Mellanox Support for TripleO Train

Application Notes

Rev 1.0

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT ("PRODUCT(S)") AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES "AS-IS" WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox TECHNOLOGIES

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Table of Contents

Document Revision History ... **6**

Definitions, Acronyms and Abbreviations ... **7**

1 Mellanox OVS Hardware Offloading Support for TripleO ... **9**

1.1 Supported Features ... 9

1.2 System Requirements ... 10

1.3 Supported Network Interface Cards and Firmware ... 10

1.4 Supported Operating Systems ... 10

1.5 Overcloud Operating System Versions ... 11

2 ASAP^2 Support ... **12**

2.1 ASAP^2 Support Over Open vSwitch ... 12

2.1.1 Network Card Support Matrix and Limitations ... 12

2.1.2 Configuration ... 12

2.1.3 Deploying the Overcloud ... 14

2.1.4 Booting the VM ... 14

2.2 Checking Hardware Offloading ... 15

2.3 Verifying Hardware Offloading Configuration ... 17

2.4 Deploying TripleO with VF LAG Configuration ... 18

2.5 Deploy with GRE Tunnel Type ... 20

2.5.1 Network Cards Support Matrix and Limitations ... 20

2.5.2 Configuration ... 20

2.5.3 Deploying the Overcloud ... 20

2.5.4 Booting the VM ... 20

3 NVMe over Fabrics (NVMe-oF) ... **22**

3.1 Network Cards Support Matrix and Limitations ... 22

3.2 Deployment of Non-Containerized Overcloud ... 22

3.2.1 Configuration ... 22

3.2.2 Deploying the NVMe-oF Overcloud ... 22

3.3 Deployment of Containerized Overcloud ... 23

3.3.1 Configuration ... 23

3.3.2 Deploying the NVMe-oF Overcloud ... 23

4 Bare Metal Provision with BlueField ... **25**

4.1 Supported Features ... 25

4.2 Preparing BlueField ... 25

4.3 Creating Neutron Agent Container on BlueField ... 26

4.4 Deployment of TripleO with Bare-Metal Service ... 26

4.5 Post-Deployment: Patch Neutron Server Container ... 26

4.6 BlueField Network Configuration 27
4.6.1 Network Configuration in BlueField 27
4.7 Add BlueField Ironic Images 28
4.8 Create Overcloud Networks 28
4.9 Bare-Metal Flavor 29
4.10 Disable Automated Cleaning for Ironic 29
4.11 Add Bare-Metal Node 29
4.12 Boot Bare Metal Instance 30
5 Configuring Mellanox SDN Mechanism Driver Plugin Using TripleO 31
5.1 Configure and Prepare the NEO 31
5.2 Enable LLDP on Mellanox Switch 31
5.3 Install RPM Package on Container Image 31
5.4 Configure the Mellanox SDN Mechanism Driver Plugin 32
5.5 Set NTP Server 33
5.6 Install LLDPAD Package to Overcloud Image 33
5.7 Configure LLDPAD in First Boot 33

List of Tables

Table 1: Document Revision History 6
Table 2: Definitions, Acronyms, and Abbreviations 7
Table 3: Undercloud Node Requirements 10
Table 4: Supported Operating Systems 10
Table 5: Overcloud Operating System Versions 11

Document Revision History

Table 1: Document Revision History

Revision	Date	Description
1.0	October 27, 2019	First update of the document

Definitions, Acronyms and Abbreviations

Table 2: Definitions, Acronyms, and Abbreviations

Term	Description
SR-IOV	Single Root I/O Virtualization (SR-IOV) is a specification that allows a PCI device to appear virtually on multiple Virtual Machines (VMs), each of which has its own virtual function. This specification defines virtual functions (VFs) for the VMs and a physical function for the hypervisor. Using SR-IOV in a cloud infrastructure helps to achieve higher performance since traffic bypasses the TCP/IP stack in the kernel.
RoCE	RDMA over Converged Ethernet (RoCE) is a standard protocol which enables RDMA's efficient data transfer over Ethernet networks allowing transport offload with hardware RDMA engine implementation, and superior performance. RoCE is a standard protocol defined in the InfiniBand Trade Association (IBTA) standard. RoCE makes use of UDP encapsulation allowing it to transcend Layer 3 networks. RDMA is a key capability natively used by the InfiniBand interconnect technology. Both InfiniBand and Ethernet RoCE share a common user API but have different physical and link layers.
ConnectX®-5	ConnectX-5 adapter cards support two ports of 100Gb/s Ethernet connectivity, sub-700 nanosecond latency, a very high message rate, and PCIe switch and NVMe over Fabric offloads, providing the highest performance and most flexible solution for the most demanding applications and markets. It uses Accelerated Switching and Packet Processing (ASAP^{2}™) technology which enhances offloading of virtual switches and virtual routers, such as Open V-Switch (OVS), which results in significantly higher data transfer performance without overloading the CPU. Together with native RoCE and DPDK (Data Plane Development Kit) support, ConnectX-5 dramatically improves Cloud and NFV platform efficiency.
Open vSwitch (OVS)	Open vSwitch (OVS) allows Virtual Machines (VM) to communicate with each other and with the outside world. OVS traditionally resides in the hypervisor and switching is based on twelve tuples matching on flows. The OVS software-based solution is CPU intensive, affecting system performance and preventing fully utilizing available bandwidth.
OVS-DPDK	OVS-DPDK extends Open vSwitch performances while interconnecting with Mellanox DPDK Poll Mode Driver (PMD). It accelerates the hypervisor networking layer for better latency and higher packet rate while maintaining Open vSwitch data plane networking characteristics.
ASAP^{2}	Mellanox ASAP^{2}—Accelerated Switching And Packet Processing® technology allows to offload OVS by handling OVS data-plane in Mellanox ConnectX-5 (and onwards) NIC hardware (Mellanox Embedded Switch or eSwitch) while maintaining OVS control-plane unmodified. As a result, we observe significantly higher OVS performance without the associated CPU load. The current actions supported by ASAP^{2} include packet parsing and matching, forward, drop along with VLAN push/pop or VXLAN encapsulated/decapsulated.

