Red Hat OpenStack Platform 13

Network Functions Virtualization Planning and Configuration Guide

Planning and Configuring the Network Functions Virtualization (NFV) OpenStack Deployment

Last Updated: 2024-03-21

Red Hat OpenStack Platform 13 Network Functions Virtualization Planning and Configuration Guide

Planning and Configuring the Network Functions Virtualization (NFV) OpenStack Deployment

OpenStack Team
rhos-docs@redhat.com

Abstract

This guide contains important planning information and describes the configuration procedures for single root input/output virtualization (SR-IOV) and dataplane development kit (DPDK) for network functions virtualization infrastructure (NFVi) in your Red Hat OpenStack Platform deployment.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message .

CHAPTER 1. OVERVIEW OF NETWORK FUNCTIONS VIRTUALIZATION

Network Functions Virtualization (NFV) is a software solution that virtualizes a network function on general purpose, cloud based infrastructure. With NFV, the Communication Service Provider is able to move away from traditional hardware.

For a high level overview of NFV concepts, see the Network Functions Virtualization Product Guide .

NOTE

OVS-DPDK and SR-IOV configuration depends on your hardware and topology. This guide provides examples for CPU assignments, memory allocation, and NIC configurations that might vary from your topology and use case.

Use Red Hat OpenStack Platform director to isolate specific network types, for example, external, project, internal API, and so on. You can deploy a network on a single network interface, or distributed over a multiple-host network interface. With Open vSwitch you can create bonds by assigning multiple interfaces to a single bridge. Configure network isolation in a Red Hat OpenStack Platform installation with template files. If you do not provide template files, the service networks deploy on the provisioning network. There are two types of template configuration files:

network-environment.yaml

This file contains network details, such as subnets and IP address ranges, for the overcloud nodes. This file also contains the different settings that override the default parameter values for various scenarios.

compute.yaml and **controller.yaml**

These files contain the host network interface configuration for the overcloud nodes.

host-config-and-reboot.yaml

This file replaces the deprecated **first-boot.yaml** file, and contains configuration for host installation.

These heat template files are located at **/usr/share/openstack-tripleo-heat-templates/** on the undercloud node.

The Hardware requirements and Software requirements sections provide more details on how to plan and configure the heat template files for NFV using the Red Hat OpenStack Platform director.

NOTE

You use YAML files to outline NFV configuration. For more information about the YAML file format, see YAML in a Nutshell

CHAPTER 2. HARDWARE REQUIREMENTS

This section describes the hardware details necessary for NFV.

You can use Red Hat Technologies Ecosystem to check for a list of certified hardware, software, cloud provider, component by choosing the category and then selecting the product version.

For a complete list of the certified hardware for Red Hat OpenStack Platform, see Red Hat OpenStack Platform certified hardware.

2.1. NETWORK ADAPTER SUPPORT

For a list of tested NICs for NFV, log in to the Network Adapter Support page of the Customer Portal.

If you configure OVS-DPDK on Mellanox ConnectX-4 or ConnectX-5 network interfaces, you must set the corresponding kernel driver in the **compute-ovs-dpdk.yaml** file:

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

2.2. DISCOVERING YOUR NUMA NODE TOPOLOGY

When you plan your deployment, you must understand the NUMA topology of your Compute node so that you can partition the CPU and memory resources for optimum performance. To determine the NUMA information, you can choose one of the following options:

- Enable hardware introspection to retrieve NUMA information from bare-metal nodes.
- Log in to each bare-metal node to manually collect the information.

NOTE

You must install and configure the undercloud before you can retrieve NUMA information through hardware introspection. For more information, see the *Director Installation and Usage* guide.

Retrieving Hardware Introspection Details

The Bare Metal service hardware inspection extras (**inspection_extras**) is enabled by default to retrieve hardware details. You can use these hardware details to configure your overcloud. For more information about the **inspection_extras** parameter in the **undercloud.conf** file, see Configuring the Director in the *Director Installation and Usage* guide.

For example, the **numa_topology** collector is part of these hardware inspection extras and includes the following information for each NUMA node:

- RAM (in kilobytes)
- Physical CPU cores and their sibling threads

- NICs associated with the NUMA node

Use the **openstack baremetal introspection data save _UUID_ | jq .numa_topology** command to retrieve this information, with the *UUID* of the bare-metal node.

The following example shows the retrieved NUMA information for a bare-metal node:

```
{
  "cpus": [
    {
      "cpu": 1,
      "thread_siblings": [
        1,
        17
      ],
      "numa_node": 0
    },
    {
      "cpu": 2,
      "thread_siblings": [
        10,
        26
      ],
      "numa_node": 1
    },
    {
      "cpu": 0,
      "thread_siblings": [
        0,
        16
      ],
      "numa_node": 0
    },
    {
      "cpu": 5,
      "thread_siblings": [
        13,
        29
      ],
      "numa_node": 1
    },
    {
      "cpu": 7,
      "thread_siblings": [
        15,
        31
      ],
      "numa_node": 1
    },
    {
      "cpu": 7,
      "thread_siblings": [
        7,
        23
      ],
      "numa_node": 0
```

```
},
{
  "cpu": 1,
  "thread_siblings": [
    9,
    25
  ],
  "numa_node": 1
},
{
  "cpu": 6,
  "thread_siblings": [
    6,
    22
  ],
  "numa_node": 0
},
{
  "cpu": 3,
  "thread_siblings": [
    11,
    27
  ],
  "numa_node": 1
},
{
  "cpu": 5,
  "thread_siblings": [
    5,
    21
  ],
  "numa_node": 0
},
{
  "cpu": 4,
  "thread_siblings": [
    12,
    28
  ],
  "numa_node": 1
},
{
  "cpu": 4,
  "thread_siblings": [
    4,
    20
  ],
  "numa_node": 0
},
{
  "cpu": 0,
  "thread_siblings": [
    8,
    24
  ],
  "numa_node": 1
```

```
    },
    {
      "cpu": 6,
      "thread_siblings": [
        14,
        30
      ],
      "numa_node": 1
    },
    {
      "cpu": 3,
      "thread_siblings": [
        3,
        19
      ],
      "numa_node": 0
    },
    {
      "cpu": 2,
      "thread_siblings": [
        2,
        18
      ],
      "numa_node": 0
    }
  ],
  "ram": [
    {
      "size_kb": 66980172,
      "numa_node": 0
    },
    {
      "size_kb": 67108864,
      "numa_node": 1
    }
  ],
  "nics": [
    {
      "name": "ens3f1",
      "numa_node": 1
    },
    {
      "name": "ens3f0",
      "numa_node": 1
    },
    {
      "name": "ens2f0",
      "numa_node": 0
    },
    {
      "name": "ens2f1",
      "numa_node": 0
    },
    {
      "name": "ens1f1",
      "numa_node": 0
    },
    {
      "name": "ens1f0",
      "numa_node": 0
    },
    {
      "name": "eno4",
      "numa_node": 0
    },
    {
      "name": "eno1",
      "numa_node": 0
    },
    {
      "name": "eno3",
      "numa_node": 0
    },
    {
      "name": "eno2",
      "numa_node": 0
    }
  ]
}
```

2.3. REVIEW BIOS SETTINGS

The following listing describes the required BIOS settings for NFV:

- **C3 Power State**: Disabled.
- **C6 Power State**: Disabled.
- **MLC Streamer**: Enabled.
- **MLC Spacial Prefetcher**: Enabled.
- **DCU Data Prefetcher**: Enabled.
- **DCA**: Enabled.
- **CPU Power and Performance**: Performance.
- **Memory RAS and Performance Config → NUMA Optimized**: Enabled.
- **Turbo Boost**: Disabled.
- **VT-d**: Enabled for Intel cards if VFIO functionality is needed.

2.4. NETWORK ADAPTER FAST DATAPATH FEATURE SUPPORT MATRIX

For a list of supported versions of FDP, log in to the Overview of Fast Datapath page of the Customer Portal.

CHAPTER 3. SOFTWARE REQUIREMENTS

This section describes the supported configurations and drivers, and subscription details necessary for NFV.

3.1. REGISTERING AND ENABLING REPOSITORIES

To install Red Hat OpenStack Platform, you must register Red Hat OpenStack Platform director using the Red Hat Subscription Manager, and subscribe to the required channels. See Registering your system for details.

Procedure

1. Disable the default repositories.

```
subscription-manager repos --disable=*
```

2. Enable required repositories for Red Hat OpenStack Platform with network functions virtualization (NFV).

```
sudo subscription-manager repos \
--enable=rhel-7-server-rpms \
--enable=rhel-7-server-extras-rpms \
--enable=rhel-7-server-rh-common-rpms \
--enable=rhel-ha-for-rhel-7-server-rpms \
--enable=rhel-7-server-openstack-13-rpms \
--enable=rhel-7-server-nfv-rpms
```

NOTE

To register your overcloud nodes, see Overcloud Registration.

3.2. SUPPORTED CONFIGURATIONS FOR NFV DEPLOYMENTS

Red Hat OpenStack Platform supports the following network functions virtualization (NFV) deployments using director:

- Single root I/O virtualization (SR-IOV)
- Open vSwitch with Data Plane Development Kit (OVS-DPDK)

Additionally, you can deploy Red Hat OpenStack Platform with any of the following features:

- Composable roles
- Hyper-converged infrastructure (limited support)
- Configuring real-time compute
- OVS hardware offload (Technology preview)

NOTE

Red Hat’s embedded OpenDaylight SDN solution is being deprecated in OpenStack Platform (OSP) 14. Red Hat will continue to provide support and bug fixes for OpenDaylight, with all support ending with the OSP 13 lifecycle (June 27, 2021).

3.3. SUPPORTED DRIVERS

For a complete list of supported drivers, see Component, Plug-In, and Driver Support in Red Hat OpenStack Platform .

For a list of NICs tested for NFV deployments with Red Hat OpenStack, see Tested NICs.

3.4. COMPATIBILITY WITH THIRD PARTY SOFTWARE

For a complete list of products and services tested, supported, and certified to perform with Red Hat technologies (Red Hat OpenStack Platform), see Third Party Software compatible with Red Hat OpenStack Platform. You can filter the list by product version and software category.

For a complete list of products and services tested, supported, and certified to perform with Red Hat technologies (Red Hat Enterprise Linux), see Third Party Software compatible with Red Hat Enterprise Linux. You can filter the list by product version and software category.

CHAPTER 4. NETWORK CONSIDERATIONS

The undercloud host requires at least the following networks:

- Provisioning network - Provides DHCP and PXE boot functions to help discover bare-metal systems for use in the overcloud.
- External network - A separate network for remote connectivity to all nodes. The interface connecting to this network requires a routable IP address, either defined statically, or dynamically through an external DHCP service.

The minimal overcloud network configuration includes:

- Single NIC configuration - One NIC for the Provisioning network on the native VLAN and tagged VLANs that use subnets for the different overcloud network types.
- Dual NIC configuration - One NIC for the Provisioning network and the other NIC for the External network.
- Dual NIC configuration - One NIC for the Provisioning network on the native VLAN and the other NIC for tagged VLANs that use subnets for the different overcloud network types.
- Multiple NIC configuration - Each NIC uses a subnet for a different overcloud network type.

For more information on the networking requirements, see Networking requirements.

