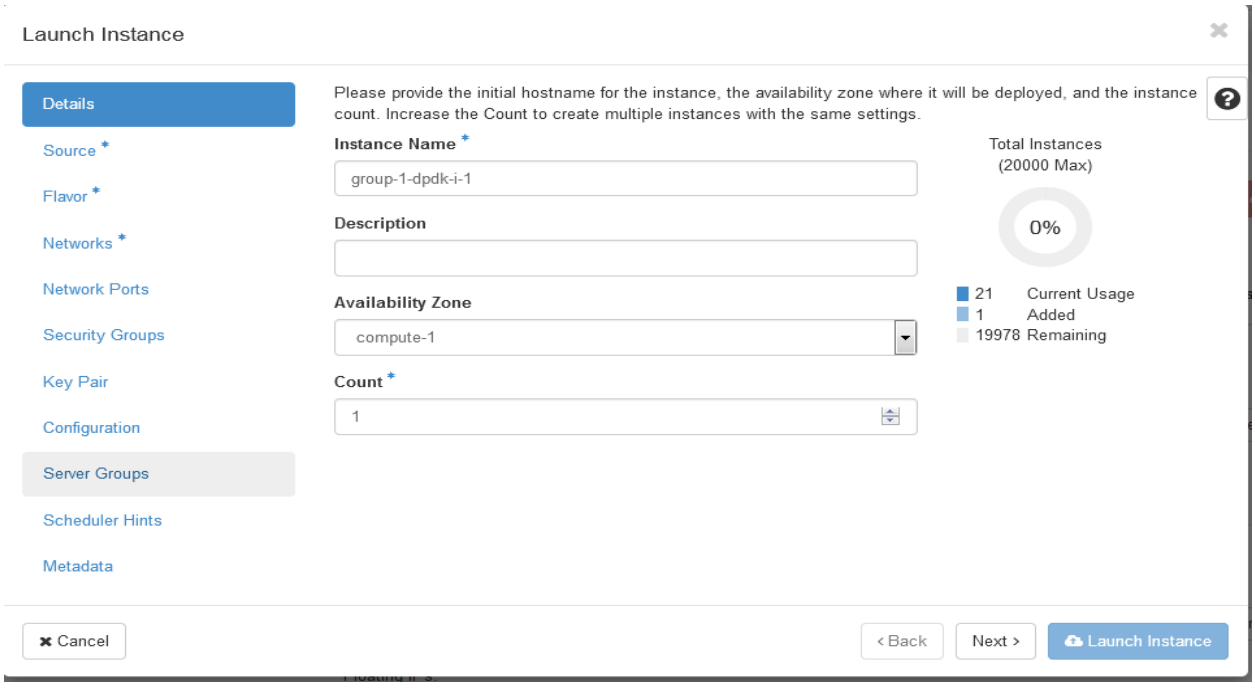
- Per Instance info
  - 2vCPUs
  - 4 GB RAM
  - 10 GB Disk
  - Ubuntu 16.04
  - Availability zones used “compute-1” and “compute-2” - Note Availability zones will be specific to user
  - Key pair as needed for SSH connection from host
- 2-3 Instances per user/group
- Additional details on DPDK are captured DPDK Information

5 DPDK Capable Instance creation from OpenStack Dashboard (Pre-configured to skip)

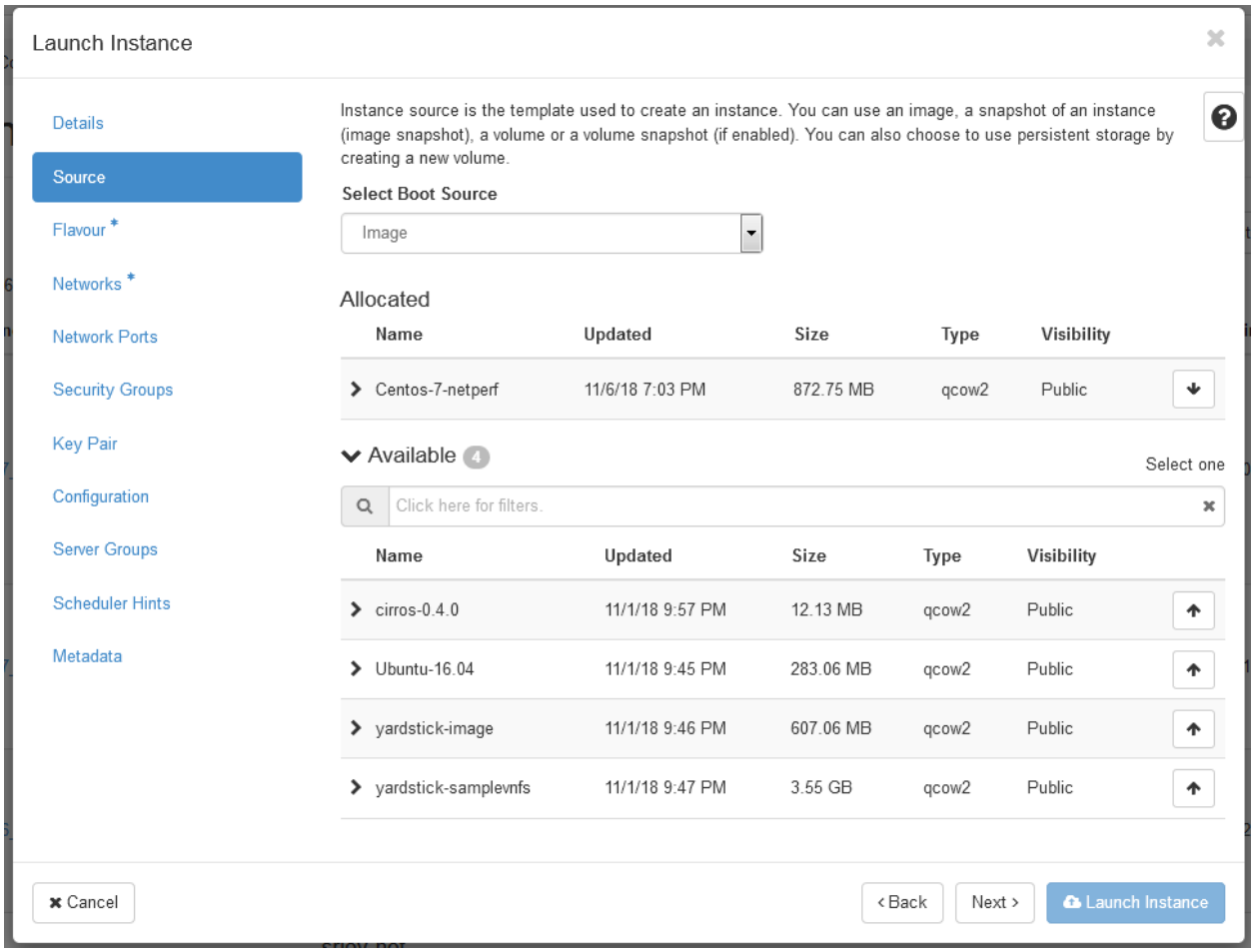
1. Login to OpenStack Dashboard using the link and access details provided to you