Term	Description
NVMEoF	NVMeOF or NVMe over Fabrics is a network protocol, like iSCSI, used to communicate between a host and a storage system over a network (a.k.a. fabric). It depends on and requires the use of RDMA. NVMeOF can use any of the RDMA technologies including InfiniBand and RoCE.

1 Mellanox OVS Hardware Offloading Support for TripleO

TripleO (OpenStack On OpenStack) is a program aimed at installing, upgrading, and operating OpenStack clouds using OpenStack's own cloud facilities as the foundations, building on Nova, Neutron, and Heat to automate fleet management at datacenter scale.

Open vSwitch (OVS) allows Virtual Machines (VMs) to communicate with each other and with the outside world. OVS traditionally resides in the hypervisor and the switching is based on twelve-tuple matching on flows. The OVS software-based solution is CPU intensive, affecting system performance and prevents the full utilization of the available bandwidth. $ASAP^2$—Accelerated Switching and Packet Processing® technology allows to offload OVS by handling the OVS data plane in Mellanox ConnectX-5 Network Interface Card (NIC) hardware (embedded switch or eSwitch) while leaving the control-plane of the OVS unmodified. As a result, the OVS performance significantly increases without the associated CPU load.

This Application Notes document details how to enable the Mellanox $ASAP^2$ technology feature of hardware-offloading support over OVS and OVN mechanism drivers.

1.1 Supported Features

TripleO Stein supports the following features:

- $ASAP^2$ with OVS mechanism driver
- $ASAP^2$ with OVN mechanism driver
- OVS over DPDK with inbox driver
- NVMe over Fabric (NVMeOF)
- BlueField with bare metal

1.2 System Requirements

The system requirements are detailed in the following table.

Table 3: Undercloud Node Requirements

Platform	Type and Version
OS	Red Hat Enterprise Linux 7.6.
CPU	8-core 64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions.
Memory	Minimum 16 GB of RAM.
Disk Space	Minimum 40 GB of available disk space on the root disk. At least 10 GB of free space should be left before attempting an overcloud deployment or update. This free space accommodates image conversion and caching during the node provisioning process.
Networking	Minimum of 2 x 1Gb/s NICs. However, it is recommended to use a 10Gb/s interface for provisioning network traffic, especially if provisioning many nodes in the overcloud environment. Use Mellanox NIC for tenant network.

1.3 Supported Network Interface Cards and Firmware

Mellanox support for TripleO Stein supports the following Mellanox NICs and their corresponding firmware versions:

NIC	Supported Protocols	Recommended Firmware Rev.
BlueField	Ethernet	18.25.1020
ConnectX®-5	Ethernet	16.25.1020
ConnectX®-4 Lx	Ethernet	14.25.1020
ConnectX®-4	Ethernet	12.25.1020
ConnectX®-3 Pro	Ethernet	2.42.5000

1.4 Supported Operating Systems

The following operating systems are the supported:

Table 4: Supported Operating Systems

OS	Platform
RHEL7.6	x86_64

1.5 Overcloud Operating System Versions

The following overcloud operating system versions are supported:

Table 5: Overcloud Operating System Versions

Item	Version
Kernel	kernel-5.0.0-1.x86_64 kernel-headers-5.0.0-1.x86_64
Iproute	openvswitch-2.11.1-1.el7.centos.x86_64
Open vSwitch	openvswitch-2.11.1-1.el7.centos.x86_64

2 ASAP² Support

2.1 ASAP² Support Over Open vSwitch

2.1.1 Network Card Support Matrix and Limitations

The following Mellanox cards support ASAP² hardware offloading feature:

NICs	Supported Protocols
ConnectX®-5	Ethernet

2.1.2 Configuration

Starting from a fresh RHEL 7.6 bare metal server, install and configure the undercloud according to the official TripleO installation documentation.

1. Use the `ovs-hw-offload.yaml` file from the following location:

```
/usr/share/openstack-tripleo-heat-templates/environments/ovs-hw-
offload.yaml
```

Configure it over VLAN/VXLAN setup in the following way:

a. In the case of a **VLAN** setup, configure the `ovs-hw-offload.yaml`:

```
# A Heat environment file that enables OVS Hardware Offload in the
overcloud.
# This works by configuring SR-IOV NIC with switchdev and OVS Hardware
Offload on
# compute nodes. The feature supported in OVS 2.8.0


parameter_defaults:
  NeutronFlatNetworks: datacentre
  NeutronNetworkType:
  - vlan
  NeutronTunnelTypes: ''

  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','ComputeFilter','ComputeCapabi
litiesFilter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter',
'ServerGroupAffinityFilter','PciPassthroughFilter','NUMATopologyFilter
']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_pass
through_filter.PciPassthroughFilter"]
  NovaPCIPassthrough:
    - devname: <interface_name>
      physical_network: datacentre
  # Mapping of SR-IOV PF interface to neutron physical_network.
  #     # In case of Vxlan/GRE physical_network should be null.
  #         # In case of flat/vlan the physical_network should as
configured in neutron.


  ComputeSriovParameters:

    NeutronBridgeMappings:
    - datacentre:br-ex
    OvsHwOffload: True
```

b. In the case of a **VXLAN** setup, do the following:

i. Configure the `ovs-hw-offload.yaml`:

```
    # A Heat environment file that enables OVS Hardware Offload in
the overcloud.
    # This works by configuring SR-IOV NIC with switchdev and OVS
Hardware Offload on
    # compute nodes. The feature supported in OVS 2.8.0


    parameter_defaults:
      NeutronFlatNetworks: datacentre

      NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','ComputeFilter','ComputeCapa
bilitiesFilter','ImagePropertiesFilter','ServerGroupAntiAffinityFilt
er','ServerGroupAffinityFilter','PciPassthroughFilter','NUMATopology
Filter']
      NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_pa
ssthrough_filter.PciPassthroughFilter"]
  NovaPCIPassthrough:
    - devname: <interface_name>
      physical_network: null
      # Mapping of SR-IOV PF interface to neutron physical_network.
      # In case of Vxlan/GRE physical_network should be null.
      # In case of flat/vlan the physical_network should as
configured in
  #neutron.

      ComputeSriovParameters:
    NeutronBridgeMappings:
    - datacentre:br-ex
    OvsHwOffload: True
```

ii. Configure the interface names in the `/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans/control.yaml` files by adding the following code to move the tenant network from VLAN on a bridge to be on a separated interface.

```
-type: interface
 name: <interface_name>
 addresses:
 -ip_netmask:
     get_param: TenantIpSubnet
```

iii. Configure the interface names in the `/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans/compute.yaml` files by adding the following code to move the tenant network from VLAN on a bridge to be on a separated interface.

```
- type: sriov_pf
  name:
  get_param: enp3s0f0
  link_mode: switchdev
  numvfs: 64
  promisc: true
  use_dhcp: false
```