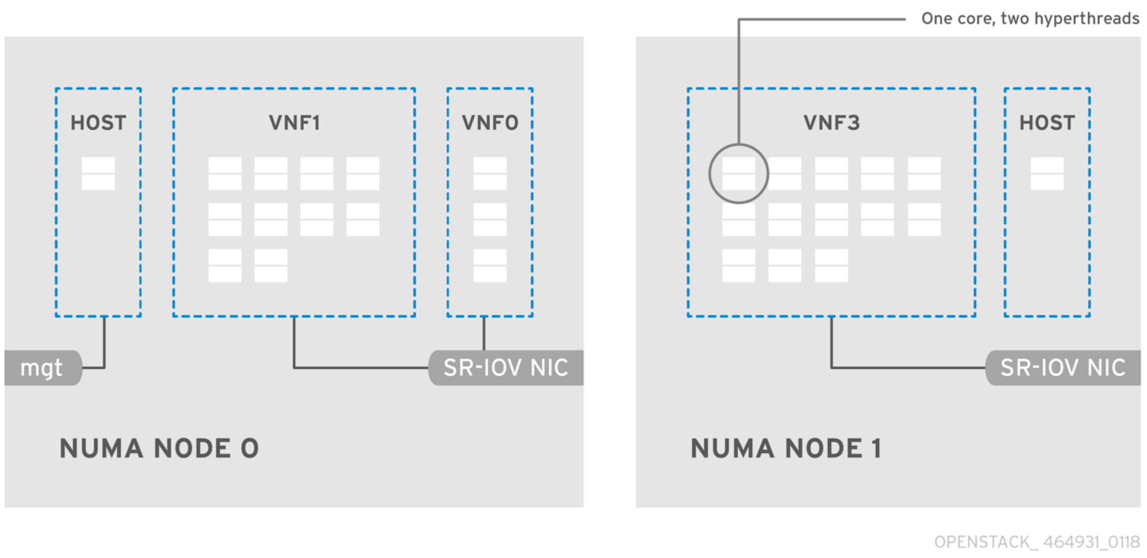
CHAPTER 5. PLANNING AN SR-IOV DEPLOYMENT

Optimize single root I/O virtualization (SR-IOV) deployments for NFV by setting individual parameters based on your Compute node hardware.

See Discovering your NUMA node topology to evaluate your hardware impact on the SR-IOV parameters.

5.1. HARDWARE PARTITIONING FOR AN SR-IOV DEPLOYMENT

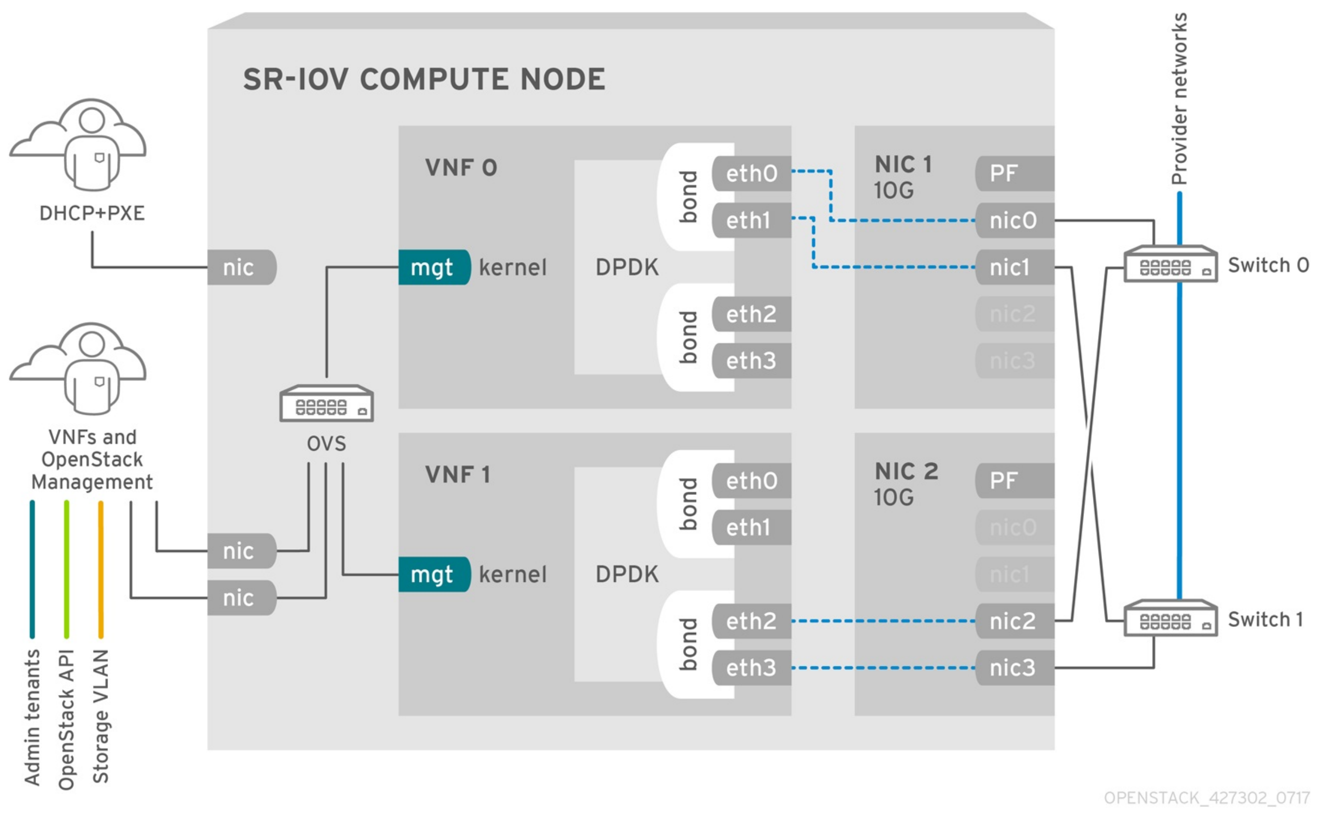
To achieve high performance with SR-IOV, you need to partition the resources between the host and the guest.



A typical topology includes 14 cores per NUMA node on dual socket Compute nodes. Both hyper-threading (HT) and non-HT cores are supported. Each core has two sibling threads. One core is dedicated to the host on each NUMA node. The VNF handles the SR-IOV interface bonding. All the interrupt requests (IRQs) are routed on the host cores. The VNF cores are dedicated to the VNFs. They provide isolation from other VNFs as well as isolation from the host. Each VNF must use resources on a single NUMA node. The SR-IOV NICs used by the VNF must also be associated with that same NUMA node. This topology does not have a virtualization overhead. The host, OpenStack Networking (neutron) and Compute (nova) configuration parameters are exposed in a single file for ease, consistency and to avoid incoherence that is fatal to proper isolation, causing preemption and packet loss. The host and virtual machine isolation depend on a **tuned** profile, which takes care of the boot parameters and any OpenStack modifications based on the list of CPUs to isolate.

5.2. TOPOLOGY OF AN NFV SR-IOV DEPLOYMENT

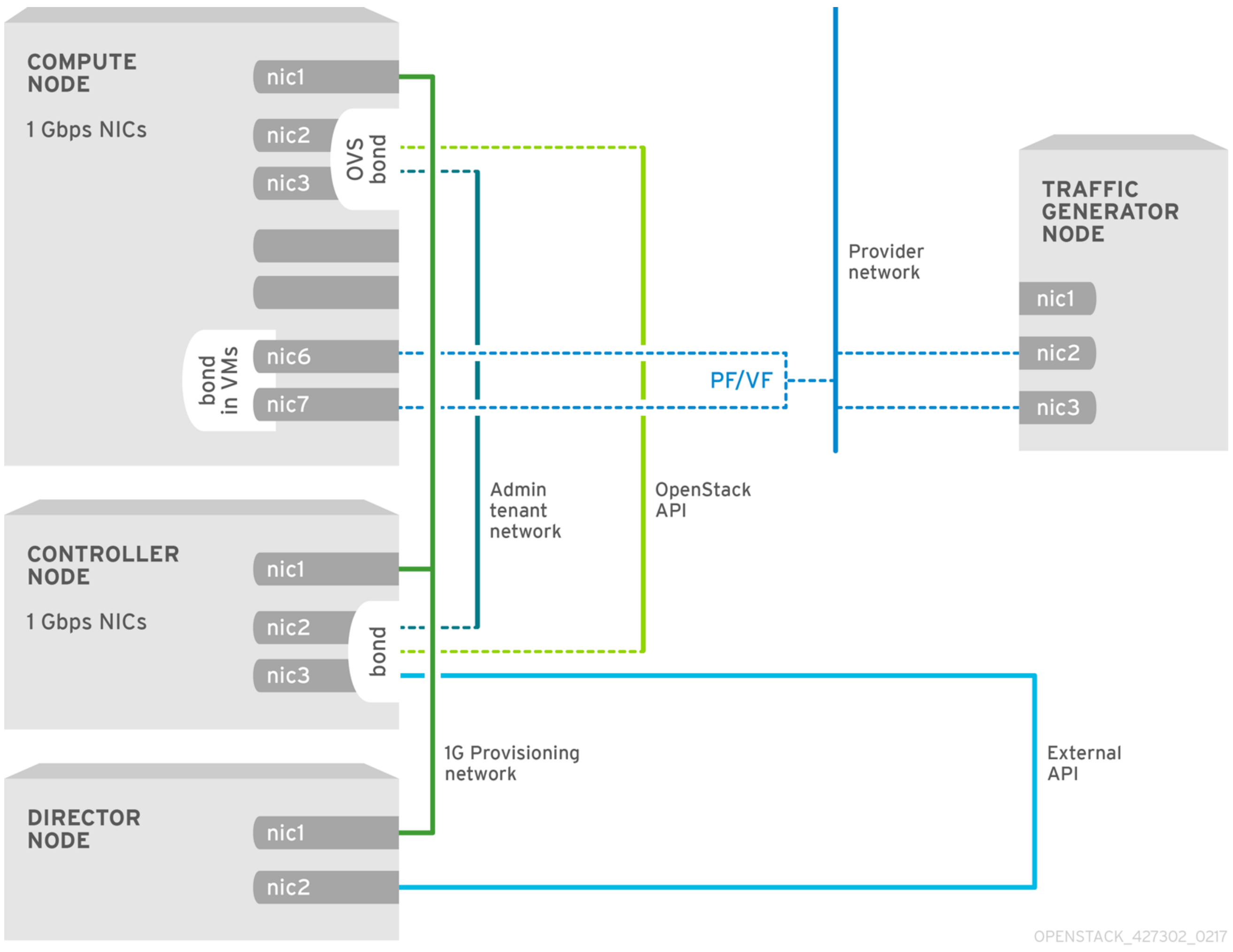
The following image has two virtual network functions (VNFs) each with the management interface represented by **mgt** and the data plane interfaces. The management interface manages the **ssh** access and so on. The data plane interfaces bond the VNFs to Data Plane Development Kit (DPDK) to ensure high availability (VNFs bond the data plane interfaces using the DPDK library). The image also has two redundant provider networks. The Compute node has two regular NICs bonded together and shared between the VNF management and the Red Hat OpenStack Platform API management.



The image shows a VNF that leverages DPDK at an application level and has access to single root I/O virtualization (SR-IOV) virtual functions (VFs) and physical functions (PFs), together for better availability or performance (depending on the fabric configuration). DPDK improves performance, while the VF/PF DPDK bonds provide support for failover (availability). The VNF vendor must ensure their DPDK poll mode driver (PMD) supports the SR-IOV card that is being exposed as a VF/PF. The management network uses Open vSwitch (OVS) so the VNF sees a “mgmt” network device using the standard virtIO drivers. Operators can use that device to initially connect to the VNF and ensure that their DPDK application bonds properly the two VF/PFs.

