Mellanox Support for TripleO Rocky

Application Notes

Rev 1.1

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
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

List of Tables

Document Revision History

Table 1: Document Revision History

Revision	Date	Description
1.1	February 3, 2019	Added section Deploying TripleO with VF LAG Configuration.
1.0	August 29th, 2018	Initial version of this release.

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-3 Pro	ConnectX-3 Pro adapter cards with 10/40/56 Gigabit Ethernet connectivity with hardware offload engines to Overlay Networks ("Tunneling"), provide the highest performing and most flexible interconnect solution for PCI Express Gen3 servers used in public and private clouds, enterprise data centers, and high-performance computing.
ConnectX-4	ConnectX-4 adapter cards with Virtual Protocol Interconnect (VPI), supporting EDR 100Gb/s InfiniBand and 100Gb/s Ethernet connectivity, provide the highest performance and most flexible solution for high-performance, Web 2.0, Cloud, data analytics, database, and storage platforms.
ConenctX-4 Lx	ConnectX-4 Lx EN Network Controller with 1/10/25/40/50Gb/s Ethernet connectivity addresses virtualized infrastructure challenges, delivering best-in-class and highest performance to various demanding markets and applications. Providing true hardware-based I/O isolation with unmatched scalability and efficiency, achieving the most cost-effective and flexible solution for Web 2.0, Cloud, data analytics, database, and storage platforms.
ConnectX-5	supports two ports of 100Gb/s Ethernet connectivity, sub-700 nanosecond latency, and very high message rate, plus PCIe switch and NVMe over Fabric offloads, providing the highest performance and most flexible solution for the most demanding applications and markets. It Accelerated Switching and Packet Processing (ASAP2™) technology enhances offloading of virtual switches and virtual routers, for example, 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.
Virtual Function (VF)	A VF is virtual NIC that will be available for VMs on Compute nodes.
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.

Term	Description
OpenDayLight	OpenDayLight (ODL) is an open source project aimed to enhance Software Defined Networking with an Openstack Integration Plugin with the support of Network Function Virtualization NFV.
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 Direct	Mellanox Accelerated Switching and Packet Processing (ASAP2) Direct technology allows to offload OVS by handling OVS data-plane in Mellanox ConnectX-4 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 ASAP2 Direct include packet parsing and matching, forward, drop along with VLAN push/pop or VXLAN encapsulated/decapsulated.
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 (aka fabric). It depends on and requires the use of RDMA. NVMe over Fabrics can use any of the RDMA technologies, including InfiniBand, RoCE and iWARP.

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 datacentre scale.

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.

OpenDayLight (ODL) is an open source project aimed to enhance Software Defined Networking with an Openstack Integration Plugin with the support of Network Function Virtualization NFV.

Mellanox Accelerated Switching and Packet Processing (ASAP2) Direct technology allows to offload OVS by handling OVS data-plane in Mellanox ConnectX-4 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 ASAP2 Direct include packet parsing and matching, forward, drop along with VLAN push/pop or VXLAN encapsulated/decapsulated.

This Application Notes details how to enable the Mellanox “Accelerated Switching And Packet Processing” (ASAP^2) Direct technology feature of Hardware Off-loading support over Open vSwitch (OVS) and ODL in TripleO setup for both VLAN and VXLAN networks for both containerized and non-containerized architectures.

1.1 Supported Features

TripleO Rocky supports the following Features:

- ASAP^2 Direct support:
  - over Open vSwitch
  - over Opendaylight
- OVS over DPDK with Inbox Driver
- NVMe over Fabric (NVMeOF)

1.2 System Requirements

The system requirements are detailed in the following tables:

Table 3: Undercloud Node Requirements

Platform	Type and Version
OS	Red Hat Enterprise Linux 7.5
CPU	An 8-core 64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions.
Memory	A minimum of 16 GB of RAM.
Disk Space	A minimum of 40 GB of available disk space on the root disk. Make sure to leave at least 10 GB free space before attempting an Overcloud deployment or update. This free space accommodates image conversion and caching during the node provisioning process.
Networking	A minimum of 2 x 1 Gbps Network Interface Cards. However, it is recommended to use a 10 Gbps interface for Provisioning network traffic, especially if provisioning many nodes in your Overcloud environment. We need to use Mellanox NIC for tenant network.