2. Create a new role for the compute node and change it to ComputeSriov.

```
# openstack overcloud roles generate -o roles_data.yaml Controller
ComputeSriov
```

3. Update the `~/cloud-names.yaml` accordingly.
   See the following example:

```
parameter_defaults:
ComputeSriovCount: 2
OvercloudComputeSriovFlavor: compute
```

4. Assign the `compute.yaml` file to the `ComputeSriov` role. Update the `~/heat-templates/environments/net-single-nic-with-vlans.yaml` file by adding the following line:

```
OS::TripleO::ComputeSriov::Net::SoftwareConfig: ../network/config/single-nic-vlans/compute.yaml
```

5. Run `overcloud-prep-containers.sh`

2.1.3 Deploying the Overcloud

Deploy the overcloud using the appropriate templates and yamls from `~/heat-templates` as in the following example:

```
openstack overcloud deploy \
  --templates ~/heat-templates \
  --libvirt-type kvm  -r ~/roles_data.yaml \
  -e /home/stack/containers-default-parameters.yaml  \
  -e ~/heat-templates/environments/docker.yaml \
  -e ~/heat-templates/enviroments/ovs-hw-offload.yaml \
  --control-flavor oooq_control \
  --compute-flavor oooq_compute \
  --ceph-storage-flavor oooq_ceph \
  --block-storage-flavor oooq_blockstorage \
  --swift-storage-flavor oooq_objectstorage \
  --timeout 90 \
-e /home/stack/cloud-names.yaml  \
-e ~/heat-templates/environments/network-isolation.yaml \
-e ~/heat-templates/environments/net-single-nic-with-vlans.yaml \
-e /home/stack/network-environment.yaml \
-e ~/heat-templates/environments/disable-telemetry.yaml  \
--validation-warnings-fatal \
--ntp-server pool.ntp.org
```

2.1.4 Booting the VM

➢ ***To boot the VM on the undercloud machine, do the following:***

1. Load the overcloudrc configuration.

```
# source ./overcloudrc
```

2. Create a flavor.

```
# openstack flavor create m1.small --id 3 --ram 2048 --disk 20 --vcpus 1
```

3. Create “cirrios” image.

```
$ openstack image create --public --file cirros-mellanox_eth.img --disk-format qcow2 --container-format bare mellanox
```

4. Create a network.

   a. In the case of VLAN network:

```
$ openstack network create private --provider-physical-network datacentre --provider-network-type vlan –share
```

   b. In the case of VXLAN network:

```
$ openstack network create private --provider-network-type vxlan –
share
```

5. Create subnet.

```
$ openstack subnet create private_subnet --dhcp --network private --
subnet-range 11.11.11.0/24
```

6. Boot a VM on the overcloud using the following command after creating the direct port accordingly.

   - For the first VM:

```
$ direct_port1=`openstack port create direct1 --vnic-type=direct --
network private --binding-profile '{"capabilities":["switchdev"]}' |
grep ' id ' | awk '{print $4}'`

$openstack server create --flavor 3 --image mellanox --nic port-
id=$direct_port1 vm1
```

   - For the second VM:

```
$ direct_port2=`openstack port create direct2 --vnic-type=direct --
network private --binding-profile '{"capabilities":["switchdev"]}' |
grep ' id ' | awk '{print $4}'`

$ openstack server create --flavor 3 --image mellanox --nic port-
id=$direct_port2 vm2
```

2.2 Checking Hardware Offloading

To check whether or no hardware offloading is working, create two VMs: one on each compute node, as described below, and then use `tcpdump` on the representor port on the compute node to see if only two ICMP packets exist.

1. Use the Nova list to view the IP address created VMs from step 6 in section 1.

```
$ count=1 | for i in `nova list | awk 'NR > 2 {print $12}' | cut -d'=' -f
2` ; do echo "VM$count=$i"; count=$(($count+1)) ; done
VM1=11.11.11.8
VM2=11.11.11.9
```

2. Ping from a VM to VM over two hypervisors in same network.

   a. On the first VM, run the ping command `ping <second_vm_ip_address>`. In the example below, 11.11.11.9 is used as the second VM IP address.

```
$ ping 11.11.11.9
PING 11.11.11.9 (11.11.11.9): 56 data bytes
64 bytes from 11.11.11.9: seq=0 ttl=64 time=65.600 ms
64 bytes from 11.11.11.9: seq=1 ttl=64 time=0.153 ms
64 bytes from 11.11.11.9: seq=2 ttl=64 time=0.109 ms
64 bytes from 11.11.11.9: seq=3 ttl=64 time=0.095 ms
64 bytes from 11.11.11.9: seq=4 ttl=64 time=0.121 ms
64 bytes from 11.11.11.9: seq=5 ttl=64 time=0.081 ms
64 bytes from 11.11.11.9: seq=6 ttl=64 time=0.121 ms
64 bytes from 11.11.11.9: seq=7 ttl=64 time=0.127 ms
64 bytes from 11.11.11.9: seq=8 ttl=64 time=0.123 ms
64 bytes from 11.11.11.9: seq=9 ttl=64 time=0.123 ms
```