5.2.1. NFV SR-IOV without HCI

You can see the topology for single root I/O virtualization (SR-IOV) without hyper-converged infrastructure (HCI) for the NFV use case in the image below. It consists of compute and controller nodes with 1 Gbps NICs, and the Director node.


COMPUTE NODE
1 Gbps NICs
nic1
nic2
nic3
OVS bond
bond in VMs
nic6
nic7
Provider network
TRAFFIC GENERATOR NODE
nic1
nic2
nic3
PF/VF
Admin tenant network
OpenStack API
CONTROLLER NODE
1 Gbps NICs
nic1
nic2
nic3
bond
1G Provisioning network
External API
DIRECTOR NODE
nic1
nic2
OPENSTACK_427302_0217

CHAPTER 6. DEPLOYING SR-IOV TECHNOLOGIES

Single root I/O virtualization (SR-IOV) allows near bare metal performance by allowing instances from OpenStack direct access to a shared PCIe resource through virtual resources.

6.1. PREREQUISITES

- Install and configure the undercloud before deploying the overcloud. See the Director Installation and Usage Guide for details.

NOTE

Do not manually edit values in **/etc/tuned/cpu-partitioning-variables.conf** that are modified by Director heat templates.

6.2. CONFIGURING SR-IOV

NOTE

The CPU assignments, memory allocation and NIC configurations of the following examples may differ from your topology and use case.

1. Generate the built-in **ComputeSriov** to define nodes in the OpenStack cluster that will run **NeutronSriovAgent**, **NeutronSriovHostConfig** and default compute services.

```
# openstack overcloud roles generate \
-o /home/stack/templates/roles_data.yaml \
Controller ComputeSriov
```

2. Include the **neutron-sriov.yaml** and **roles_data.yaml** files when generating **overcloud_images.yaml** so that SR-IOV containers are prepared.

```
SERVICES=\
/usr/share/openstack-tripleo-heat-templates/environments/services

openstack overcloud container image prepare \
--namespace=registry.redhat.io/rhosp13 \
--push-destination=192.168.24.1:8787 \
--prefix=openstack- \
--tag-from-label {version}-{release} \
-e ${SERVICES}/neutron-sriov.yaml \
--roles-file /home/stack/templates/roles_data.yaml \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file=/home/stack/local_registry_images.yaml
```

NOTE

The push-destination IP address is the address that you previously set with the **local_ip** parameter in the **undercloud.conf** configuration file.

For more information on container image preparation, see Director Installation and Usage.

3. To apply the **KernelAgs** and **TunedProfile** parameters, include the **host-config-and-reboot.yaml** file from **/usr/share/openstack-tripleo-heat-templates/environments** to your deployment script.

```
openstack overcloud deploy --templates \
… \
-e /usr/share/openstack-tripleo-heat-templates/environments/host-config-and-reboot.yaml \
...
```

4. Configure the parameters for the SR-IOV nodes under **parameter_defaults** in accordance with the needs of your cluster, and the configuration of your hardware. These settings are typically added to the **network-environment.yaml** file.

```
  NeutronNetworkType: 'vlan'
  NeutronNetworkVLANRanges:
    - tenant:22:22
    - tenant:25:25
  NeutronTunnelTypes: ''
```

5. In the same file, configure role specific parameters for SR-IOV compute nodes.

NOTE

The **NeutronSriovNumVFs** parameter will soon be deprecated in favor of the **numvfs** attribute in the network configuration templates. Red Hat does not support modification of the **NeutronSriovNumVFs** parameter, nor the **numvfs** parameter, after deployment. Changing either parameter within a running environment is known to cause a permanent outage for all running instances which have an SR-IOV port on that PF. Unless you hard reboot these instances, the SR-IOV PCI device will not be visible to the instance.

```
  ComputeSriovParameters:
    IsolCpusList: "1-19,21-39"
    KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=1-19,21-39"
    TunedProfileName: "cpu-partitioning"
    NeutronBridgeMappings:
      - tenant:br-link0
    NeutronPhysicalDevMappings:
      - tenant:p7p1
      - tenant:p7p2
    NeutronSriovNumVFs:
      - p7p1:5
      - p7p2:5
    NovaPCIPassthrough:
      - vendor_id: "8086"
        product_id: "1528"
        address: "0000:06:00.0"
        physical_network: "tenant"
      - vendor_id: "8086"
        product_id: "1528"
        address: "0000:06:00.1"
    physical_network: "tenant"
  NovaVcpuPinSet: '1-19,21-39'
  NovaReservedHostMemory: 4096
```

NOTE

Do not use the **devname** parameter when configuring PCI passthrough, as the device name of a NIC can change. Instead, use **vendor_id** and **product_id** because they are more stable, or use the **address** of the NIC. For more information about how to configure **NovaPCIPassthrough**, see Guidelines for configuring **NovaPCIPassthrough**.

6. Configure the SR-IOV enabled interfaces in the **compute.yaml** network configuration template. Ensure the interfaces are configured as standalone NICs for the purposes of creating SR-IOV virtual functions (VFs):

```
             - type: interface
               name: p7p3
               mtu: 9000
               use_dhcp: false
               defroute: false
               nm_controlled: true
               hotplug: true

              - type: interface
               name: p7p4
               mtu: 9000
               use_dhcp: false
               defroute: false
               nm_controlled: true
               hotplug: true
```

7. Ensure that the list of default filters includes the value **AggregateInstanceExtraSpecsFilter**.

```
NovaSchedulerDefaultFilters:
['AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','ImagePropertiesF
ilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter','AggregateI
nstanceExtraSpecsFilter']
```

8. Deploy the overcloud.

```
TEMPLATES_HOME="/usr/share/openstack-tripleo-heat-templates"
CUSTOM_TEMPLATES="/home/stack/templates"

openstack overcloud deploy --templates \
  -r ${CUSTOM_TEMPLATES}/roles_data.yaml \
  -e ${TEMPLATES_HOME}/environments/host-config-and-reboot.yaml \
  -e ${TEMPLATES_HOME}/environments/services/neutron-sriov.yaml \
  -e ${CUSTOM_TEMPLATES}/network-environment.yaml
```

6.3. CONFIGURING HARDWARE OFFLOAD (TECHNOLOGY PREVIEW)

Open vSwitch (OVS) hardware offload is a technology preview and not recommended for production deployments. For more information about technology preview features, see Scope of Coverage Details.

The procedure for OVS hardware offload configuration shares many of the same steps as configuring SR-IOV.

Procedure

1. Generate the **ComputeSriov** role:

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

2. Add the **OvsHwOffload** parameter under role-specific parameters with a value of **true**.

3. To configure neutron to use the iptables/hybrid firewall driver implementation, include the line: **NeutronOVSFirewallDriver: iptables_hybrid**. For more information about **NeutronOVSFirewallDriver**, see Using the Open vSwitch Firewall in the *Advanced Overcloud Customization* Guide.

4. Configure the **physical_network** parameter to match your environment.

   - For VLAN, set the **physical_network** parameter to the name of the network you create in neutron after deployment. This value should also be in **NeutronBridgeMappings**.

   - For VXLAN, set the **physical_network** parameter to **null**.
     Example:

     ```
     parameter_defaults:
       NeutronOVSFirewallDriver: iptables_hybrid
       ComputeSriovParameters:
         IsolCpusList: 2-9,21-29,11-19,31-39
         KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=128
     intel_iommu=on iommu=pt"
         OvsHwOffload: true
         TunedProfileName: "cpu-partitioning"
         NeutronBridgeMappings:
           - tenant:br-tenant
         NovaPCIPassthrough:
           - vendor_id: <vendor-id>
             product_id: <product-id>
             address: <address>
             physical_network: "tenant"
           - vendor_id: <vendor-id>
             product_id: <product-id>
             address: <address>
             physical_network: "null"
         NovaReservedHostMemory: 4096
         NovaComputeCpuDedicatedSet: 1-9,21-29,11-19,31-39
     ```

   - Replace **<vendor-id>** with the vendor ID of the physical NIC.

   - Replace **<product-id>** with the product ID of the NIC VF.

   - Replace **<address>** with the address of the physical NIC.

For more information about how to configure **NovaPCIPassthrough**, see Guidelines for configuring **NovaPCIPassthrough**.

5. Ensure that the list of default filters includes **NUMATopologyFilter**:

```
 NovaSchedulerDefaultFilters:
[\'RetryFilter',\'AvailabilityZoneFilter',\'ComputeFilter',\'ComputeCapabilitiesFilter',\'ImagePropertiesFilter',\'ServerGroupAntiAffinityFilter',\'ServerGroupAffinityFilter',\'PciPassthroughFilter',\'NUMATopologyFilter']
```

6. Configure one or more network interfaces intended for hardware offload in the **compute-sriov.yaml** configuration file:

```
 - type: ovs_bridge
   name: br-tenant
   mtu: 9000
   members:
   - type: sriov_pf
     name: p7p1
     numvfs: 5
     mtu: 9000
     primary: true
     promisc: true
     use_dhcp: false
     link_mode: switchdev
```

NOTE

- Do not use the **NeutronSriovNumVFs** parameter when configuring Open vSwitch hardware offload. The number of virtual functions is specified using the **numvfs** parameter in a network configuration file used by **os-net-config**. Red Hat does not support modifying the **numvfs** setting during update or redeployment.

- Do not configure Mellanox network interfaces as a nic-config interface type **ovs-vlan** because this prevents tunnel endpoints such as VXLAN from passing traffic due to driver limitations.

7. Include the **ovs-hw-offload.yaml** file in the **overcloud deploy** command:

```
TEMPLATES_HOME=”/usr/share/openstack-tripleo-heat-templates”
CUSTOM_TEMPLATES=”/home/stack/templates”

openstack overcloud deploy --templates \
  -r ${CUSTOM_TEMPLATES}/roles_data.yaml \
  -e ${TEMPLATES_HOME}/environments/ovs-hw-offload.yaml \
  -e ${CUSTOM_TEMPLATES}/network-environment.yaml \
  -e ${CUSTOM_TEMPLATES}/neutron-ovs.yaml
```

6.3.1. Verifying OVS hardware offload

1. Confirm that a PCI device is in **switchdev** mode:

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

2. Verify if offload is enabled in OVS:

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

3. Confirm hardware offload is enabled on the NIC:

```
# ethtool -k $NIC | grep tc-offload
hw-tc-offload: on
```

6.4. DEPLOYING AN INSTANCE FOR SR-IOV

It is recommended to use host aggregates to separate high performance compute hosts. For information on creating host aggregates and associated flavors for scheduling see Creating host aggregates.

NOTE

You should use host aggregates to separate CPU pinned instances from unpinned instances. Instances that do not use CPU pinning do not respect the resourcing requirements of instances that use CPU pinning.

Deploy an instance for single root I/O virtualization (SR-IOV) by performing the following steps:

1. Create a flavor.

```
# openstack flavor create <flavor> --ram <MB> --disk <GB> --vcpus <#>
```

2. Create the network.

```
# openstack network create net1 --provider-physical-network tenant --provider-network-type
vlan --provider-segment <VLAN-ID>
# openstack subnet create subnet1 --network net1 --subnet-range 192.0.2.0/24 --dhcp
```

3. Create the port.

   - Use vnic-type **direct** to create an SR-IOV virtual function (VF) port.