1.3 Supported Network Adapter Cards and Firmware

Mellanox support for TripleO Rocky supports the following Mellanox network adapter cards and their corresponding firmware versions:

NICs	Supported Protocols	Supported Link Speeds	Recommended Firmware Rev.
ConnectX®-3 Pro	Ethernet	10, 25, 40 and 50Gb/s	2.42.5000
ConnectX®-4	Ethernet	10, 25, 40, 50 and 100 Gb/s	12.21.2030
ConnectX®-4 Lx	Ethernet	10, 25, 40 and 50Gb/s	14.21.2030
ConnectX®-5	Ethernet	10, 25, 40, 50 and 100 Gb/s	16.21.2030

1.4 Supported Operating Systems

The following are the supported OSes:

Table 4: Supported Operating Systems

OS	Platform
RHEL7.5	x86_64

1.5 Overcloud Operating System Versions

Table 5: Overcloud Operating System Versions

Item	Version
Kernel	kernel-3.10.0-860.el7.x86_64.rpm kernel-headers-3.10.0-860.el7.x86_64.rpm kernel-tools-3.10.0-860.el7.x86_64.rpm kernel-tools-libs-3.10.0-860.el7.x86_64.rpm
Iproute	iproute-4.11.0-13.el7.x86_64.rpm
Open vSwitch	openvswitch-2.9.0-9.el7fdn.x86_64.rpm
OpenDayLight	opendaylight-8.1.0-0.1.20180417snap64.el7.noarch.rpm
linux-firmware	linux-firmware-20171127-58.git17e6288.el7.noarch.rpm
libib	libibcm-15-4.el7.x86_64.rpm libibumad-15-4.el7.x86_64.rpm libibverbs-15-4.el7.x86_64.rpm libibverbs-utils-15-4.el7.x86_64.rpm librdmacm-15-4.el7.x86_64.rpm librdmacm-utils-15-4.el7.x86_64.rpm
Iwpmd	iwpmd-15-4.el7.x86_64.rpm
ibacm	ibacm-15-4.el7.x86_64.rpm
dpdk	dpdk-tools-17.11-3.el7fdb.x86_64 dpdk-17.11-3.el7fdb.x86_64

2 ASAP² Direct support

2.1 ASAP² Direct support over Open vSwitch

2.1.1 Network Cards Support Matrix and Limitations

Mellanox cards support ASAP² HW offloading feature as in the following table:

NICs	Supported Protocols	Supported Network Type	ASAP² Direct RDMA support
ConnectX®-4	Ethernet	Support HW-offloading over VLAN only	Only VLAN
ConnectX®-4 Lx	Ethernet	Support HW-offloading over VLAN and VXLAN.	No RDMA support.
ConnectX®-5	Ethernet		RDMA is supported over VLAN and VXLAN

2.1.2 Configuration

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

1. Update the ovs-hw-offload.yaml to identify the interface that has the VFs.

```
enviroments/ovs-hw-offload.yaml
```

You can configure it over VLAN/VXLAN setup as follow:

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

```
# 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

resource_registry:
  OS::TripleO::Services::NeutronSriovHostConfig:
../puppet/services/neutron-sriov-host-config.yaml

parameter_defaults:
  NeutronFlatNetworks: datacentre
  NeutronNetworkType:
  - vlan
  NeutronTunnelTypes: ''
  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','C
omputeCapabilitiesFilter','ImagePropertiesFilter','ServerGroupAntiAffi
nityFilter','ServerGroupAffinityFilter','PciPassthroughFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_pass
through_filter.PciPassthroughFilter"]

  # Kernel arguments for ComputeSriov node
  ComputeSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt"
    NeutronBridgeMappings:
    - datacentre:br-ex
    OvsHwOffload: True
    # Number of VFs that needs to be configured for a physical
interface
    NeutronSriovNumVFs:
    - <interface_name>:<number_of_vfs>:switchdev
```

```
    # 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.
    NovaPCIPassthrough:
    - devname: <interface_name>
      physical_network: datacenter
```

Please note that you need to change the **<interface_name>** and **<number_of_vfs>** in the file according to your setup.