   b. On the compute node that contains the VM, identify the representor port used by the VM.

```
# ip link show enp3s0f0
6: enp3s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
master ovs-system state UP mode DEFAULT group default qlen 1000
```

```
    link/ether ec:0d:9a:46:9e:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking off, link-state enable,
trust off, query_rss off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state enable,
trust off, query_rss off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state enable,
trust off, query_rss off
    vf 3 MAC fa:16:3e:b9:b8:ce, vlan 57, spoof checking on, link-state
enable, trust off, query_rss off

#ls -l /sys/class/net/|grep eth
lrwxrwxrwx 1 root root 0 Sep 11 10:54 eth0 ->
../../devices/virtual/net/eth0

lrwxrwxrwx 1 root root 0 Sep 11 10:54 eth1 ->
../../devices/virtual/net/eth1

lrwxrwxrwx 1 root root 0 Sep 11 10:54 eth2 ->
../../devices/virtual/net/eth2

lrwxrwxrwx 1 root root 0 Sep 11 10:54 eth3 ->
../../devices/virtual/net/eth3


#sudo ovs-dpctl show

system@ovs-system:

        lookups: hit:1684 missed:1465 lost:0
        flows: 0
        masks: hit:8420 total:1 hit/pkt:2.67
        port 0: ovs-system (internal)
        port 1: br-enp3s0f0 (internal)
        port 2: br-int (internal)
        port 3: br-ex (internal)
        port 4: enp3s0f0
        port 5: tapfdc744bb-61 (internal)
        port 6: qr-a7b1e843-4f (internal)
        port 7: qg-79a77e6d-8f (internal)
        port 8: qr-f55e4c5f-f3 (internal)
        port 9: eth3
```

c. Check that the hardware offloading rules are working using `tcpdump` on `eth3` (the representor port).

```
# tcpdump -i eth3 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth3, link-type EN10MB (Ethernet), capture size 262144
bytes
08:51:35.792856 IP 11.11.11.8 > 11.11.11.9: ICMP echo request, id
58113, seq 0, length 64
08:51:35.858251 IP 11.11.11.9 > 11.11.11.8: ICMP echo reply, id 58113,
seq 0, length 64
```

2.3 Verifying Hardware Offloading Configuration

1. Check that hardware offload is configured on the compute.

```
# ovs-vsctl get Open_vSwitch . other_config:hw-offload
"true"
```

2. Check the mode and inline-mode for the offloaded port for the ConectX-5 card.

```
# devlink dev eswitch show pci/0000:03:00.0
pci/0000:03:00.0: mode switchdev inline-mode none encap enable
```

3. Check if your version of ethtool support setting can enable TC offloads.

```
# ethtool -k <interface_name>
Features for <interface_name>:
rx-checksumming: on
tx-checksumming: on
        tx-checksum-ipv4: on
        tx-checksum-ip-generic: off [fixed]
        tx-checksum-ipv6: on
        tx-checksum-fcoe-crc: off [fixed]
        tx-checksum-sctp: off [fixed]
scatter-gather: on
        tx-scatter-gather: on
        tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp-ecn-segmentation: off [fixed]
        tx-tcp-mangleid-segmentation: off
        tx-tcp6-segmentation: on
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
rx-vlan-offload: on
tx-vlan-offload: on
ntuple-filters: off
receive-hashing: on
highdma: on [fixed]
rx-vlan-filter: on
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-gre-csum-segmentation: off [fixed]
tx-ipxip4-segmentation: off [fixed]
tx-ipxip6-segmentation: off [fixed]
tx-udp_tnl-segmentation: on
tx-udp_tnl-csum-segmentation: on
tx-gso-partial: on
tx-sctp-segmentation: off [fixed]
tx-esp-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
rx-fcs: off
rx-all: off
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off [fixed]
hw-tc-offload: on
esp-hw-offload: off [fixed]
esp-tx-csum-hw-offload: off [fixed]
```

4. Reboot the compute node to make sure the VFs still exist to verify that the configuration of the `switchdev` is persistent.

```
# lspci | grep Mellanox
03:00.0 Ethernet controller: Mellanox Technologies MT27800 Family
[ConnectX-5]
03:00.1 Ethernet controller: Mellanox Technologies MT27800 Family
[ConnectX-5]
03:00.2 Ethernet controller: Mellanox Technologies MT27800 Family
[ConnectX-5 Virtual Function]
03:00.3 Ethernet controller: Mellanox Technologies MT27800 Family
[ConnectX-5 Virtual Function]
03:00.4 Ethernet controller: Mellanox Technologies MT27800 Family
[ConnectX-5 Virtual Function]
03:00.5 Ethernet controller: Mellanox Technologies MT27800 Family
[ConnectX-5 Virtual Function]
81:00.0 Ethernet controller: Mellanox Technologies MT27710 Family
[ConnectX-4 Lx]
81:00.1 Ethernet controller: Mellanox Technologies MT27710 Family
[ConnectX-4 Lx]
```

5. On the ComputeSriov node, check that the `dumpxml` on the compute node contains the VF port:

```
# virsh list
 Id    Name                           State
----------------------------------------------------
 1      instance-00000001              running
```

6. Check the `dmpxml` for the VF port.

```
# virsh dumpxml instance-00000001
<interface type='hostdev'
managed='yes'>
      <mac address='fa:16:3e:57:ea:a2'/>
      <driver name='vfio'/>
      <source>
       <address type='pci' domain='0x0000' bus='0x03' slot='0x00'
function='0x5'/>
      </source>
      <alias name='hostdev0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04'
function='0x0'/>
    </interface>
```

2.4 Deploying TripleO with VF LAG Configuration

This feature is supported in **kernel v5.0 RC and above**.