5.1 Instance Creation with DPDK Network

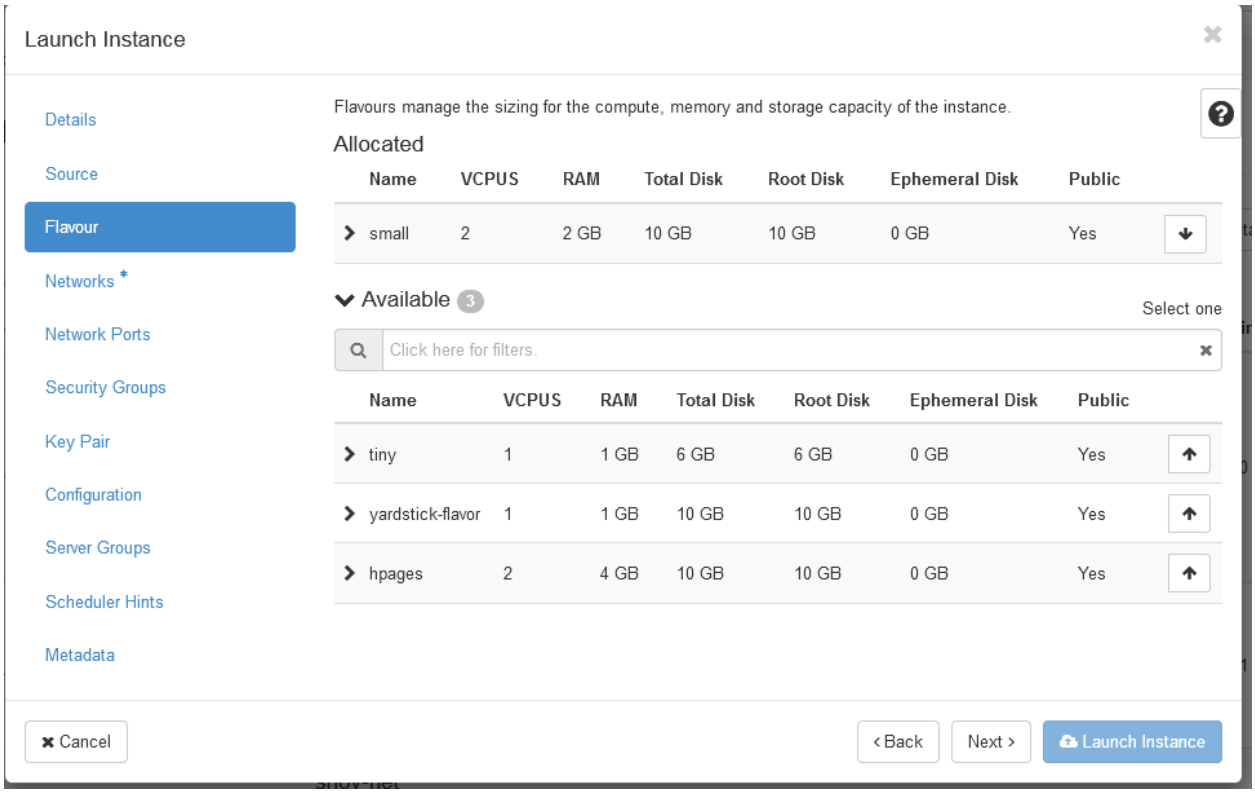
1. Go the Project → compute → instances. Click on new instance. Enter the name and select the availability zone.