   ```
   # openstack port create --network net1 --vnic-type direct sriov_port
   ```

   - Use the following to create a virtual function with hardware offload.

   ```
   # openstack port create --network net1 --vnic-type direct --binding-profile '{"capabilities":
   ["switchdev"]} sriov_hwoffload_port
   ```

   - Use vnic-type **direct-physical** to create an SR-IOV PF port.

   ```
   # openstack port create --network net1 --vnic-type direct-physical sriov_port
   ```

4. Deploy an instance

```
# openstack server create --flavor <flavor> --image <image> --nic port-id=<id> <instance
name>
```

6.5. CREATING HOST AGGREGATES

Deploy guests using cpu pinning and hugepages for increased performance. You can schedule high performance instances on a subset of hosts by matching aggregate metadata with flavor metadata.

Procedure

1. You can configure the **AggregateInstanceExtraSpecsFilter** value, and other necessary filters, through the heat parameter **NovaSchedulerDefaultFilters** under **parameter_defaults** in the **nova.conf** configuration file before deployment.

```
  parameter_defaults:
    NovaSchedulerDefaultFilters: ['AggregateInstanceExtraSpecsFilter',
'RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','Image
PropertiesFilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter','
NUMATopologyFilter']
```

NOTE

To add the **AggregateInstanceExtraSpecsFilter** configuration to an exiting cluster, you can add this parameter to the heat templates, and run the original deployment script again.

2. Create an aggregate group for single root I/O virtualization (SR-IOV), and add relevant hosts. Define metadata, for example, **sriov=true**, that matches defined flavor metadata.

```
# openstack aggregate create sriov_group
# openstack aggregate add host sriov_group compute-sriov-0.localdomain
# openstack aggregate set --property sriov=true sriov_group
```

3. Create a flavor.

```
# openstack flavor create <flavor> --ram <MB> --disk <GB> --vcpus <#>
```

4. Set additional flavor properties. Note that the defined metadata, **sriov=true**, matches the defined metadata on the SR-IOV aggregate.

```
openstack flavor set --property sriov=true --property hw:cpu_policy=dedicated --property
hw:mem_page_size=1GB <flavor>
```

CHAPTER 7. PLANNING YOUR OVS-DPDK DEPLOYMENT

To optimize your Open vSwitch with Data Plane Development Kit (OVS-DPDK) deployment for NFV, you should understand how OVS-DPDK uses the Compute node hardware (CPU, NUMA nodes, memory, NICs) and the considerations for determining the individual OVS-DPDK parameters based on your Compute node.

IMPORTANT

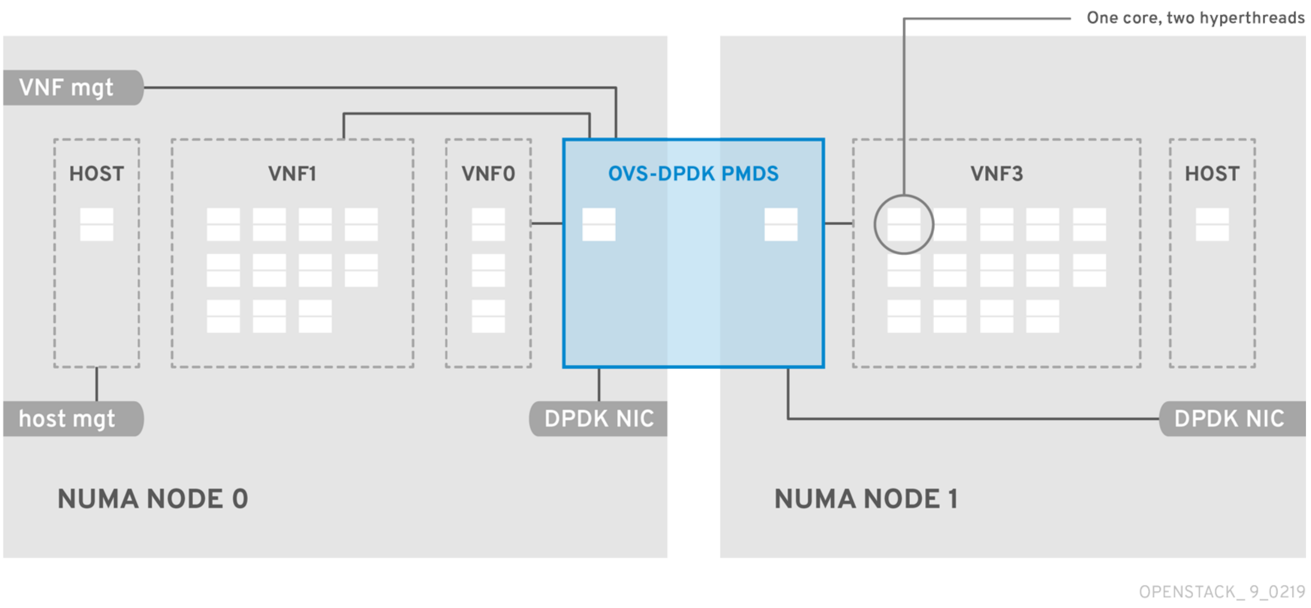
When using OVS-DPDK and the OVS native firewall (a stateful firewall based on conntrack), you can track only packets that use ICMPv4, ICMPv6, TCP, and UDP protocols. OVS marks all other types of network traffic as invalid.

See NFV performance considerations for a high-level introduction to CPUs and NUMA topology.

7.1. OVS-DPDK WITH CPU PARTITIONING AND NUMA TOPOLOGY

OVS-DPDK partitions the hardware resources for host, guests, and OVS-DPDK itself. The OVS-DPDK Poll Mode Drivers (PMDs) run DPDK active loops, which require dedicated cores. This means a list of CPUs and Huge Pages are dedicated to OVS-DPDK.

A sample partitioning includes 16 cores per NUMA node on dual socket Compute nodes. The traffic requires additional NICs since the NICs cannot be shared between the host and OVS-DPDK.



NOTE

DPDK PMD threads must be reserved on both NUMA nodes even if a NUMA node does not have an associated DPDK NIC.

OVS-DPDK performance also depends on reserving a block of memory local to the NUMA node. Use NICs associated with the same NUMA node that you use for memory and CPU pinning. Also ensure both interfaces in a bond are from NICs on the same NUMA node.

7.2. OVERVIEW OF WORKFLOWS AND DERIVED PARAMETERS

IMPORTANT

This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

You can use the OpenStack Workflow (mistral) service to derive parameters based on the capabilities of your available bare-metal nodes. OpenStack workflows use a .yaml file to define a set of tasks and actions to perform. You can use a pre-defined workbook, **derive_params.yaml**, in the **tripleo-common/workbooks/** directory. This workbook provides workflows to derive each supported parameter from the results retrieved from Bare Metal introspection. The **derive_params.yaml** workflows use the formulas from **tripleo-common/workbooks/derive_params_formulas.yaml** to calculate the derived parameters.

NOTE

You can modify the formulas in **derive_params_formulas.yaml** to suit your environment.

The **derive_params.yaml** workbook assumes all nodes for a *given composable* role have the same hardware specifications. The workflow considers the flavor-profile association and nova placement scheduler to match nodes associated with a role and uses the introspection data from the first node that matches the role.

See Troubleshooting Workflows and Executions for details on OpenStack workflows.

You can use the **-p** or **--plan-environment-file** option to add a custom **plan_environment.yaml** file to the **openstack overcloud deploy** command. The custom **plan_environment.yaml** file provides the list of workbooks and any input values to pass into the workbook. The triggered workflows merge the derived parameters back into the custom **plan_environment.yaml**, where they are available for the overcloud deployment. You can use these derived parameter results to prepare your overcloud images.

See Plan Environment Metadata for details on how to use the **--plan-environment-file** option in your deployment.

7.3. DERIVED OVS-DPDK PARAMETERS

The workflows in **derive_params.yaml** derive the DPDK parameters associated with the matching role that uses the **ComputeNeutronOvsDpdk** service.

The following is the list of parameters the workflows can automatically derive for OVS-DPDK:

- IsolCpusList
- KernelArgs
- NovaReservedHostMemory
- NovaVcpuPinSet
- OvsDpdkCoreList
- OvsDpdkSocketMemory
- OvsPmdCoreList

The **OvsDpdkMemoryChannels** parameter cannot be derived from the introspection memory bank data since the format of memory slot names are not consistent across different hardware environments.

In most cases, **OvsDpdkMemoryChannels** should be 4 (default). Use your hardware manual to determine the number of memory channels per socket and use this value to override the default.

See Section 8.1, "Deriving DPDK parameters with workflows" for configuration details.

7.4. OVERVIEW OF MANUALLY CALCULATED OVS-DPDK PARAMETERS

This section describes how Open vSwitch with Data Plane Development Kit (OVS-DPDK) uses parameters within the director **network_environment.yaml** HEAT templates to configure the CPU and memory for optimum performance. Use this information to evaluate the hardware support on your Compute nodes and how best to partition that hardware to optimize your OVS-DPDK deployment.

NOTE

You do not need to manually calculate these parameters if you use the **derived_parameters.yaml** workflow to generate these values automatically. See Overview of workflows and derived parameters

NOTE

Always pair CPU sibling threads (logical CPUs) together for the physical core when allocating CPU cores.

See Discovering your NUMA node topology to determine the CPU and NUMA nodes on your Compute nodes. You use this information to map CPU and other parameters to support the host, guest instance, and OVS-DPDK process needs.

7.4.1. CPU parameters

OVS-DPDK uses the following CPU partitioning parameters:

OvsPmdCoreList

Provides the CPU cores that are used for the DPDK poll mode drivers (PMD). Choose CPU cores that are associated with the local NUMA nodes of the DPDK interfaces. **OvsPmdCoreList** is used for the **pmd-cpu-mask** value in Open vSwitch.

- Pair the sibling threads together.
- Exclude all cores from the **OvsDpdkCoreList**
- Avoid allocating the logical CPUs (both thread siblings) of the first physical core on both NUMA nodes as these should be used for the **OvsDpdkCoreList** parameter.
- Performance depends on the number of physical cores allocated for this PMD Core list. On the NUMA node which is associated with DPDK NIC, allocate the required cores.
- For NUMA nodes with a DPDK NIC:
    - Determine the number of physical cores required based on the performance requirement and include all the sibling threads (logical CPUs) for each physical core.

- For NUMA nodes without DPDK NICs:
    - Allocate the sibling threads (logical CPUs) of one physical core (excluding the first physical core of the NUMA node).

> NOTE
>
> DPDK PMD threads must be reserved on both NUMA nodes even if a NUMA node does not have an associated DPDK NIC.

NovaVcpuPinSet

Sets cores for CPU pinning. The Compute node uses these cores for guest instances. **NovaVcpuPinSet** is used as the **vcpu_pin_set** value in the **nova.conf** file.

- Exclude all cores from the **OvsPmdCoreList** and the **OvsDpdkCoreList**.
- Include all remaining cores.
- Pair the sibling threads together.

NovaComputeCpuSharedSet

Sets the cores to be used for emulator threads. This will define the value of the nova.conf parameter **cpu_shared_set**. The recommended value for this parameter matches the value set for **OvsDpdkCoreList**.