1. Make sure the `compute.yaml` file has a Linux bond:

```
              - type: linux_bond
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
                name: bond0
                bonding_options:
                  get_param: BondInterfaceOvsOptions
                members:
                - type: sriov_pf
                  name:
                  get_param: enp3s0f0
                  link_mode: switchdev
```

```
                  numvfs: 64
                  promisc: true
                  use_dhcp: false
                - type: sriov_pf
                  name:
                  get_param: enp3s0f1
                  link_mode: switchdev
                  numvfs: 64
                  promisc: true
                  use_dhcp: false
```

2. Make sure the `/usr/share/openstack-tripleo-heat-templates/environments/ovs-hw-offload.yaml` file has VFs for the two ports of the Linux bond and `hw-offloading` enabled:

```
  ComputeSriovParameters:
    NeutronSriovNumVFs: ["enp3s0f0:4:switchdev","enp3s0f1:4:switchdev"]
    OvsHwOffload: True
```

3. Configure the bonding option in the same file.

```
parameter_defaults
  BondInterfaceOvsOptions: "mode=active-backup miimon=100"
```

> The supported bonding mode for vf-lag are:
> - Active-Backup
> - Active-Active
> - LACP

Below is an example of an uplink over a VLAN number 77 over a bond use:

```
              - name: bond0.77
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
                type: interface
                use_dhcp: false
```

Below is an example of an uplink of a general interface:

```
              - type: interface
                name: enp2s0f1.70
                use_dhcp: false
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
```

2.5 Deploy with GRE Tunnel Type

2.5.1 Network Cards Support Matrix and Limitations

The following Mellanox cards support the ASAP[2] hardware-offloading feature:

NICs	Supported Protocols	Supported Network Type
ConnectX®-5	Ethernet	Support hardware offloading over VLAN, VXLAN, and GER.

Use firmware version 16.24.1000 or newer for ConnectX-5 to support GRE hardware offloading.

2.5.2 Configuration

Starting from a fresh RHEL 7.5 bare metal server, install and configure the undercloud according to the official TripleO installation documentation.

4. Update `enviroments/ovs-hw-offload.yaml` to use GRE as Neutron tunnel type.

```
parameter_defaults:
  NeutronFlatNetworks: datacentre
  NeutronNetworkType: 'vlan,gre'
  NeutronTunnelTypes: 'gre'
```

2.5.3 Deploying the Overcloud

Deploy overcloud using the appropriate templates and yamls from `~/heat-templates` as descried in section 2.1.3.

2.5.4 Booting the VM

On the undercloud machine, do the following:

5. Load the `overcloudrc` configuration.

```
# source overcloudrc
```

6. Create a flavor.

```
# openstack flavor create m1.small --id 3 --ram 2048 --disk 20 --vcpus 1
```

7. Create "cirrios" image.

```
$ openstack image create --public --file cirros-mellanox_eth.img --disk-format qcow2 --container-format bare mellanox
```

8. Create a network.

```
$ openstack network create private --provider-network-type gre –-share
```

9. Create subnet.

```
$ openstack subnet create private_subnet --dhcp --network private --
subnet-range 11.11.11.0/24
```

10. Boot a VM on the overcloud using the following command after creating the direct port accordingly.

- For the first VM:

```
$ direct_port1=`openstack port create direct1 --vnic-type=direct --
network private --disable-port-security  --binding-profile
'{"capabilities":["switchdev"]}' | grep ' id ' | awk '{print $4}'`

$openstack server create --flavor 3 --image mellanox --nic port-
id=$direct_port1 vm1
```

- For the second VM:

```

$ direct_port2=`openstack port create direct2 --vnic-type=direct --
network private --disable-port-security  --binding-profile
'{"capabilities":["switchdev"]}' | grep ' id ' | awk '{print $4}'`

$ openstack server create --flavor 3 --image mellanox --nic port-
id=$direct_port2 vm2
```

3 NVMe over Fabrics (NVMe-oF)

3.1 Network Cards Support Matrix and Limitations

The following Mellanox network cards support the NVMe-oF feature:

NICs	Supported Protocols
ConnectX®-4	Ethernet
ConnectX®-4 Lx	Ethernet
ConnectX®-5	Ethernet

3.2 Deployment of Non-Containerized Overcloud

3.2.1 Configuration

Starting from a fresh RHEL 7.5 bare metal server, install and configure the undercloud according to the official TripleO installation documentation.

1. Install `nvmetcli` and `nvme-cli` packages on the image.

```
virt-customize -x -v -a overcloud-full.qcow2  --run-command 'sudo yum
install nvmetcli nvme-cli -y'
```

2. Upload the image.

```
openstack overcloud image upload --image-path ~/ --update-existing
```

3. Change the `cinder-nvmeof-config.yaml` environment file (if needed).
   The `cinder-nvmeof-config.yaml` file contains the Cinder NVMe-oF backend parameters.

```
vi ~/tripleo-heat-templates/environments/cinder-nvmeof-config.yaml
```

4. Prepare your deployment files as needed, then add the `cinder-nvmeof-config.yaml` environment file to your deployment script `cinder-nvmeof-config.yaml`.

```
-e /home/stack/tripleo-heat-templates/environments/cinder-nvmeof-
config.yaml
```

3.2.2 Deploying the NVMe-oF Overcloud

Deploy the overcloud using the appropriate templates and yamls from `~/heat-templates` as in the following example:

```
openstack overcloud deploy \
    --templates /home/stack/tripleo-heat-templates \
    -r /home/stack/roles_data.yaml \
    --libvirt-type kvm
    --timeout 90
    -e /home/stack/cloud-names.yaml
    -e /home/stack/tripleo-heat-templates/environments/network-
isolation.yaml
    -e /home/stack/tripleo-heat-templates/environments/net-single-nic-with-
vlans.yaml -e /home/stack/network-environment.yaml
    -e /home/stack/enable-tls.yaml
    -e /home/stack/tripleo-heat-templates/environments/tls-endpoints-public-
ip.yaml -e /home/stack/inject-trust-anchor.yaml
```

```
    --verbose --ntp-server pool.ntp.org -e /home/stack/tripleo-heat-
templates/environments/host-config-and-reboot.yaml
    -e /home/stack/tripleo-heat-templates/environments/baremetal-
services.yaml
    -e ~/nic_configs/network.yaml
    -e /home/stack/tripleo-heat-templates/environments/cinder-nvmeof-
config.yaml
    -e /home/stack/tripleo-heat-templates/environments/disable-
telemetry.yaml
```

3.3 Deployment of Containerized Overcloud

3.3.1 Configuration

Starting from a fresh RHEL 7.5 bare metal server, install and configure the undercloud according to the official TripleO installation documentation.