- In the case of a **VXLAN** setup, you need to:

  i. Configure the ovs-hw-offload.yaml as follows:

```
# 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

resource_registry:
  OS::TripleO::Services::NeutronSriovHostConfig:
../puppet/services/neutron-sriov-host-config.yaml

parameter_defaults:

  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter',
'ComputeCapabilitiesFilter','ImagePropertiesFilter','ServerGroupAnti
AffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_pa
ssthrough_filter.PciPassthroughFilter"]

  # Kernel arguments for ComputeSriov node
  ComputeSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt"
    OvsHwOffload: True
    # Number of VFs that needs to be configured for a physical
interface
    #NeutronSriovNumVFs: ["ens3f0:4:switchdev"]
    # 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.
    NeutronSriovNumVFs:
    - <interface_name>:<number_of_vfs>:switchdev
    NovaPCIPassthrough:
    - devname: <interface_name>
      physical_network: null
```

  ii. Configure the interface names in the `/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans/compute.yaml` and `/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
```

The Tenant netwrok should be moved from the VLAN on a bridge to be on a seperated interface due to a driver limitation when using ASAP^2 Direct HW offloading as the network traffic is not offloaded when using tunnel IP on the OVS internal port.
For further information, see Known Isssue 1327510 in the Known Issues document.

Please note that you need to change the **<interface_name>** and **<number_of_vfs>** in the file according to your setup.

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 example below:

```
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

In the case of Bare-metal, there is no need to run overcloud-prep-containers.sh

2.1.3 Deploying the Overcloud

Deploy 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 \
  -e ~/heat-templates/environments/host-config-and-reboot.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
```

- If you want to deploy using baremetal: remove the following lines from the deploy command:0

```
  -e /home/stack/containers-default-parameters.yaml  \
  -e ~/heat-templates/environments/docker.yaml \
```

- And add the following file:

```
  -e ~/heat-templates/environments/baremetal-services.yaml
```

2.1.4 Booting the VM

On the Undercloud machine:

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 as follows:

```
$ 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 ASAP² Direct support over Opendaylight[1]

2.2.1 Network Cards Support Matrix and Limitations

Mellanox cards support ASAP² HW offloading feature as in the following table:

NICs	Supported Protocols	Supported Network Type	ASAP² Direct RDMA support
ConnectX®-4	Ethernet	Support HW-offloading over VLAN only	Only VLAN
ConnectX®-4 Lx	Ethernet	Support HW-offloading over VLAN and VXLAN.	No RDMA support.
ConnectX®-5	Ethernet		RDMA is supported over VLAN and VXLAN

2.2.2 Configuration

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

1. Update enviroments/ovs-hw-offload.yaml to identify the interface that has the VFs.

   You can configure it over VLAN/VXLAN setup as follow:

   - In the case of a **VLAN** setup, configure the neutron-opendaylight.yaml and ovs-hw-offload.yaml as follows:

```
# 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

resource_registry:
  OS::TripleO::Services::NeutronSriovHostConfig:
../puppet/services/neutron-sriov-host-config.yaml

parameter_defaults:
  NeutronFlatNetworks: datacentre
  NeutronNetworkType:
  - vlan
  NeutronTunnelTypes: ''
  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','C
omputeCapabilitiesFilter','ImagePropertiesFilter','ServerGroupAntiAffi
nityFilter','ServerGroupAffinityFilter','PciPassthroughFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_pass
through_filter.PciPassthroughFilter"]

  # Kernel arguments for ComputeSriov node
  ComputeSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt"
    NeutronBridgeMappings:
    - datacentre:br-ex
    OvsHwOffload: True
    # Number of VFs that needs to be configured for a physical
interface
```