IsolCpusList

A set of CPU cores isolated from the host processes. This parameter is used as the **isolated_cores** value in the **cpu-partitioning-variable.conf** file for the **tuned-profiles-cpu-partitioning** component.

- Match the list of cores in **OvsPmdCoreList** and **NovaVcpuPinSet**.
- Pair the sibling threads together.

OvsDpdkCoreList

Provides CPU cores for non data path OVS-DPDK processes, such as handler and revalidator threads. This parameter has no impact on overall data path performance on multi-NUMA node hardware. This parameter is used for the **dpdk-lcore-mask** value in Open vSwitch, and these cores are shared with the host.

- Allocate the first physical core (and sibling thread) from each NUMA node (even if the NUMA node has no associated DPDK NIC).
- These cores must be mutually exclusive from the list of cores in **OvsPmdCoreList** and **NovaVcpuPinSet**.

DerivePciWhitelistEnabled

To reserve virtual functions (VF) for VMs, use the **NovaPCIPassthrough** parameter to create a list of VFs passed through to Nova. VMs excluded from the list remain available for the host.
Red Hat recommends that you change the **DerivePciWhitelistEnabled** value to **false** from the default of **true**, and then manually configure the list in the **NovaPCIPassthrough** parameter.

For each VF in the list, populate the address parameter with a regular expression that resolves to the address value.

The following is an example of the manual list creation process. If NIC partitioning is enabled in a device named **eno2**, list the PCI addresses of the VFs with the following command:

```
[heat-admin@compute-0 ~]$ ls -lh /sys/class/net/eno2/device/ | grep virtfn
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn0 -> ../0000:18:06.0
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn1 -> ../0000:18:06.1
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn2 -> ../0000:18:06.2
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn3 -> ../0000:18:06.3
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn4 -> ../0000:18:06.4
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn5 -> ../0000:18:06.5
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn6 -> ../0000:18:06.6
lrwxrwxrwx. 1 root root    0 Apr 16 09:58 virtfn7 -> ../0000:18:06.7
```

In this case, the VFs 0, 4, and 6 are used by **eno2** for NIC Partitioning. Manually configure **NovaPCIPassthrough** to include VFs 1-3, 5, and 7, and consequently exclude VFs 0,4, and 6, as in the following example:

```
NovaPCIPassthrough:
  - physical_network: "sriovnet2"
  address: {"domain": ".*", "bus": "18", "slot": "06", "function": "[1-3]"}
  - physical_network: "sriovnet2"
  address: {"domain": ".*", "bus": "18", "slot": "06", "function": "[5]"}
  - physical_network: "sriovnet2"
  address: {"domain": ".*", "bus": "18", "slot": "06", "function": "[7]"}
```

7.4.2. Memory parameters

OVS-DPDK uses the following memory parameters:

OvsDpdkMemoryChannels

Maps memory channels in the CPU per NUMA node. The **OvsDpdkMemoryChannels** parameter is used by Open vSwitch as the **other_config:dpdk-extra=”-n <value>”** value.

- Use **dmidecode -t memory** or your hardware manual to determine the number of memory channels available.
- Use **ls /sys/devices/system/node/node* -d** to determine the number of NUMA nodes.
- Divide the number of memory channels available by the number of NUMA nodes.

NovaReservedHostMemory

Reserves memory in MB for tasks on the host. This value is used by the Compute node as the **reserved_host_memory_mb** value in **nova.conf**.

- Use the static recommended value of 4096 MB.

OvsDpdkSocketMemory

Specifies the amount of memory in MB to pre-allocate from the hugepage pool, per NUMA node. This value is used by Open vSwitch as the **other_config:dpdk-socket-mem** value.

- Provide as a comma-separated list. Calculate the **OvsDpdkSocketMemory** value from the MTU value of each NIC on the NUMA node.

- For a NUMA node without a DPDK NIC, use the static recommendation of 1024 MB (1GB)
- The following equation approximates the value for **OvsDpdkSocketMemory**:
  - MEMORY_REQD_PER_MTU = (ROUNDUP_PER_MTU + 800) * (4096 * 64) Bytes
    - 800 is the overhead value.
    - 4096 * 64 is the number of packets in the mempool.
- Add the MEMORY_REQD_PER_MTU for each of the MTU values set on the NUMA node and add another 512 MB as buffer. Round the value up to a multiple of 1024.

NOTE

If the MTU size is not 1500, you might get a **Failed to create memory pool** error message in **/var/log/messages**. You can ignore this error message if it occurs at instance start up. To avoid this message, add the extra **OvsDpdkSocketMemory** amount for 1500 MTU onto your **OvsDpdkSocketMemory** calculation.

Sample Calculation - MTU 2000 and MTU 9000

DPDK NICs dpdk0 and dpdk1 are on the same NUMA node 0 and configured with MTUs 9000 and 2000 respectively. The sample calculation to derive the memory required is as follows:

1. Round off the MTU values to the nearest 1024 bytes.

```
The MTU value of 9000 becomes 9216 bytes.
The MTU value of 2000 becomes 2048 bytes.
```

2. Calculate the required memory for each MTU value based on these rounded byte values.

```
Memory required for 9000 MTU = (9216 + 800) * (4096*64) = 2625634304
Memory required for 2000 MTU = (2048 + 800) * (4096*64) = 746586112
```

3. Calculate the combined total memory required, in bytes.

```
2625634304 + 746586112 + 536870912 = 3909091328 bytes.
```

   This calculation represents (Memory required for MTU of 9000) + (Memory required for MTU of 2000) + (512 MB buffer).

4. Convert the total memory required into MB.

```
3909091328 / (1024*1024) = 3728 MB.
```

5. Round this value up to the nearest 1024.

```
3724 MB rounds up to 4096 MB.
```

6. Use this value to set **OvsDpdkSocketMemory**.

```
   OvsDpdkSocketMemory: “4096,1024”
```

Sample Calculation - MTU 2000

DPDK NICs dpdk0 and dpdk1 are on the same NUMA node 0 and configured with MTUs 2000 and 2000 respectively. The sample calculation to derive the memory required is as follows:

1. Round off the MTU values to the nearest 1024 bytes.

    ```
    The MTU value of 2000 becomes 2048 bytes.
    ```

2. Calculate the required memory for each MTU value based on these rounded byte values.

    ```
    Memory required for 2000 MTU = (2048 + 800) * (4096*64) = 746586112
    ```

3. Calculate the combined total memory required, in bytes.

    ```
    746586112 + 536870912 = 1283457024 bytes.
    ```

    This calculation represents (Memory required for MTU of 2000) + (512 MB buffer).

4. Convert the total memory required into MB.

    ```
    1283457024 / (1024*1024) = 1224 MB.
    ```

5. Round this value up to the nearest 1024.

    ```
    1224 MB rounds up to 2048 MB.
    ```

6. Use this value to set **OvsDpdkSocketMemory**.

    ```
      OvsDpdkSocketMemory: “2048,1024”
    ```

7.4.3. Networking parameters

OvsDpdkDriverType

Sets the driver type used by DPDK. Use the default value of **vfio-pci**.

NeutronDatapathType

Datapath type for OVS bridges. DPDK uses the default value of **netdev**.

NeutronVhostuserSocketDir

Sets the vhost-user socket directory for OVS. Use **/var/lib/vhost_sockets** for vhost client mode.

7.4.4. Other parameters

NovaSchedulerDefaultFilters

Provides an ordered list of filters that the Compute node uses to find a matching Compute node for a requested guest instance.

VhostuserSocketGroup

Sets the vhost-user socket directory group. The default value is **qemu**. ***VhostuserSocketGroup*** should be set to **hugetlbfs** so that the **ovs-vswitchd** and **qemu** processes can access the shared hugepages and unix socket used to configure the virtio-net device. This value is role specific and should be applied to any role leveraging OVS-DPDK.

KernelArgs

Provides multiple kernel arguments to **/etc/default/grub** for the Compute node at boot time. Add the following based on your configuration:

- **hugepagesz**: Sets the size of the huge pages on a CPU. This value can vary depending on the CPU hardware. Set to 1G for OVS-DPDK deployments (**default_hugepagesz=1GB hugepagesz=1G**). Check for the **pdpe1gb** CPU flag to ensure your CPU supports 1G.

```
lshw -class processor | grep pdpe1gb
```

- **hugepages count**: Sets the number of huge pages available. This value depends on the amount of host memory available. Use most of your available memory (excluding **NovaReservedHostMemory**). You must also configure the huge pages count value within the OpenStack flavor associated with your Compute nodes.

- **iommu**: For Intel CPUs, add **"intel_iommu=on iommu=pt"**`

- **isolcpus**: Sets the CPU cores to be tuned. This value matches **IsolCpusList**.

7.4.5. Instance extra specifications

Before deploying instances in an NFV environment, create a flavor that will utilize CPU pinning, emulator thread pinning, and huge pages.

hw:cpu_policy

Set the value of this parameter to **dedicated**, so that a guest will use pinned CPUs. Instances created from a flavor with this parameter set will have an effective overcommit ratio of 1:1. The default is **shared**.

hw:mem_page_size

Set the value of this parameter to a valid string of a specific value with standard suffix, for example, **4KB**, **8MB**, or **1GB**. Use **1GB** to match the hugepagesz boot parameter. The number of huge pages available for the virtual machines is the boot parameter minus the **OvsDpdkSocketMemory**. Other valid parameter values include the following:

- small (default) - The smallest page size is used
- large - Only use large page sizes. (2MB or 1GB on x86 architectures)
- any - The compute driver may attempt large pages, but default to small if none available.

hw:emulator_threads_policy

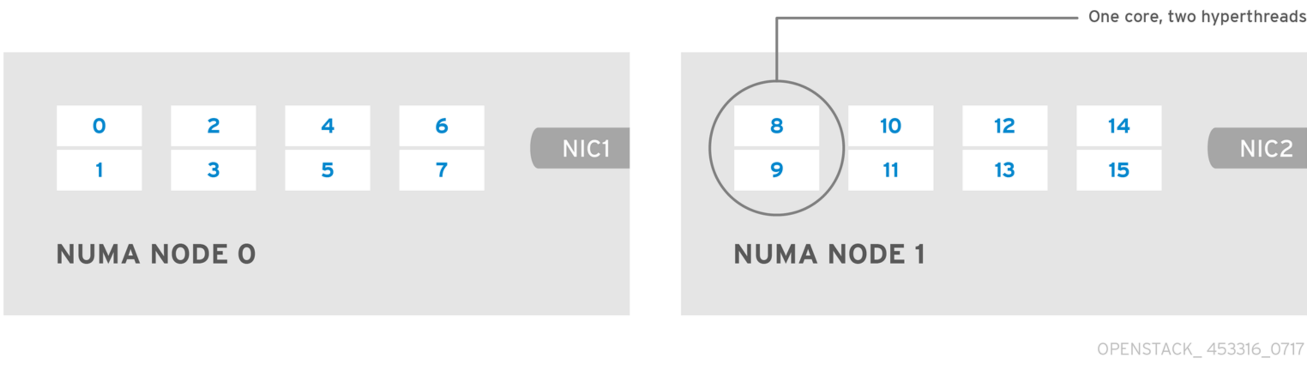
Set the value of this parameter to **share** so that emulator threads are locked to CPUs that you've identified in the heat parameter, **NovaComputeCpuSharedSet**. If an emulator thread is running on a vCPU being used for the poll mode driver (PMD) or real-time processing, you can experience packet loss or missed deadlines.