1. Prepare the container images.

```
./overcloud-prep-containers.sh
```

2. Change the `cinder-nvmeof-config.yaml` environment file (if needed). The `cinder-nvmeof-config.yaml` file contains the Cinder NVMe-oF backend parameters.

```
vi ~/tripleo-heat-templates/environments/cinder-nvmeof-config.yaml
```

3. Prepare deployment files as desired. Then add the `cinder-nvmeof-config.yaml` environment file to the deployment script `cinder-nvmeof-config.yaml`.

```
-e /home/stack/tripleo-heat-templates/environments/cinder-nvmeof-
config.yaml
```

3.3.2 Deploying the NVMe-oF Overcloud

Deploy the overcloud using the appropriate templates and yamls from `~/heat-templates`, as in the following example:

```
openstack overcloud deploy \
    --templates /usr/share/openstack-tripleo-heat-templates \
    --libvirt-type kvm \
    --control-flavor oooq_control \
    --compute-flavor oooq_compute  \
    --ceph-storage-flavor oooq_ceph \
    --block-storage-flavor oooq_blockstorage  \
    --swift-storage-flavor oooq_objectstorage   \
    --timeout 90  \
    -e /usr/share/openstack-tripleo-heat-templates/environments/docker.yaml\
    -e /home/stack/cloud-names.yaml \
    -e /home/stack/containers-default-parameters.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml  \
    -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-
nic-with-vlans.yaml \
    -e /home/stack/network-environment.yaml  \
    -e /usr/share/openstack-tripleo-heat-templates/environments/low-memory-
usage.yaml \
    -e /home/stack/enable-tls.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/tls-
endpoints-public-ip.yaml \
    -e /home/stack/inject-trust-anchor.yaml  \
    -e /usr/share/openstack-tripleo-heat-templates/environments/disable-
telemetry.yaml \
    --validation-warnings-fatal  \
```

```
    --ntp-server pool.ntp.org \
    -e ~/nic_configs/network.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/cinder-
nvmeof-config.yaml \
```

4 Bare Metal Provision with BlueField

BlueField® SmartNIC adapters accelerate a wide range of applications through flexible data and control-plane offloading. Enabling a more efficient use of compute resources, BlueField adapters empower the CPU to focus on running applications rather than on networking or security processing. Additionally, as software-defined adapters, BlueField SmartNICs ensure the ultimate flexibility by adapting to future protocols and features through simple software updates.

4.1 Supported Features

Mellanox BlueField SmartNIC supports the following Features:

- Mellanox ASAP^2—Accelerated Switching and Packet Processing® for Open vSwitch (OVS) delivers flexible, highly efficient virtual switching and routing capabilities. OVS accelerations can be further enhanced using BlueField processing and memory. For example, the scale of OVS actions can be increased by utilizing BlueField internal memory, and more OVS actions and vSwitch/vRouter implementations can be supported.
- Network overlay technology (VXLAN) offload, including encapsulation and decapsulation, allows the traditional offloads to operate on the tunneled protocols, and offload Network Address Translation (NAT) routing capabilities.

4.2 Preparing BlueField

1. Install the latest operating system on BlueField according to the BlueField Installation Guide.
2. Validate representor ports presence.
3. Check in the BlueField which representor ports are present. Representor ports should be named pf0.
   If the ports are not found, run the following:

```
$ mst start
$ mlxconfig -d /dev/mst/mt41682_pciconf0 s INTERNAL_CPU_MODEL=1
$ mlxconfig -d /dev/mst/mt41682_pciconf0 s ECPF_ESWITCH_MANAGER=1
ECPF_PAGE_SUPPLIER=0
```

4. Install missing packages.

   Some important packages do not come with BlueField operating systems pre-installed. Run the following command to install the packages:

```
$ yum install -y openvswitch
```

5. Install docker.

   OpenStack service runs as containers on BlueField. Run the following to install the docker:

```
$ yum-config-manager --enable extras
$ yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
$ yum install -y docker-ce docker-ce-cli containerd.io
$ usermod -aG docker $(whoami)
$ systemctl start docker.service
$ systemctl enable docker.service
```

4.3 Creating Neutron Agent Container on BlueField

To run OpenStack service as a container, the image and a starting script will be required.

1. Download container image at the following link: here and import the image to the docker repository:

```
docker load -i centos-binary-neutron-openvswitch-agent.tar
```

2. Start the script. This script is used to start the OpenStack service inside the container:

```
$ vi /root/neutron_ovs_agent_launcher.sh
```

and add the following lines:

```
#!/bin/bash
set -xe
/usr/bin/neutron-openvswitch-agent --config-file
/etc/neutron/neutron.conf --config-file
/etc/neutron/plugins/ml2/ml2_conf.ini --config-file
/etc/neutron/plugins/ml2/openvswitch_agent.ini --log-
file=/var/log/neutron/neutron.log
```

3. Create the container. Use the following script to create and start the container:

```
LOG_DIR_HOST=/var/log/neutron
CONF_DIR_HOST=/etc/neutron
IMAGE_ID=c47985e0fbad
CONTAINER_NAME=neutron_ovs_agent

# Create log folder and grant permissions
mkdir -p $LOG_DIR_HOST
chmod -R 755 $LOG_DIR_HOST

# Create container
docker container create \
--network host \
--privileged \
--name $CONTAINER_NAME \
--restart unless-stopped \
-v /run/openvswitch:/run/openvswitch/ \
-v $LOG_DIR_HOST:/var/log/neutron \
-v $CONF_DIR_HOST:/etc/neutron \
-v /root/neutron_ovs_agent_launcher.sh:/neutron_ovs_agent_launcher.sh \
$IMAGE_ID \
bash /neutron_ovs_agent_launcher.sh

# Start container
docker start $CONTAINER_NAME
```

4.4 Deployment of TripleO with Bare-Metal Service

Starting from a fresh RHEL 7.5 bare metal server, install and configure the undercloud according to the official TripleO installation documentation.

Follow this link for TripleO instructions to prepare bare metal overcloud.