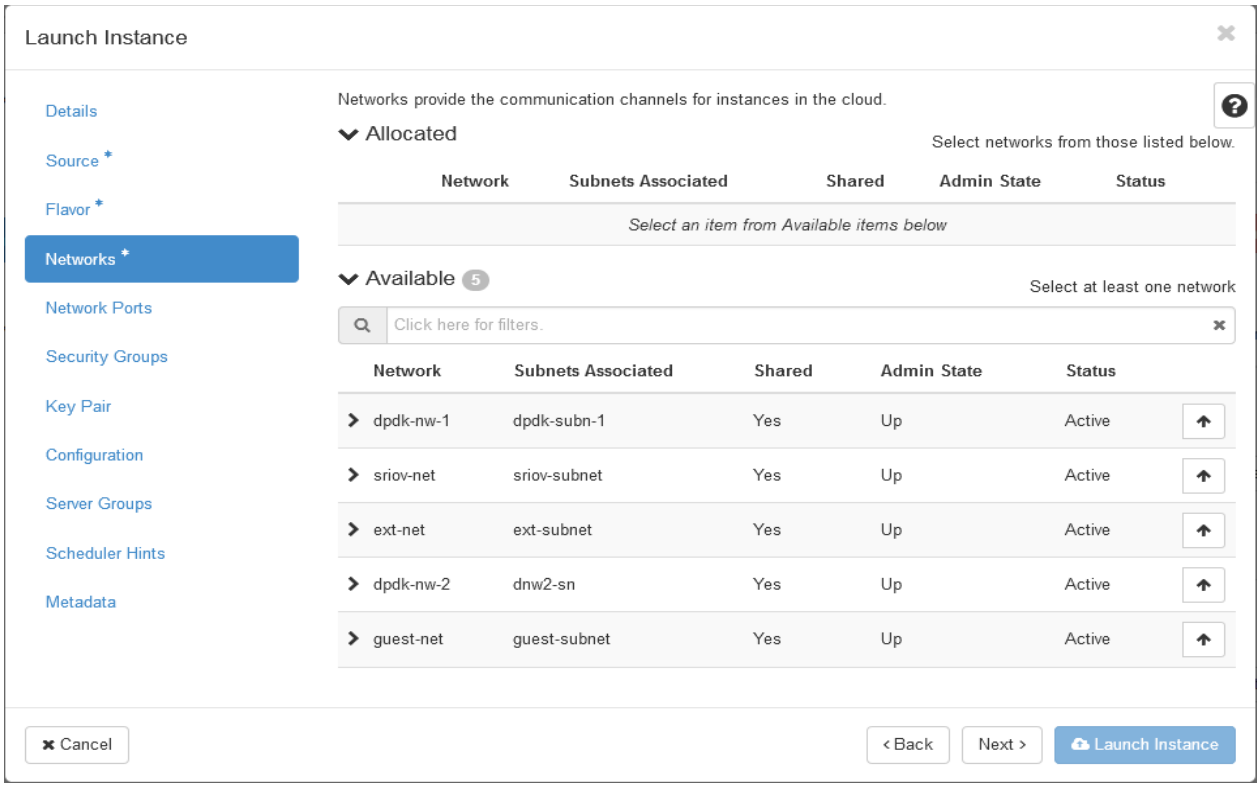
2. Select **yardstick-samplevnfs** image for your instance.