7.5. TWO NUMA NODE EXAMPLE OVS-DPDK DEPLOYMENT

This sample Compute node includes two NUMA nodes as follows:

- NUMA 0 has cores 0-7. The sibling thread pairs are (0,1), (2,3), (4,5), and (6,7)
- NUMA 1 has cores 8-15. The sibling thread pairs are (8,9), (10,11), (12,13), and (14,15).

- Each NUMA node connects to a physical NIC (NIC1 on NUMA 0 and NIC2 on NUMA 1).



NOTE

Reserve the first physical cores (both thread pairs) on each NUMA node (0,1 and 8,9) for non data path DPDK processes (**OvsDpdkCoreList**).

This example also assumes a 1500 MTU configuration, so the **OvsDpdkSocketMemory** is the same for all use cases:

```
OvsDpdkSocketMemory: “1024,1024”
```

NIC 1 for DPDK, with one physical core for PMD

In this use case, you allocate one physical core on NUMA 0 for PMD. You must also allocate one physical core on NUMA 1, even though there is no DPDK enabled on the NIC for that NUMA node. The remaining cores (not reserved for **OvsDpdkCoreList**) are allocated for guest instances. The resulting parameter settings are:

```
OvsPmdCoreList: “2,3,10,11”
NovaVcpuPinSet: “4,5,6,7,12,13,14,15”
```

NIC 1 for DPDK, with two physical cores for PMD

In this use case, you allocate two physical cores on NUMA 0 for PMD. You must also allocate one physical core on NUMA 1, even though there is no DPDK enabled on the NIC for that NUMA node. The remaining cores (not reserved for **OvsDpdkCoreList**) are allocated for guest instances. The resulting parameter settings are:

```
OvsPmdCoreList: “2,3,4,5,10,11”
NovaVcpuPinSet: “6,7,12,13,14,15”
```

NIC 2 for DPDK, with one physical core for PMD

In this use case, you allocate one physical core on NUMA 1 for PMD. You must also allocate one physical core on NUMA 0, even though there is no DPDK enabled on the NIC for that NUMA node. The remaining cores (not reserved for **OvsDpdkCoreList**) are allocated for guest instances. The resulting parameter settings are:

```
OvsPmdCoreList: “2,3,10,11”
NovaVcpuPinSet: “4,5,6,7,12,13,14,15”
```

NIC 2 for DPDK, with two physical cores for PMD

In this use case, you allocate two physical cores on NUMA 1 for PMD. You must also allocate one physical core on NUMA 0, even though there is no DPDK enabled on the NIC for that NUMA node. The remaining cores (not reserved for **OvsDpdkCoreList**) are allocated for guest instances. The resulting parameter settings are:

```
OvsPmdCoreList: “2,3,10,11,12,13”
NovaVcpuPinSet: “4,5,6,7,14,15”
```

NIC 1 and NIC2 for DPDK, with two physical cores for PMD

In this use case, you allocate two physical cores on each NUMA node for PMD. The remaining cores (not reserved for **OvsDpdkCoreList**) are allocated for guest instances. The resulting parameter settings are:

```
OvsPmdCoreList: “2,3,4,5,10,11,12,13”
NovaVcpuPinSet: “6,7,14,15”
```

NOTE

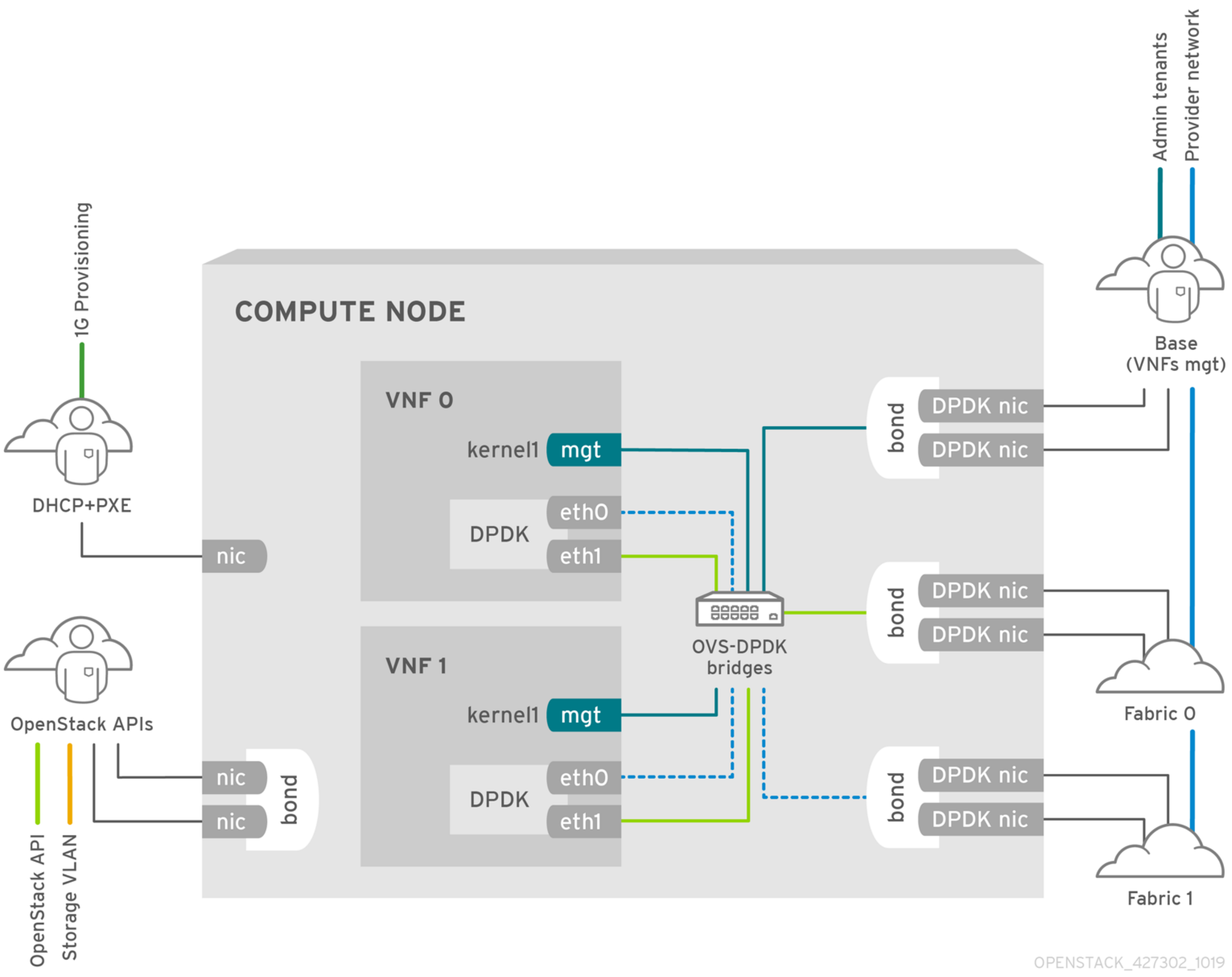
Red Hat recommends using 1 physical core per NUMA node.

7.6. TOPOLOGY OF AN NFV OVS-DPDK DEPLOYMENT

This example deployment shows an OVS-DPDK configuration and consists of two virtual network functions (VNFs) with two interfaces each:

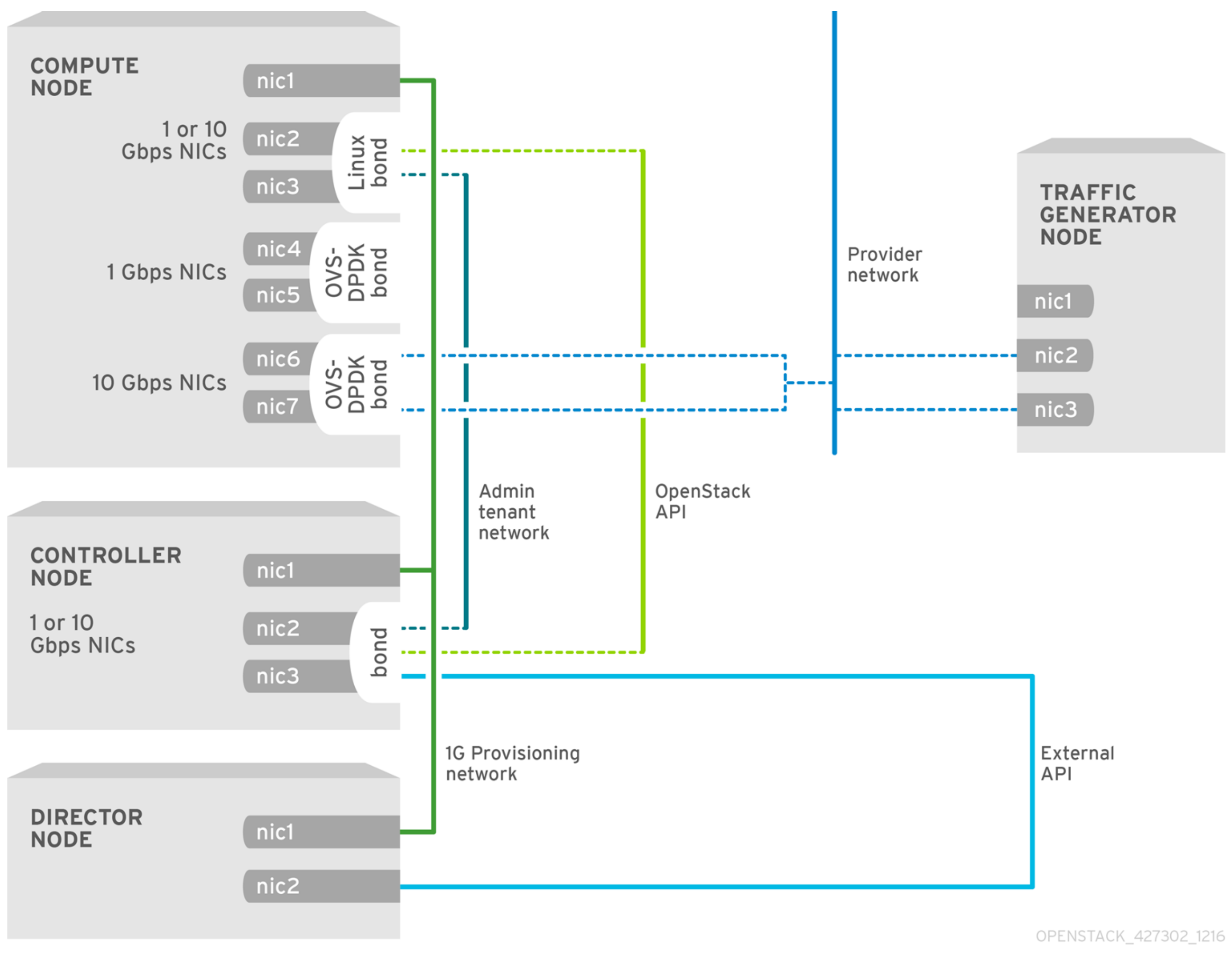
- The management interface, represented by **mgt**.
- The data plane interface.

In the OVS-DPDK deployment, the VNFs operate with inbuilt DPDK that supports the physical interface. OVS-DPDK enables bonding at the vSwitch level. For improved performance in your OVS-DPDK deployment, it is recommended that you separate kernel and OVS-DPDK NICs. To separate the management (**mgt**) network, connected to the Base provider network for the virtual machine, ensure you have additional NICs. The Compute node consists of two regular NICs for the Red Hat OpenStack Platform API management that can be reused by the Ceph API but cannot be shared with any OpenStack project.