4.5 Post-Deployment: Patch Neutron Server Container

After deployment, apply the Neutron Patch to the Neutron Server Container:

1. Get the patch file. Download the patch file from here and copy it to the Neutron server container.
2. Access the container bash shell.

```
$ docker exec -u root -it neutron_api bash
```

3. Install patch command.

```
$ yum install -y patch
```

4. Apply the patch.

```
$ cd /usr/lib/python2.7/site-packages/neutron/
$ patch -p2 agent/rpc.py ~/neutron.patch
$ patch -p2 objects/ports.py ~/neutron.patch
$ patch -p2 plugins/ml2/plugin.py ~/neutron.patch
$ patch -p2 plugins/ml2/rpc.py ~/neutron.patch
```

5. Install `pip` and `os-vif` package.

```
$ yum install -y python-pip
$ pip install os-vif
```

6. Restart the container.

```
$ docker restart neutron_api
```

4.6 BlueField Network Configuration

4.6.1 Network Configuration in BlueField

1. Set static IP to the BlueField from the overcloud external network subnet.
   Inside the BlueField, create network script in `/etc/sysconfig/network-scripts/` to create the external network interface, make sure the IP is free and the gateway is the controller IP.

```
vi /etc/sysconfig/network-scripts/ifcfg-enp3s0f1
# Generated by dracut initrd
NAME="enp3s0f1"
ONBOOT=yes
NETBOOT=yes
IPV6INIT=no
BOOTPROTO=static
IPADDR=192.168.24.111
NETMASK=255.255.255.0
GATEWAY=192.168.24.30
DEFROUTE=yes
DEVICE=enp3s0f1
TYPE=Ethernet
```

2. Add an external bridge.

   In BlueField, the operator must manually add the external bridge and its relevant interface.

```
$ ovs-vsctl add-br br-ext
$ ovs-vsctl add-port br-ext enp3s0f0
```

3. Get Neutron configuration to the BlueField.
   a. Copy Neutron configuration from the controller to the BlueField in the `/var/lib/config-data/puppet-generated/neutron/` directory.
   b. Copy the following files to BlueField:

```
etc/neutron/neutron.conf
etc/neutron/plugins/ml2/ml2_conf.ini
etc/neutron/plugins/ml2/openvswitch_agent.ini
```

4. Update Neutron configuration.

On the BlueField, changes must be made to set the correct values for the BlueField host.

a. In file `/etc/neutron/neutron.conf`, change the following:

```
bind_host=<Bluefield IP>
host=<Bluefield host>
```

b. In file `/etc/neutron/plugins/ml2/ml2_conf.ini`, change the following:

```
local_ip=<Bluefield IP>
firewall_driver=noop
```

5. Install the correct version of python-ironicclient-2.7.0 to support SmartNIC port-creation.

   On the undercloud, run:

```
sudo yum install python2-ironicclient-2.7.0
```

4.7 Add BlueField Ironic Images

1. Download the ironic images for BlueField from here.
2. Download the following three files:
   - ironic-deploy.kernel
   - ironic-deploy.initramfs
   - bm_centos.qcow2
3. Add the images to the overcloud.

```
openstack image create ironic-deploy-kernel --public --disk-format ari --
container-format ari --file . /ironic-deploy.kernel
openstack image create ironic-deploy-ram --public --disk-format ari --
container-format ari --file . /ironic-deploy.initramfs
openstack image create bm_centos --public --disk-format ari --container-
format ari --file . /bm_centos.qcow2
```

4.8 Create Overcloud Networks

A bare metal environment needs at least two networks: provisioning network and tenant network.

4. Create a provisioning network.

```
openstack network create --share --provider-network-type flat \
    --provider-physical-network datacentre --external provisioning
openstack subnet create --network provisioning \
    --subnet-range 192.168.24.0/24 --gateway 192.168.24.40 \
    --allocation-pool start=192.168.24.41,end=192.168.24.100
provisioning-subnet
```

5. Create a tenant network.

```
openstack network create tenant-net
openstack subnet create --network tenant-net --subnet-range 192.0.3.0/24
\
   --allocation-pool start=192.0.3.10,end=192.0.3.20 tenant-subnet
```

4.9 Bare-Metal Flavor

To create bare metal flavor, run the following command:

```
openstack flavor create --ram 1024 --disk 20 --vcpus 1 baremetal
openstack flavor set baremetal --property resources:CUSTOM_BAREMETAL=1
openstack flavor set baremetal --property resources:VCPU=0
openstack flavor set baremetal --property resources:MEMORY_MB=0
openstack flavor set baremetal --property resources:DISK_GB=0
```

4.10 Disable Automated Cleaning for Ironic

1. Login to the overcloud controller and edit `ironic.conf` file and set `automated_clean=False`
2. Restart ironic conductor container:

```
$ sudo docker restart ironic_conductor
```

4.11 Add Bare-Metal Node

Use the following script to add the bare metal nodes. Set the variables values (HOST_NAME, BM_NAME, IPMI_IP, BF_MAC):

```
#!/bin/bash -xe
. overcloudrc

export HOST_NAME=r-dcs81-005
export BM_NAME=r-dcs81-bf
export IPMI_IP=10.209.226.164
export BF_MAC=50:6b:4b:34:a5:3a
export KERNEL=$(glance image-list|grep ironic-deploy-kernel|awk '{print
$2}')
export RAM=$(glance image-list|grep ironic-deploy-ram|awk '{print $2}')

openstack baremetal node create --network-interface neutron --name $BM_NAME
--driver ipmi --driver-info ipmi_address=$IPMI_IP --driver-info
ipmi_password=ADMIN --driver-info ipmi_username=ADMIN --resource-class
baremetal --driver-info deploy_kernel=$KERNEL --driver-info
deploy_ramdisk=$RAM

#ironic node-update $BM_NAME replace boot_interface=ipxe
ironic node-update $BM_NAME replace deploy_interface=direct
ironic node-update $BM_NAME add properties/capabilities="boot_option:local"

openstack --os-baremetal-api-version 1.21 baremetal node set $BM_NAME --
resource-class baremetal
nova flavor-key baremetal set capabilities:boot_option="local" 2>&1|tee >
/dev/null
openstack flavor unset baremetal --property trait:CUSTOM_GOLD 2>&1|tee >
/dev/null

ironic node-update $BM_NAME add properties/memory_mb="65536"
ironic node-update $BM_NAME add properties/cpu_arch="x86_64"
ironic node-update $BM_NAME add properties/local_gb="371"
ironic node-update $BM_NAME add properties/cpus="24"
ironic node-update $BM_NAME add
properties/capabilities="cpu_hugepages:true,cpu_txt:true,boot_option:local,c
pu_aes:true,cpu_vt:true,cpu_hugepages_1g:true"

node_uuid=$(ironic node-list | grep $BM_NAME |awk '{print $2}')
```

```
openstack baremetal port create $BF_MAC --node $node_uuid --local-link-
connection hostname=$HOST_NAME --local-link-connection port_id="rep0-0" --
physical-network datacentre --pxe-enabled true --is-smartnic

ironic node-set-provision-state $BM_NAME manage
ironic node-set-provision-state $BM_NAME provide

openstack quota set --class --instances 60 default
openstack quota set --class --cores 60 default
```