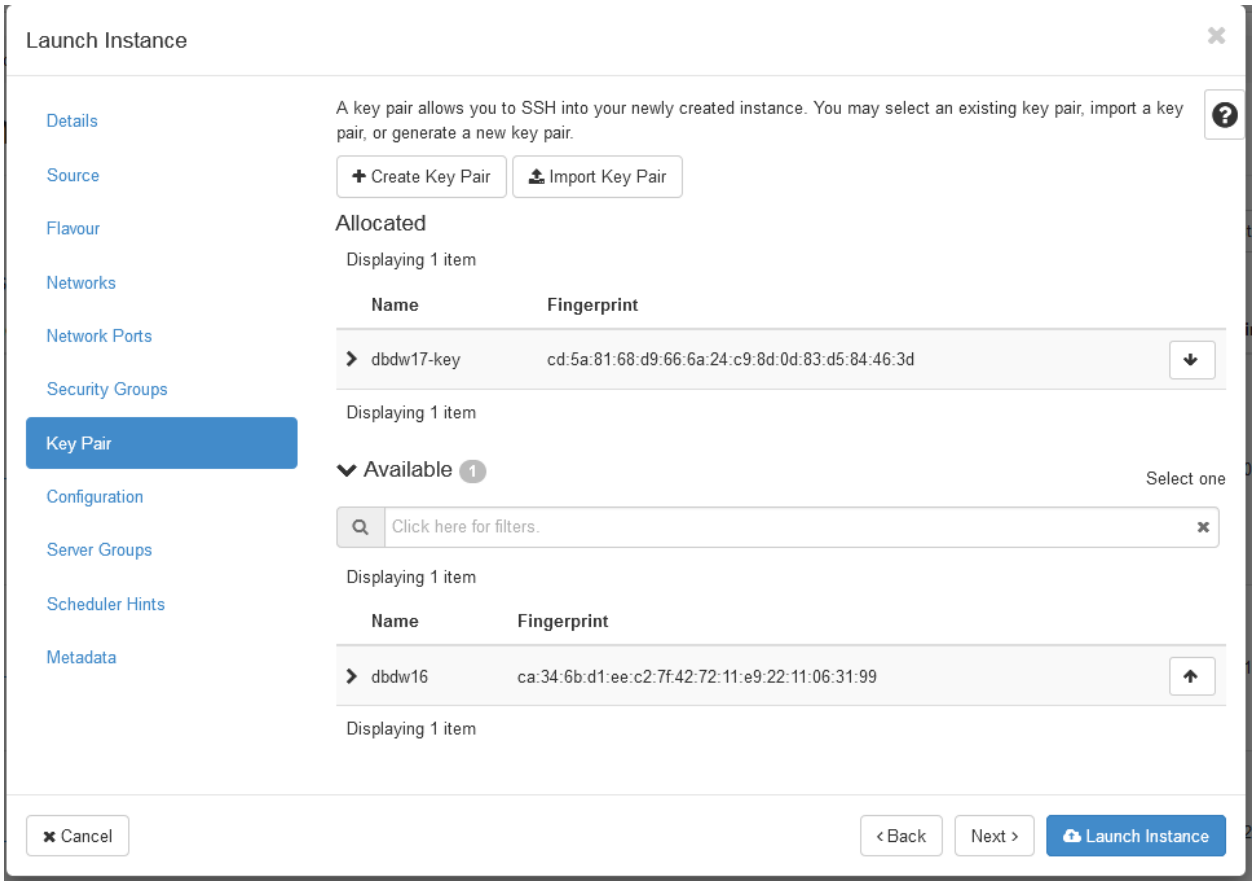
3. Select the **hpages** flavor



4. Select the **guest-net** as first network and **dpdk-nw-1.**



5. Select the key pair, this very important to ssh login to the instance using floating ip. Just click on launch instance once done.



5.2 Verification Post DPDK Instance Creation

Follow the below steps to check if a DPDK instance is properly created.

1. ssh to the Openstack compute host on which DPDK instance is created.
2. Use **virsh list** to list all the instances on that host. Below is a sample.

```
root@dskl13:/home/ubuntu
# virsh list
 Id    Name
State
----------------------------------------
----------
 1     instance-000001ee
running
```

```
 2     instance-000001ef
running
 3     instance-000001f3
running
 4     instance-000001f4
running
```

3. Instance details can be identified using the command **virsh dumpxml <instance id>**
4. To the newly created DPDK instance, the below should be available.

```
<memoryBacking>
  <hugepages>
   <page size='1048576' unit='KiB' nodeset='0'/>
  </hugepages>
 </memoryBacking>

<vcpu placement='static'>2</vcpu>
 <cputune>
  <shares>2048</shares>
  <vcpupin vcpu='0' cpuset='83'/>
  <vcpupin vcpu='1' cpuset='39'/>
  <emulatorpin cpuset='39,83'/>
 </cputune>
 <numatune>
  <memory mode='strict' nodeset='1'/>
  <memnode cellid='0' mode='strict' nodeset='1'/>
 </numatune>

 <interface type='vhostuser'>
  <mac address='fa:16:3e:65:e9:00'/>
  <source type='unix' path='/var/run/openvswitch/vhu34b907e3-cd'
mode='server'/>
  <target dev='vhu34b907e3-cd'/>
  <model type='virtio'/>
  <alias name='net1'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04'
function='0x0'/>
 </interface>
```

- hugepages - indicate that large page size memory is assigned to the instance
- vcpupin - If dedicated CPUs are pinned to the instance, then as highlighted in the sample output above, single CPU is listed. If shared CPUs are configured, then it should look like <vcpupin vcpu='0' cpuset='26-43,66-67,70-87'/>
- numatune - Indicates on which NUMA node the instance is created on
- interface type "**vhostuser**" - Indicates that DPDK vhost user port is created in OVS and attached to the instance

6 DPDK Capable Instance Login

1.In Openstack Dashboard -> Project -> Compute -> Instances, find the floating ip of the instance along with the key-pair of the access host from which the instance can be accessed.
2. Have 3 terminal sessions and use each of the sessions to connect to the jump server and then to the access host in the previous step
3. From one session, "ssh" using the floating ip / ext-net IP to the instance from the access host. Please make to execute "sudo su" before connecting to instance.

```
root@<host-with-keypair>:# ssh ubuntu@<Instance external / floating IP>
```

```
root@<dbdw16>:# ssh ubuntu@192.168.50.94
```

Note: If you see WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! during login to the instance, run below command with the instance ip address.

```
root@<hostname>:# ssh-keygen -f "/root/.ssh/known_hosts" -R <Instance external IP>
```

3. Need to repeat the same steps for 2 other instances for 2 other terminal sessions

7 Assigning IP Address to the DPDK Ports in the Ubuntu Instance

1. Update the interfaces file with the below lines. Make sure to provide your dpdk network IP at the **address** field.

```
# vi /etc/network/interfaces
auto ens4
iface ens4 inet static
address <DPDK Network IP Address provided>
netmask 255.255.255.0
```

2. Restart the network service or instance.

```
#systemctl restart networking
```

3. Repeat the same steps for the other instance also.

8 Latency Test

8.1 East-West

Of the 3 Instances on the same compute node, 2 instances have dedicated CPUs and 1 instance has shared CPUs. First ping between the instances with dedicated CPUs.