[1] ASAP2 Direct is supported in the header re-write with ODL but not supported in OVS due to the OVS mechanism driver lack of support of Layer 3 switching.

```
    NeutronSriovNumVFs:
    - <interface_name>:<number_of_vfs>:switchdev
    # 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.
    NovaPCIPassthrough:
    - devname: <interface_name>
      physical_network: datacenter
```

Please note that you need to change the **<interface_name>** and **<number_of_vfs>** in the file according to your setup.

- In the case of a **VXLAN** setup, you need to:

  i. Configure the ovs-hw-offload.yaml as follows:

```
# 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

resource_registry:
  OS::TripleO::Services::NeutronSriovHostConfig:
../puppet/services/neutron-sriov-host-config.yaml

parameter_defaults:

  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter',
'ComputeCapabilitiesFilter','ImagePropertiesFilter','ServerGroupAnti
AffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_pa
ssthrough_filter.PciPassthroughFilter"]

  # Kernel arguments for ComputeSriov node
  ComputeSriovParameters:
    KernelArgs: "intel_iommu=on iommu=pt"
    OvsHwOffload: True
    # Number of VFs that needs to be configured for a physical
interface
    #NeutronSriovNumVFs: ["ens3f0:4:switchdev"]
    # 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.
    NeutronSriovNumVFs:
    - <interface_name>:<number_of_vfs>:switchdev
    NovaPCIPassthrough:
    - devname: <interface_name>
      physical_network: null
```

Please note that you need to change the **<interface_name>** and **<number_of_vfs>** in the file according to your setup.

  ii. Configure the interface names in the `/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans/compute.yaml` and `/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 in a bridge to be in separated interface.

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

The Tenant netwrok should be moved from the VLAN on a bridge to be on a seperated interface due to a driver limitation when using

ASAP^2 Direct HW offloading as the network traffic is not offloaded when using tunnel IP on the OVS internal port.

Please note that you need to change the **<interface_name>** and **<number_of_vfs>** in the file according to your setup.

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. Add/update to the `~/cloud-names.yaml` accordingly the following lines:

```
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. Add the ODL packages to the Overcloud image in the case of Baremetal deployment,:

```
LIBGUESTFS_BACKEND=direct virt-customize --upload opendaylight-
<version_no>.rpm:/root/ --run-command "yum -y install /root/*.rpm" -a
overcloud-full.qcow2
```

Please note the **<version_no>** we tested for OpenDayLight is ***opendaylight-8.1.0-0.1.20180417snap64.el7.noarch.rpm*** which is built from source code.

6. Add this parameter to PREPARE_ARGS variable in overcloud-prep-containers.sh

```
“-e /usr/share/openstack-tripleo-heat-templates/environments/services-
docker/neutron-opendaylight.yaml”
```

7. Run overcloud-prep-containers.sh

In the case of Bare-metal, there is no need to run overcloud-prep-containers.sh

2.2.3 Deploying the Overcloud

Deploy 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 environments/docker.yaml
  -e enviroments/services-docker/neutron-opendaylight.yaml
  -e enviroments/ovs-hw-offload.yaml
  -e ~/heat-templates/environments/host-config-and-reboot.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
```

- If you want to deploy using baremetal, remove the following lines from the deploy command.

```
  -e /home/stack/containers-default-parameters.yaml  \
  -e ~/heat-templates/environments/docker.yaml \
  -e enviroments/services-docker/neutron-opendaylight.yaml
```

- And add the following file:

```
  -e ~/heat-templates/environments/baremetal-services.yaml
  -e enviroments/neutron-opendaylight.yaml
```

2.2.4 Booting the VM

On the Undercloud machine:

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 as follows:

```
$ 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 --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
```

2.3 Checking Hardware Offloading

To check whether or no hardware offloading is working, you need to create 2 VMs, one on each compute node as described below and then using tcpdump on the representor port on the compute node to see if only 2 ICMP packets exist.