4.12 Boot Bare Metal Instance

To boot the bare metal instance, do the following:

```
openstack server create --flavor baremetal --image bm_centos --nic net-
id=private r-bm-dcs81-bf
```

5 Configuring Mellanox SDN Mechanism Driver Plugin Using TripleO

5.1 Configure and Prepare the NEO

Use the links below, to execute the necessary preliminary steps to prepare the NEO machine:

1. NEO introduction/installation.
2. Configure NTP on NEO machine.

```
ssh root@<NEO_IP> ntpdate 0.asia.pool.ntp.org
```

5.2 Enable LLDP on Mellanox Switch

Enable LLDP on the Mellanox switch:

1. Login as admin.
2. Enter config mode.

```
switch > enable
switch # configure terminal
```

3. Enable LLDP globally on the switch.

```
switch (config) # lldp
```

5.3 Install RPM Package on Container Image

Before deploying the overcloud, create `/home/stack/containers-prepare-parameter.yaml` in order to modify the container image with required package python-networking-mlnx.

1. Download the package rpm file `python2-networking-mlnx` from this repo.
2. Move the package to `/home/stack/rpm_path/` folder.
3. Generate `/home/stack/containers-prepare-parameter.yaml`, if it does not yet exist, using this command:

```
(undercloud) [stack@undercloud ~]$openstack tripleo container image
prepare\
  default --local-push-destination \
  --output-env-file /home/stack/containers-prepare-parameter.yaml
```

4. Modify the `/home/stack/containers-prepare-parameter.yaml` file to install the package on the container image.

Example:

```
  parameter_defaults:
  DockerInsecureRegistryAddress:
  - 192.168.24.1:8787
  ContainerImagePrepare:
  - push_destination: true
    set:
      tag: "current-tripleo-rdo"
      namespace: "docker.io/tripleostein"
```

```
      name_prefix: "centos-binary-"
      name_suffix: ""
      neutron_driver: null

  - push_destination: true
    includes:
    - neutron-server
    modify_role: tripleo-modify-image
    modify_append_tag: "-updated"
    modify_vars:
      tasks_from: rpm_install.yml
      rpms_path: /home/stack/rpm_path/
```

Add the file `/home/stack/ containers-prepare-parameter.yaml` to the deploy command using -e parameter and deploy the overcloud.

5.4 Configure the Mellanox SDN Mechanism Driver Plugin

Modify the `file /usr/share/openstack-tripleo-heat-templates/environments/neutron-ml2-mlnx-sdn.yaml` in the following way:

```
# A Heat environment file which can be used to configure Mellanox SDN
resource_registry:
OS::TripleO::Services::NeutronCorePlugin:
OS::TripleO::Services::NeutronCorePluginMLNXSDN
parameter_defaults:
MlnxSDNUsername: '<sdn_username>'
MlnxSDNPassword: '<sdn_password>'
MlnxSDNUrl: '<sdn_url>'
MlnxSDNDomain: 'cloudx'
NeutronCorePlugin: 'neutron.plugins.ml2.plugin.Ml2Plugin'
NeutronMechanismDrivers: ['mlnx_sdn_assist','sriovnicswitch','openvswitch']
```

> The domain name 'cloudx' and the SDN credintials need to be changed as required.

Add the file `/usr/share/openstack-tripleo-heat-templates/environments/neutron-ml2-mlnx-sdn.yaml` to the deploy command using -e parameter and deploy the overcloud.

5.5 Set NTP Server

It is important that the time on the overcloud and NEO machine are synchronized.

In parameter defaults section, add the following line to set NTP server:

```
parameter_defaults:
  NtpServer: ['0.asia.pool.ntp.org', '1.asia.pool.ntp.org']
```

NEO machine and overcloud nodes must have the same NTP servers.

5.6 Install LLDPAD Package to Overcloud Image

LLDPAD package must be installed on all overcloud nodes by installing the package on the overcloud image before deploying:

```
virt-customize -v -a overcloud-full.qcow2 --run-command \
  "sudo subscription-manager register --username <USERNAME> \
   --password <PASSWORD> --auto-attach ; yum install lldpad -y ;\
   sudo subscription-manager unregister"
```

5.7 Configure LLDPAD in First Boot

a. Create file `first_boot.yaml` that contains the following content:

```
heat_template_version: rocky

description: >
  Start and configure LLDPAD

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: configure_lldp}

  configure_lldp:
    type: OS::Heat::SoftwareConfig
    properties:
      config: |
        #!/bin/bash
        set -eux
        set -o pipefail
        hostnamectl set-hostname $(hostname -s).localdomain
        hostnamectl set-hostname --transient $(hostname -s)
        sudo systemctl enable lldpad
        sudo systemctl start lldpad
        interface=p6p1
        lldptool set-lldp -i $interface adminStatus=rxtx
        lldptool -T -i $interface -V sysName enableTx=yes
        lldptool -T -i $interface -V portDesc enableTx=yes
        lldptool -T -i $interface -V sysDesc enableTx=yes
        lldptool -T -i $interface -V sysCap enableTx=yes
```

```
        lldptool -T -i $interface -V mngAddr enableTx=yes
        lldptool set-tlv -i $interface -V mngAddr ipv4=$(ip a sh br-ex |
grep " inet " | head -1 | awk '{print $2}' | cut -d'/' -f1);
outputs:
  OS::stack_id:
    value: {get_resource: userdata}
```

b. In your `network-environment.yaml` add the following line:

```
resource_registry:
  OS::TripleO::ComputeSriov::NodeUserData: ./first_boot.yaml
```