1. From the terminal session of the selected instance, ping external IP of other instance

```
[ubuntu@<instance1-name>~]$ping <external ip of other instance>
```

```
root@dpdk-gr4-i2:/home/ubuntu# ping -c 10 192.168.50.56
PING 192.168.50.56 (192.168.50.56) 56(84) bytes of data.
64 bytes from 192.168.50.56: icmp_seq=1 ttl=64 time=0.340 ms
64 bytes from 192.168.50.56: icmp_seq=2 ttl=64 time=0.218 ms
64 bytes from 192.168.50.56: icmp_seq=3 ttl=64 time=0.218 ms
64 bytes from 192.168.50.56: icmp_seq=4 ttl=64 time=0.273 ms
```

```
64 bytes from 192.168.50.56: icmp_seq=5 ttl=64 time=0.273 ms
64 bytes from 192.168.50.56: icmp_seq=6 ttl=64 time=0.277 ms
64 bytes from 192.168.50.56: icmp_seq=7 ttl=64 time=0.293 ms
64 bytes from 192.168.50.56: icmp_seq=8 ttl=64 time=0.276 ms
64 bytes from 192.168.50.56: icmp_seq=9 ttl=64 time=0.369 ms
64 bytes from 192.168.50.56: icmp_seq=10 ttl=64 time=0.300 ms

--- 192.168.50.56 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.218/0.283/0.369/0.048 ms
```

2. From the terminal session of the selected instance, ping guest-net IP of other instance.

```
[ubuntu@<instance1-name>~]$ping <guest-net ip of other instance>
```

```
root@dpdk-gr4-i2:/home/ubuntu# ping -c 10 10.5.0.38
PING 10.5.0.38 (10.5.0.38) 56(84) bytes of data.
64 bytes from 10.5.0.38: icmp_seq=1 ttl=64 time=0.364 ms
64 bytes from 10.5.0.38: icmp_seq=2 ttl=64 time=0.322 ms
64 bytes from 10.5.0.38: icmp_seq=3 ttl=64 time=0.284 ms
64 bytes from 10.5.0.38: icmp_seq=4 ttl=64 time=0.234 ms
64 bytes from 10.5.0.38: icmp_seq=5 ttl=64 time=0.297 ms
64 bytes from 10.5.0.38: icmp_seq=6 ttl=64 time=0.233 ms
64 bytes from 10.5.0.38: icmp_seq=7 ttl=64 time=0.277 ms
64 bytes from 10.5.0.38: icmp_seq=8 ttl=64 time=0.279 ms
64 bytes from 10.5.0.38: icmp_seq=9 ttl=64 time=0.311 ms
64 bytes from 10.5.0.38: icmp_seq=10 ttl=64 time=0.265 ms

--- 10.5.0.38 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8998ms
rtt min/avg/max/mdev = 0.233/0.286/0.364/0.042 ms
```

3. From the terminal session of the selected instance, ping DPDK IP of other instance

```
[ubuntu@<instance1-name>~]$ping <DPDK ip of other instance>
```

```
root@dpdk-gr4-i2:/home/ubuntu# ping -c 10 12.0.0.13
PING 12.0.0.13 (12.0.0.13) 56(84) bytes of data.
64 bytes from 12.0.0.13: icmp_seq=1 ttl=64 time=0.452 ms
64 bytes from 12.0.0.13: icmp_seq=2 ttl=64 time=0.283 ms
64 bytes from 12.0.0.13: icmp_seq=3 ttl=64 time=0.245 ms
64 bytes from 12.0.0.13: icmp_seq=4 ttl=64 time=0.185 ms
64 bytes from 12.0.0.13: icmp_seq=5 ttl=64 time=0.170 ms
```

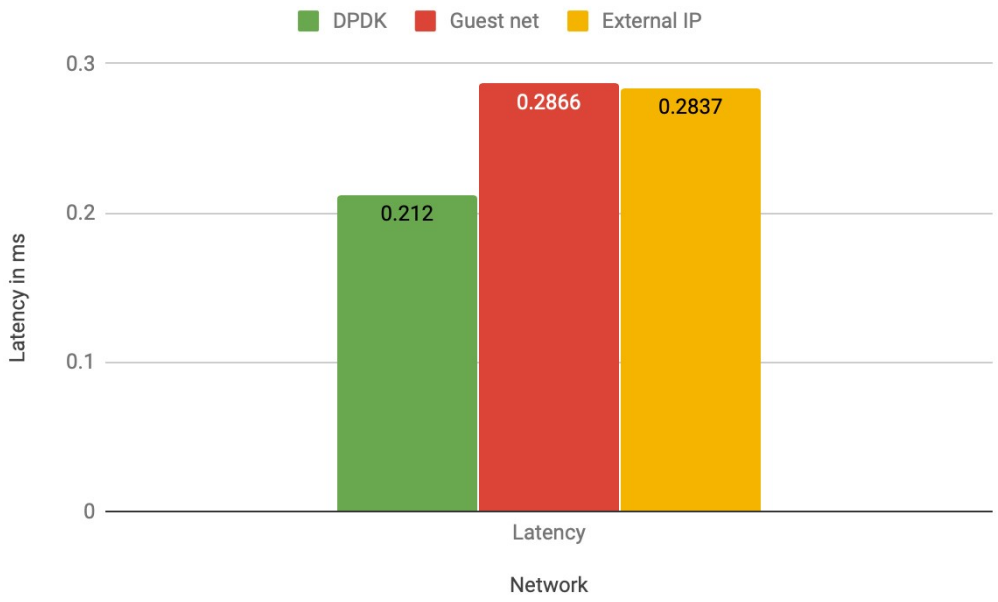
```
64 bytes from 12.0.0.13: icmp_seq=6 ttl=64 time=0.182 ms
64 bytes from 12.0.0.13: icmp_seq=7 ttl=64 time=0.125 ms
64 bytes from 12.0.0.13: icmp_seq=8 ttl=64 time=0.167 ms
64 bytes from 12.0.0.13: icmp_seq=9 ttl=64 time=0.173 ms
64 bytes from 12.0.0.13: icmp_seq=10 ttl=64 time=0.138 ms

--- 12.0.0.13 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8999ms
rtt min/avg/max/mdev = 0.125/0.212/0.452/0.091 ms
```