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

```
$ 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 2 hypervisors in same network.

   - On the first VM, run the ping command “ping <second_vm_ip_address>”. In the following we will use 11.11.11.9 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
```

   - 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
```

- Check 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
```

2.4 Verifying Hardware Offloading Configuration (Troubleshooting HW Offloading Configuration)

- Check that hw-offload is configured on the compute.

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

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

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

- For ConectX-4/ConnectX-4 Lx card:

```
# devlink dev eswitch show pci/0000:03:00.0
pci/0000:03:00.0: mode switchdev inline-mode transport encap enable
```

- 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]
```

- 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]
```

- 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
```

- 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>
```

2.5 Deploying TripleO with VF LAG Configuration

Please note that 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: interface
                name: enp3s0f0
                primary: true
              - type: interface
                name: enp3s0f1
```

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"
```

Please note, the supported bonding mode for vf-lag are:

- Active-Backup
- Active-Active
- LACP

➢ ***To create the uplink over a vlan number 77 over a bond for example, you can use:***

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

or this one for general interface:

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

3 OVS-DPDK

3.1 Network Cards Support Matrix and Limitations

Mellanox cards support OVS-DPDK feature as in the following table:

NICs	Supported Protocols	Supported Network Type
ConnectX®-3 Pro	Ethernet	User is required to use first boot file as explain in the configuration section below.
ConnectX®-4	Ethernet	
ConnectX®-4 Lx	Ethernet	
ConnectX®-5	Ethernet	

3.2 Configuration

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

To configure OVS-DPDK follow the instructions available at Deploying with OVS DPDK Support.

1. Create an Env file `dpdk.yaml` for example that contain the following configurations:

```
resource_registry:
  OS::TripleO::ComputeOvsDPDK::Net::SoftwareConfig: ./compute-dpdk.yaml

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

2. Network Config: DPDK supported network interfaces should be specified in the network config templates to configure OVS DPDK on the node. The following new network config types have been added in the compute-dpdk.yaml to support DPDK.
   - ovs_user_bridge
   - ovs_dpdk_port
   - ovs_dpdk_bond
3. As an example:

```
members:
 - type: ovs_dpdk_port
   name: dpdk0
   members:
   - type: interface
     name: enp3s0f0
   driver: mlx5_core
```

Where driver is mlx5_core for ConnectX-4 and ConnectX-5 and mlx4_core for ConnectX-3. Also note that in the case of ConnectX-3 you need to specify the resource registry in the file compute-dpdk.yaml:

```
OS::TripleO::ComputeOvsDpdk::NodeUserData:../firstboot/conntectx3_streering.
yaml
```

4. Create a new role for the compute node and change it to ComputeOvsDpdk.

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

5. Add/update to the `~/cloud-names.yaml` accordingly the following lines:

```
parameter_defaults:
ComputeOvsDpdkCount: 2
OvercloudComputeOvsDpdkFlavor: compute
```

6. Run overcloud-prep-containers.sh[2]

3.3 DPDK bonding:

In the case of DPDK bonding, you need to update compute-dpdk.yaml to contain the following configuration:

```
            - type: ovs_user_bridge
              name: br-mlnx
              use_dhcp: false
              addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet
              members:
                - type: ovs_dpdk_bond
                  name: dpdkbond0
                  members:
                    - type: ovs_dpdk_port
                      name: dpdk0
                      driver: mlx5_core
                      members:
                        - type: interface
                          name: nic1
                    - type: ovs_dpdk_port
                      name: dpdk1
                      driver: mlx5_core
                      members:
                        - type: interface
                          name: nic2
```

3.4 NUMA Configuration

Specify the server core that has the same NUMA as the Mellanox NIC in the `neutron-ovs-dpdk.yaml` file:

1. Find out what is the NUMA node of the NIC.