NFV OVS-DPDK topology

The following image shows the topology for OVS_DPDK for the NFV use case. It consists of Compute and Controller nodes with 1 or 10 Gbps NICs, and the Director node.


COMPUTE NODE
nic1
1 or 10 Gbps NICs
nic2
nic3
Linux bond
1 Gbps NICs
nic4
nic5
OVS-DPDK bond
10 Gbps NICs
nic6
nic7
OVS-DPDK bond
CONTROLLER NODE
nic1
1 or 10 Gbps NICs
nic2
nic3
bond
DIRECTOR NODE
nic1
nic2
Admin tenant network
OpenStack API
Provider network
TRAFFIC GENERATOR NODE
nic1
nic2
nic3
1G Provisioning network
External API
OPENSTACK_427302_1216

CHAPTER 8. CONFIGURING AN OVS-DPDK DEPLOYMENT

This section deploys DPDK with Open vSwitch (OVS-DPDK) within the Red Hat OpenStack Platform environment. The overcloud usually consists of nodes in predefined roles such as Controller nodes, Compute nodes, and different storage node types. Each of these default roles contains a set of services defined in the core Heat templates on the director node.

You must install and configure the undercloud before you can deploy the overcloud. See the Director Installation and Usage Guide for details.

IMPORTANT

You must determine the best values for the OVS-DPDK parameters that you set in the **network-environment.yaml** file to optimize your OpenStack network for OVS-DPDK.

NOTE

Do not edit or change **isolated_cores** or other values in **etc/tuned/cpu-partitioning-variables.conf** that are modified by these director heat templates.

8.1. DERIVING DPDK PARAMETERS WITH WORKFLOWS

IMPORTANT

This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

See Section 7.2, “Overview of workflows and derived parameters” for an overview of the Mistral workflow for DPDK.

Prerequisites

You must have Bare Metal introspection, including hardware inspection extras (**inspection_extras**) enabled to provide the data retrieved by this workflow. Hardware inspection extras are enabled by default. See Inspecting the Hardware of Nodes.

Define the Workflows and Input Parameters for DPDK

The following lists the input parameters you can provide to the OVS-DPDK workflows:

num_phy_cores_per_numa_node_for_pmd

This input parameter specifies the required minimum number of cores for the NUMA node associated with the DPDK NIC. One physical core is assigned for the other NUMA nodes not associated with DPDK NIC. This parameter should be set to 1.

huge_page_allocation_percentage

This input parameter specifies the required percentage of total memory (excluding **NovaReservedHostMemory**) that can be configured as huge pages. The **KernelArgs** parameter is derived using the calculated huge pages based on the **huge_page_allocation_percentage** specified. This parameter should be set to 50.

The workflows use these input parameters along with the bare-metal introspection details to calculate appropriate DPDK parameter values.

To define the workflows and input parameters for DPDK:

1. Copy the **/usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-params.yaml** file to a local directory and set the input parameters to suit your environment.

   ```
   workflow_parameters:
     tripleo.derive_params.v1.derive_parameters:
       # DPDK Parameters #
       # Specifices the minimum number of CPU physical cores to be allocated for DPDK
       # PMD threads. The actual allocation will be based on network config, if
       # the a DPDK port is associated with a numa node, then this configuration
       # will be used, else 1.
       num_phy_cores_per_numa_node_for_pmd: 1
       # Amount of memory to be configured as huge pages in percentage. Ouf the
       # total available memory (excluding the NovaReservedHostMemory), the
       # specified percentage of the remaining is configured as huge pages.
       huge_page_allocation_percentage: 50
   ```

2. Run the **openstack overcloud deploy** command and include the following:

   - The **update-plan-only** option
   - The role file and all environment files specific to your environment
   - The **plan-environment-derived-parms.yaml** file with the **--plan-environment-file** optional argument

     ```
     $ openstack overcloud deploy --templates --update-plan-only \
     -r /home/stack/roles_data.yaml \
     -e /home/stack/<environment-file> \
     ... #repeat as necessary ...
     -p /home/stack/plan-environment-derived-params.yaml
     ```

The output of this command shows the derived results, which are also merged into the **plan-environment.yaml** file.

```
Started Mistral Workflow tripleo.validations.v1.check_pre_deployment_validations. Execution ID:
55ba73f2-2ef4-4da1-94e9-eae2fdc35535
Waiting for messages on queue 472a4180-e91b-4f9e-bd4c-1fbdfbcf414f with no timeout.
Removing the current plan files
Uploading new plan files
Started Mistral Workflow tripleo.plan_management.v1.update_deployment_plan. Execution ID:
7fa995f3-7e0f-4c9e-9234-dd5292e8c722
Plan updated.
Processing templates in the directory /tmp/tripleoclient-SY6RcY/tripleo-heat-templates
Invoking workflow (tripleo.derive_params.v1.derive_parameters) specified in plan-environment file
Started Mistral Workflow tripleo.derive_params.v1.derive_parameters. Execution ID: 2d4572bf-4c5b-
41f8-8981-c84a363dd95b
Workflow execution is completed. result:
ComputeOvsDpdkParameters:
 IsolCpusList: 1-7,17-23,9-15,25-31
 KernelArgs: default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt intel_iommu=on
```

```
isolcpus=1-7,17-23,9-15,25-31
 NovaReservedHostMemory: 4096
 NovaVcpuPinSet: 2-7,18-23,10-15,26-31
 OvsDpdkCoreList: 0,16,8,24
 OvsDpdkMemoryChannels: 4
 OvsDpdkSocketMemory: 1024,1024
 OvsPmdCoreList: 1,17,9,25
```

NOTE

The **OvsDpdkMemoryChannels** parameter cannot be derived from introspection details. In most cases, this value should be 4.

Deploy the Overcloud with the Derived Parameters

To deploy the overcloud with these derived parameters:

1. Copy the derived parameters from the **plan-environment.yaml** to the **network-environment.yaml** file.

```
  # DPDK compute node.
  ComputeOvsDpdkParameters:
    KernelArgs: default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=1-7,17-23,9-15,25-31
    TunedProfileName: "cpu-partitioning"
    IsolCpusList: "1-7,17-23,9-15,25-31"
    NovaVcpuPinSet: ['2-7,18-23,10-15,26-31']
    NovaReservedHostMemory: 4096
    OvsDpdkSocketMemory: "1024,1024"
    OvsDpdkMemoryChannels: "4"
    OvsDpdkCoreList: "0,16,8,24"
    OvsPmdCoreList: "1,17,9,25"
```

NOTE

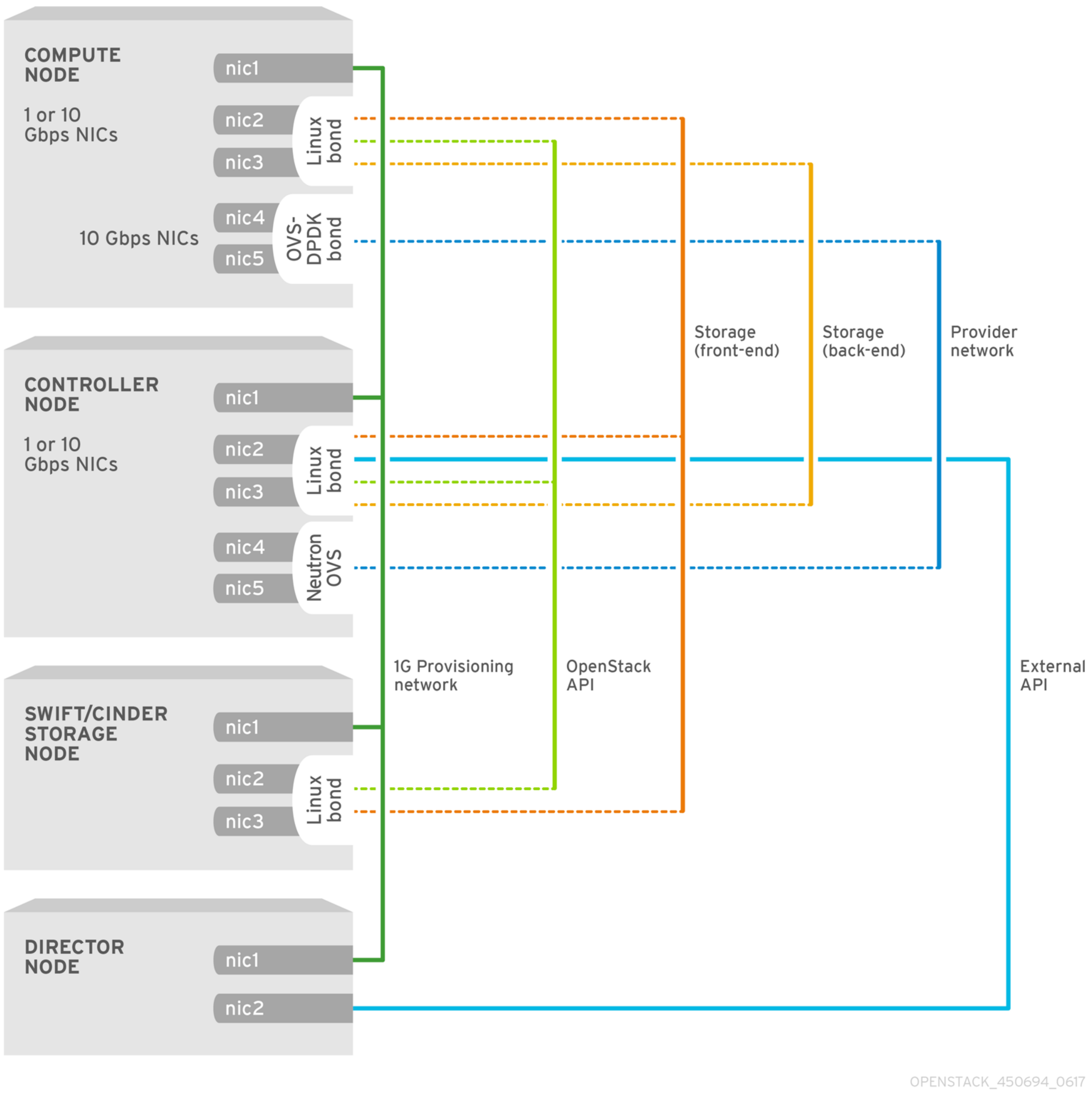
These parameters apply exclusively to the role **ComputeOvsDpdk**, and will not apply to other roles, including **Compute** or **ComputeSriov** that may exist on the same cluster. You can apply these parameters globally, but any global parameters are overwritten by role-specific parameters.

2. Deploy the overcloud using the role file and all environment files specific to your environment. See Deploying the Overcloud for details.

8.2. OVS-DPDK TOPOLOGY

With Red Hat OpenStack Platform, you can create custom deployment roles, using the composable roles feature, adding or removing services from each role. For more information on Composable Roles, see Composable Roles and Services.