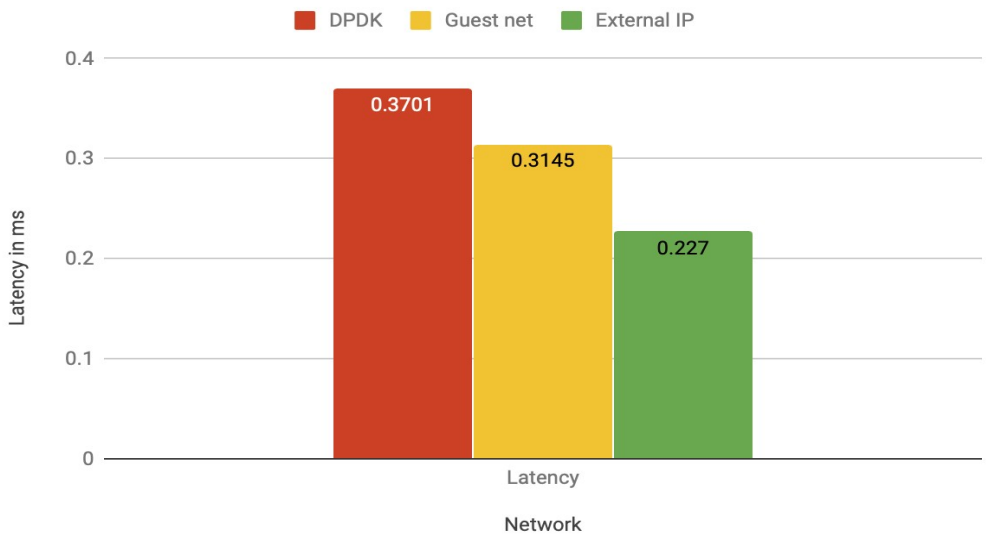
4. Perform the steps #1 thru #3 between an instance with shared CPU and other instance.

8.2 Latency Comparison Graphs

8.2.1 Between instances with dedicated CPUs



8.2.2 From an instance with shared CPUs



9 Running iperf

9.1 East-West

Of the 3 Instances on the same compute node, 2 instances have dedicated CPUs and 1 instance has shared CPUs. Pick an instance with dedicated CPUs to start iperf server and other instance with dedicated CPUs s as iperf client.

1. From the terminal session of the selected instance, start “iperf server”

```
[ubuntu@<instance1-name>~]$ iperf3 -s
```

```
root@dpdk-gr4-i1:/home/ubuntu# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

2. On the other instance, start iperf client on dpdk nic using the below command.

```
[root@<instance2-name>~]$ iperf3 -c <iperf server DPDK ip>
```

```
root@dpdk-gr4-i2:/home/ubuntu# iperf3 -c 12.0.0.13
Connecting to host 12.0.0.13, port 5201
[  4] local 12.0.0.8 port 45804 connected to 12.0.0.13 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   869 MBytes  7.29 Gbits/sec   22    683 KBytes
[  4]   1.00-2.00   sec   876 MBytes  7.35 Gbits/sec    0    683 KBytes
[  4]   2.00-3.00   sec   880 MBytes  7.38 Gbits/sec    0    683 KBytes
[  4]   3.00-4.00   sec   882 MBytes  7.40 Gbits/sec    0    683 KBytes
[  4]   4.00-5.00   sec   880 MBytes  7.38 Gbits/sec    0    684 KBytes
[  4]   5.00-6.00   sec   858 MBytes  7.19 Gbits/sec    0    684 KBytes
[  4]   6.00-7.00   sec   881 MBytes  7.39 Gbits/sec   32    615 KBytes
[  4]   7.00-8.00   sec   882 MBytes  7.40 Gbits/sec    0    615 KBytes
[  4]   8.00-9.00   sec   879 MBytes  7.37 Gbits/sec    0    626 KBytes
[  4]   9.00-10.00  sec   881 MBytes  7.39 Gbits/sec    0    656 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  8.56 GBytes  7.35 Gbits/sec   54             sender
[  4]   0.00-10.00  sec  8.56 GBytes  7.35 Gbits/sec                  receiver
```

4. Once the test completes (10 seconds), note down the transfer and bandwidth displayed.

5. Re-start iperf client using the floating IP

```
# iperf3 -c <iperf server floating ip>
```

```
root@dpdk-gr4-i2:/home/ubuntu# iperf3 -c 10.5.0.56
iperf3: error - unable to connect to server: Connection refused
root@dpdk-gr4-i2:/home/ubuntu# iperf3 -c 192.168.50.56
Connecting to host 192.168.50.56, port 5201
[  4] local 192.168.50.60 port 33816 connected to 192.168.50.56 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   148 MBytes  1.24 Gbits/sec    0   1.38 MBytes
[  4]   1.00-2.00   sec   154 MBytes  1.29 Gbits/sec    0   1.38 MBytes
[  4]   2.00-3.00   sec   155 MBytes  1.30 Gbits/sec    0   1.45 MBytes
[  4]   3.00-4.00   sec   152 MBytes  1.28 Gbits/sec    0   1.45 MBytes
[  4]   4.00-5.00   sec   155 MBytes  1.30 Gbits/sec    0   1.45 MBytes
[  4]   5.00-6.00   sec   151 MBytes  1.27 Gbits/sec    0   1.67 MBytes
[  4]   6.00-7.00   sec   154 MBytes  1.29 Gbits/sec    0   1.67 MBytes
[  4]   7.00-8.00   sec   152 MBytes  1.28 Gbits/sec    0   1.67 MBytes
[  4]   8.00-9.00   sec   155 MBytes  1.30 Gbits/sec    0   1.67 MBytes
[  4]   9.00-10.00  sec   152 MBytes  1.28 Gbits/sec    0   1.67 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.49 GBytes  1.28 Gbits/sec    0          sender
[  4]   0.00-10.00  sec  1.49 GBytes  1.28 Gbits/sec               receiver
```

6. Once the test completes (10 seconds), note down the transfer and bandwidth displayed.

7. Re-start iperf client using the guest-net IP