```
#/sys/class/net/ens2f0/device/numa_node
```

2. Find out what is the NUMA node per core.

```
#lscpu
```

3. Edit the configuration file respectively. Split the core list (example 1-9) to separate cores for ovs-dpdk and VMs:

   (If NUMA 1 is used, change the OvsDpdkSocketMemory to "1024,0")

```
#vi ~/environments/services-docker/neutron-ovs-dpdk.yaml
```

[2] The "overcloud-prep-containers.sh" script is available by default in TripleO undercloud and can be downloaded from: tripleo-quickstart-extras.

In the case of Bare-metal, the edit the ~/environments/neutron-ovs-dpdk.yaml file instead.

4. Add the following lines.

```
    IsolCpusList: "1,2,3,4,5,6,7,8,9"
    KernelArgs: "default_hugepagesz=1G hugepagesz=1G hugepages=12
iommu=pt intel_iommu=on"
    OvsDpdkSocketMemory: "1024,0"
    OvsPmdCoreList: "1,2,3,4"
    NovaVcpuPinSet: "5,6,7,8,9"
```

5. Update the `~/cloud-names.yaml` accordingly.

```
parameter_defaults:
ComputeOvsDpdkCount: 2
OvercloudComputeSriovFlavor: compute
```

3.5 Deploying the OVS-DPDK Overcloud

Deploy 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\
-r roles_data_dpdk.yaml \
--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 180  \
-e /home/stack/cloud-names.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 /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 /home/stack/containers-default-parameters.yaml  \
-e ~/heat-templates/environments/docker.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/disable-
telemetry.yaml \
--validation-warnings-fatal  \
--ntp-server pool.ntp.org \
-e ~/services-docker/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/host-config-and-
reboot.yaml \
-e ~/nic_configs/network-dpdk.yaml  \
-e /usr/share/openstack-tripleo-heat-templates/environments/ovs-dpdk-
permissions.yaml \
dpdk.yaml
```

- If you want to deploy using baremetal: remove the following lines from the deploy command:0

```
  -e /home/stack/containers-default-parameters.yaml  \
  -e ~/heat-templates/environments/docker.yaml \
```

- And add the following file:

```
-e ~/heat-templates/environments/baremetal-services.yaml
-e enviroments/neutron-ovs-dpdk.yaml
```

3.6 Booting the VM

On the Undercloud machine:

1. Load the `overcloudrc` configuration.

```
# source overcloudrc
```

2. Create a flavor.

```
# openstack flavor create m1.large --id 5 --ram 2048 --disk 20 --vcpus 1
```

3. Run the following command.

```
# openstack flavor set m1.large --property hw:mem_page_size=large
```

4. Create “cirrios” image.

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

5. 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
```

6. Create subnet as follows:

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

7. Boot a VM on the Overcloud using the following command:

   - For the first VM:

```
$ openstack server create --nic net-id=private --flavor 5 --image mellanox vm1
```

   - For the Second VM

```
$ openstack server create --nic net-id=private --flavor 3 --image mellanox vm2
```

4 NVMe over Fabrics (NVMe-oF)

4.1 Network Cards Support Matrix and Limitations

Mellanox cards support NVMe-oF feature as in the following table:

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

4.2 Deployment of Non-Containerized Overcloud

4.2.1 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 NVMeOF backend parameters.

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

4. Prepare your deployment files as you need, 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
```

4.2.2 Deploying the NVMEoF Overcloud

Deploy 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
    --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 /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
```

4.3 Deployment of Containerized Overcloud

4.3.1 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 NVMeOF backend parameters.

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

a. Prepare your deployment files as you need, 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
```

4.3.2 Deploying the NVMEoF Overcloud

Deploy 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 \
```