```
# iperf -c <iperf server guest net ip>
```

```
root@dpdk-gr4-i2:/home/ubuntu# iperf3 -c 10.5.0.38
Connecting to host 10.5.0.38, port 5201
[  4] local 10.5.0.22 port 48386 connected to 10.5.0.38 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   157 MBytes  1.32 Gbits/sec    0   1.14 MBytes
```

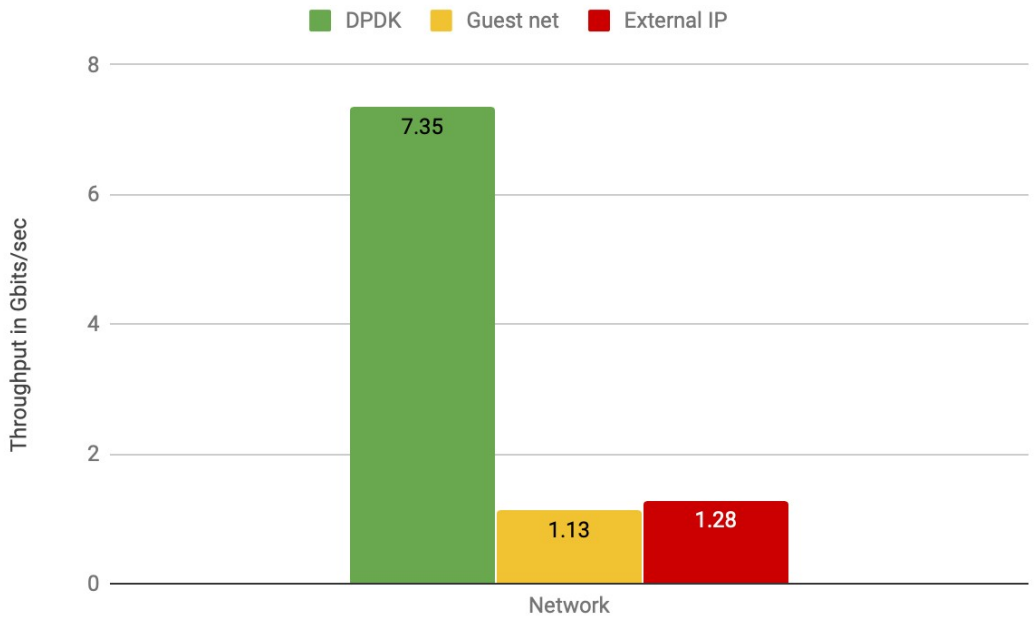
```
[  4]   1.00-2.00   sec   158 MBytes  1.32 Gbits/sec    0   1.21 MBytes
[  4]   2.00-3.00   sec   156 MBytes  1.31 Gbits/sec    0   1.21 MBytes
[  4]   3.00-4.00   sec   152 MBytes  1.28 Gbits/sec    0   1.21 MBytes
[  4]   4.00-5.00   sec   158 MBytes  1.32 Gbits/sec   16    557 KBytes
[  4]   5.00-6.00   sec   156 MBytes  1.31 Gbits/sec    0   1.10 MBytes
[  4]   6.00-7.00   sec   159 MBytes  1.33 Gbits/sec    0   1.13 MBytes
[  4]   7.00-8.00   sec   152 MBytes  1.28 Gbits/sec    0   1.26 MBytes
[  4]   8.00-9.00   sec   158 MBytes  1.32 Gbits/sec    0   1.26 MBytes
[  4]   9.00-10.00  sec   159 MBytes  1.33 Gbits/sec    0   1.26 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.53 GBytes  1.31 Gbits/sec   16          sender
[  4]   0.00-10.00  sec  1.53 GBytes  1.31 Gbits/sec               receiver
```

8. Once the test completes (10 seconds), note down the transfer and bandwidth displayed.

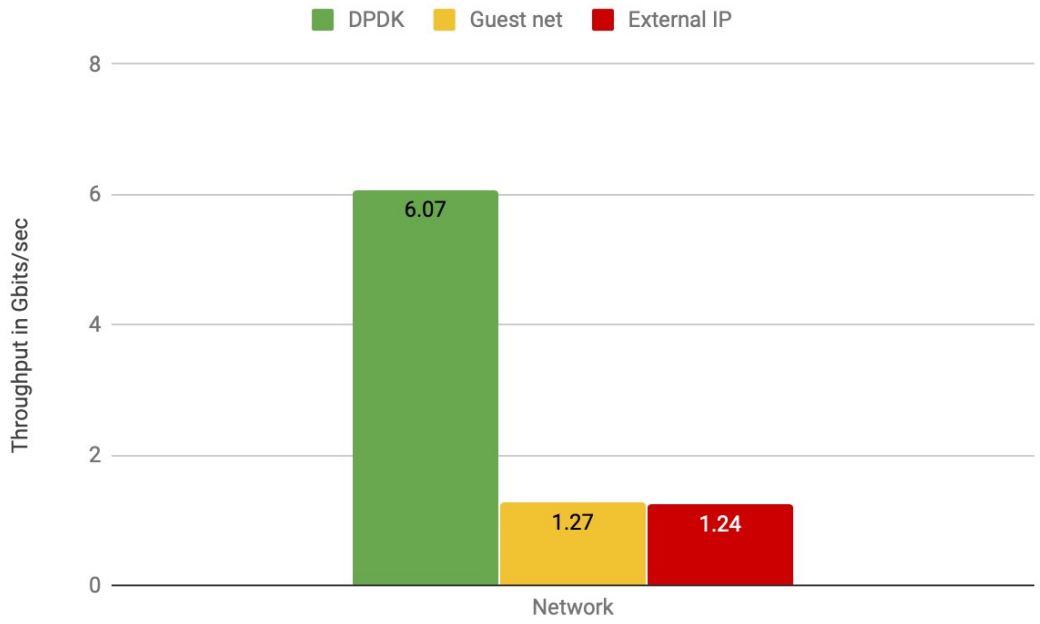
9. Compare the throughput between steps 4, 6 & 8. Step 4 throughput is in general higher than the throughput in steps 6 & 8. See the highlighted part in steps 4, 6 & 8 to observe the difference.

9.2 iperf Comparison Graphs

9.2.1 East - West



10. Perform the steps #1 thru #9 with iperf server on instance with dedicated CPUs and iperf client on instance with shared CPUs.



10 Appendix

10.1 Configuring OVS with DPDK in Openstack Environment

Below are a high level steps to configure OVS with DPDK in openstack environment.

10.1.1 Compute Node specific operations

- Configure Huge pages
- Install openvswitch-switch-dpdk package
- There are issues configuring OVS with DPDK that comes with the ubuntu package. So, using the link below, configure OVS with DPDK option.

  http://docs.openvswitch.org/en/latest/intro/install/dpdk/

  - Observed issues with Openstack Ansible, when OVS was configured from sources. To workaround the issue, we installed openvswitch-switch-dpdk package and then updating the files built from sources
- Once Openvswitch is configured with DPDK, set the below properties for OVS.
  - dpdk-extra="--pci-whitelist <interface PCI bus id>"
  - dpdk-init="true"
  - dpdk-lcore-mask=<specific cpu cores>
  - dpdk-socket-mem=<huge page memory>
  - pmd-cpu-mask=<cores dedicated for Poll mode drivers>
  - vhost-iommu-support="true"
- Below are the values configured for the lab environment.

  {dpdk-extra="--pci-whitelist 0000:b1:00.0,0000:b1:00.1", dpdk-init="true", dpdk-lcore-mask="0xc00000", dpdk-socket-mem="0,4096", max-idle="50000", n-dpdk-rxqs="512", n-dpdk-txqs="512", pmd-cpu-mask="0x3000000000003000000", vhost-iommu-support="true"}

- For each NIC that you plan to use for DPDK, create a bridge in OVS. Below is the bridge created for Create Below are the values configured for the lab environment.

  Bridge "br0"

  Controller "tcp:127.0.0.1:6633"

  is_connected: true

  fail_mode: secure

```
        Port "port0"
            Interface "port0"
                type: dpdk
                options: {dpdk-devargs="0000:b1:00.0", n_rxq_desc="128",
n_txq_desc="128"}
        Port "br0"
            Interface "br0"
                type: internal
        Port "phy-br0"
            Interface "phy-br0"
                type: patch
                options: {peer="int-br0"}
```

10.1.2 Openstack Specific Settings

- On the controller and compute nodes, as part of ml2 configuration, add **openvswitch** and **l2population** to **mechanism_drivers**
- On the compute node(s), in the neutron openvswitch agent, configure the below
  - Under the OVS section, add **integration_bridge = br-int**
  - Under the OVS section, configure bridge_mappings property to point to the dpdk bridge. In lab environment, the below is configured
    - bridge_mappings = dpdk0:br0
  - Under the **Agent** section, **l2_population** should be **True**
  - Under the **securitygroup** section, **firewall** driver should be **openvswitch**

10.1.3 Openstack Dashboard Specific Changes

1. Create a network for DPDK with **Provider Network Type** as **Flat** and for **Physical Network**, mention the name provided for device_mappings in openvswitch_agent.ini file.
2. A flavor needs to be configured with the below properties
   a. aggregate_instance_extra_specs:hpgs='true'
   b. hw:cpu_policy='dedicated'
   c. hw:mem_page_size='1GB'
3. While creating instances, in the **Flavor** page, select a flavor with the properties mentioned in #2 above.
4. In the **Network** page, select the DPDK network
5. In the **Key Pair** page, provide a valid key-pair.

10.2 Reference

Add references of DPDK from Intel and OpenStack <TBD>

https://01.org/packet-processing/blogs/nsundar/2018/nfv-i-host-configuration-low-latency

https://software.intel.com/sites/default/files/managed/4a/00/DPDK-Cookbook.pdf

https://software.intel.com/en-us/articles/dpdk-performance-optimization-guidelines-white-paper

https://software.intel.com/sites/default/files/managed/0e/b9/A-Low-Latency-NFV-Infrastructure-for-Performance-Critical-Applications.pdf