This image shows a sample Open vSwitch with Data Plane Development Kit (OVS-DPDK) topology with two bonded ports for the control plane and data plane:



Configuring OVS-DPDK comprises the following tasks:

- If you use composable roles, copy and modify the **roles_data.yaml** file to add the custom role for OVS-DPDK.
- Update the appropriate **network-environment.yaml** file to include parameters for kernel arguments and DPDK arguments.
- Update the **compute.yaml** file to include the bridge for DPDK interface parameters.
- Update the **controller.yaml** file to include the same bridge details for DPDK interface parameters.
- Run the **overcloud_deploy.sh** script to deploy the overcloud with the DPDK parameters.

NOTE

This guide provides examples for CPU assignments, memory allocation, and NIC configurations that may vary from your topology and use case. See the Network Functions Virtualization Product Guide and Chapter 2, *Hardware requirements* to understand the hardware and configuration options.

Before you begin the procedure, ensure that, at the minimum, you have the following:

- OVS 2.9
- DPDK 17
- Tested NIC. For a list of tested NICs for NFV, see Section 2.1, "Network Adapter support".

NOTE

The Red Hat OpenStack Platform operates in OVS client mode for OVS-DPDK deployments.

8.3. SETTING THE MTU VALUE FOR OVS-DPDK INTERFACES

Red Hat OpenStack Platform supports jumbo frames for Open vSwitch with Data Plane Development Kit (OVS-DPDK). To set the maximum transmission unit (MTU) value for jumbo frames you must:

- Set the global MTU value for networking in the **network-environment.yaml** file.
- Set the physical DPDK port MTU value in the **compute.yaml** file. This value is also used by the vhost user interface.
- Set the MTU value within any guest instances on the Compute node to ensure that you have a comparable MTU value from end to end in your configuration.

NOTE

VXLAN packets include an extra 50 bytes in the header. Calculate your MTU requirements based on these additional header bytes. For example, an MTU value of 9000 means the VXLAN tunnel MTU value is 8950 to account for these extra bytes.

NOTE

You do not need any special configuration for the physical NIC since the NIC is controlled by the DPDK PMD and has the same MTU value set by the **compute.yaml** file. You cannot set an MTU value larger than the maximum value supported by the physical NIC.

To set the MTU value for OVS-DPDK interfaces:

1. Set the **NeutronGlobalPhysnetMtu** parameter in the **network-environment.yaml** file.

```
parameter_defaults:
  # MTU global configuration
  NeutronGlobalPhysnetMtu: 9000
```

NOTE

Ensure that the NeutronDpdkSocketMemory value in the **network-environment.yaml** file is large enough to support jumbo frames. See Section 7.4.2, “Memory parameters” for details.

2. Set the MTU value on the bridge to the Compute node in the **controller.yaml** file.

```
-
  type: ovs_bridge
  name: br-link0
  use_dhcp: false
  members:
    -
      type: interface
      name: nic3
      mtu: 9000
```

3. Set the MTU values for an OVS-DPDK bond in the **compute.yaml** file:

```
- type: ovs_user_bridge
  name: br-link0
  use_dhcp: false
  members:
    - type: ovs_dpdk_bond
      name: dpdkbond0
      mtu: 9000
      rx_queue: 2
      members:
        - type: ovs_dpdk_port
          name: dpdk0
          mtu: 9000
          members:
            - type: interface
              name: nic4
        - type: ovs_dpdk_port
          name: dpdk1
          mtu: 9000
          members:
            - type: interface
              name: nic5
```

8.4. CONFIGURING A FIREWALL FOR SECURITY GROUPS

Dataplane interfaces need a high degree of performance in a stateful firewall. To protect these interfaces, consider deploying a telco grade firewall as a virtual network function (VNF).

Controlplane interfaces can be configured by setting the **NeutronOVSFirewallDriver** parameter to **openvswitch**. This configures OpenStack Networking to use the flow-based OVS firewall driver. This is set in the **network-environment.yaml** file under **parameter_defaults**.

Example:

```
parameter_defaults:
  NeutronOVSFirewallDriver: openvswitch
```

When the OVS firewall driver is used, it is important to disable it for dataplane interfaces. This can be done with the **openstack port set** command.

Example:

```
openstack port set --no-security-group  --disable-port-security ${PORT}
```

8.5. SETTING MULTIQUEUE FOR OVS-DPDK INTERFACES

To set set same number of queues for interfaces in Open vSwitch with Data Plane Development Kit (OVS-DPDK) on the Compute node, modify the **compute.yaml** file as follows:

```
- type: ovs_user_bridge
  name: br-link0
  use_dhcp: false
  members:
    - type: ovs_dpdk_bond
      name: dpdkbond0
      mtu: 9000
      rx_queue: 2
      members:
        - type: ovs_dpdk_port
          name: dpdk0
          mtu: 9000
          members:
            - type: interface
              name: nic4
        - type: ovs_dpdk_port
          name: dpdk1
          mtu: 9000
          members:
            - type: interface
              name: nic5
```

8.6. DEPLOYING THE OVERCLOUD

1. Ensure parameters for your DPDK compute role are populated in **network-environment.yaml**. These can be copied from derived OVS-DPDK parameters if needed:

```
  # DPDK compute node.
   ComputeOvsDpdkParameters:
     KernelArgs: default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=1-7,17-23,9-15,25-31
     TunedProfileName: "cpu-partitioning"
     IsolCpusList: "1-7,17-23,9-15,25-31"
     NovaVcpuPinSet: ['2-7,18-23,10-15,26-31']
     NovaReservedHostMemory: 4096
     OvsDpdkSocketMemory: "1024,1024"
```

```
OvsDpdkMemoryChannels: "4"
OvsDpdkCoreList: "0,16,8,24"
OvsPmdCoreList: "1,17,9,25"
```

2. Deploy the overcloud using the **openstack overcloud deploy** command.

   - Include the role file and all environment files specific to your environment.

   - Apply the **KernelArgs** and **TunedProfile** parameters by including the **host-config-and-reboot.yaml** file from **/usr/share/openstack-tripleo-heat-templates/environments** to your deployment script:

   ```
   TEMPLATES_HOME=”/usr/share/openstack-tripleo-heat-templates”
   CUSTOM_TEMPLATES=”/home/stack/templates”

   openstack overcloud deploy --templates \
   -r ${CUSTOM_TEMPLATES}/roles_data.yaml \
   -e ${TEMPLATES_HOME}/environments/host-config-and-reboot.yaml \
   -e ${CUSTOM_TEMPLATES}/network-environment.yaml \
   -e ${CUSTOM_TEMPLATES}/controller.yaml
   -e ${CUSTOM_TEMPLATES}/computeovsdpdk.yaml \
   ...
   ```

8.7. KNOWN LIMITATIONS

There are certain limitations when configuring OVS-DPDK with Red Hat OpenStack Platform for the NFV use case:

- Use Linux bonds for control plane networks. Ensure both PCI devices used in the bond are on the same NUMA node for optimum performance. Neutron Linux bridge configuration is not supported by Red Hat.

- Huge pages are required for every instance running on the hosts with OVS-DPDK. If huge pages are not present in the guest, the interface appears but does not function.

- With OVS-DPDK, there is a performance degradation of services that use tap devices, such as Distributed Virtual Routing (DVR). The resulting performance is not suitable for a production environment.

- When using OVS-DPDK, ensure that all bridges on the same Compute node are of type **ovs_user_bridge**. Mixing **ovs_bridge** and **ovs_user_bridge** on the same node harms the performance, and is unsupported.

8.8. CREATING A FLAVOR AND DEPLOYING AN INSTANCE FOR OVS-DPDK

After you have completed configuring Open vSwitch with Data Plane Development Kit (OVS-DPDK) for your Red Hat OpenStack Platform deployment with NFV, you can create a flavor and deploy an instance with the following steps:

1. Create an aggregate group and add relevant hosts for OVS-DPDK. Define metadata, for example **dpdk=true**, that matches defined flavor metadata.

```
# openstack aggregate create dpdk_group
# openstack aggregate add host dpdk_group [compute-host]
# openstack aggregate set --property dpdk=true dpdk_group
```

NOTE

You should use host aggregates to separate CPU pinned instances from unpinned instances. Instances that do not use CPU pinning do not respect the resourcing requirements of instances that use CPU pinning.

2. Create a flavor.

   ```
   # openstack flavor create <flavor> --ram <MB> --disk <GB> --vcpus <#>
   ```

3. Set additional flavor properties. Note that the defined metadata, **dpdk=true**, matches the defined metadata in the DPDK aggregate.

   ```
   # openstack flavor set <flavor> --property dpdk=true --property hw:cpu_policy=dedicated --
   property hw:mem_page_size=1GB --property hw:emulator_threads_policy=isolate
   ```

   For details on the emulator threads policy for performance improvements, see: Configure Emulator Threads to run on a Dedicated Physical CPU.

4. Create the network.

   ```
   # openstack network create net1 --provider-physical-network tenant --provider-network-type
   vlan --provider-segment <VLAN-ID>
   # openstack subnet create subnet1 --network net1 --subnet-range 192.0.2.0/24 --dhcp
   ```

5. Optional: If you use multiqueue with OVS-DPDK, set the **hw_vif_multiqueue_enabled** property on the image that you want to use to create a instance:

   ```
   # openstack image set --property hw_vif_multiqueue_enabled=true <image>
   ```

6. Deploy an instance.

   ```
   # openstack server create --flavor <flavor> --image <glance image> --nic net-id=<network
   ID> <server_name>
   ```

8.9. TROUBLESHOOTING THE CONFIGURATION

This section describes the steps to troubleshoot the Open vSwitch with Data Plane Development Kit (DPDK-OVS) configuration.

1. Review the bridge configuration and confirm that the bridge was created with the **datapath_type=netdev**.

   ```
   # ovs-vsctl list bridge br0
   _uuid               : bdce0825-e263-4d15-b256-f01222df96f3
   auto_attach         : []
   controller          : []
   datapath_id         : "00002608cebd154d"
   ```

```
datapath_type       : netdev
datapath_version    : "<built-in>"
external_ids        : {}
fail_mode           : []
flood_vlans         : []
flow_tables         : {}
ipfix               : []
mcast_snooping_enable: false
mirrors             : []
name                : "br0"
netflow             : []
other_config        : {}
ports               : [52725b91-de7f-41e7-bb49-3b7e50354138]
protocols           : []
rstp_enable         : false
rstp_status         : {}
sflow               : []
status              : {}
stp_enable          : false
```

2. Confirm that the docker container **neutron_ovs_agent** is configured to start automatically.

```
# docker inspect neutron_ovs_agent | grep -A1 RestartPolicy
            "RestartPolicy": {
                "Name": "always",
```

If the container is having trouble starting, you can view any related messages.

```
# less /var/log/containers/neutron/openvswitch-agent.log
```

3. Confirm that the PMD CPU mask of the **ovs-dpdk** are pinned to the CPUs. In case of HT, use sibling CPUs.
For example, take **CPU4**:

```
# cat /sys/devices/system/cpu/cpu4/topology/thread_siblings_list
4,20
```

So, using CPU 4 and 20:

```
# ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=0x100010
```

Display their status:

```
# tuna -t ovs-vswitchd -CP
thread  ctxt_switches pid SCHED_ rtpri affinity voluntary nonvoluntary       cmd
3161 OTHER  0     6 765023       614 ovs-vswitchd
3219   OTHER  0     6      1         0    handler24
3220   OTHER  0     6      1         0    handler21
3221   OTHER  0     6      1         0    handler22
3222   OTHER  0     6      1         0    handler23
3223   OTHER  0     6      1         0    handler25
3224   OTHER  0     6      1         0    handler26
3225   OTHER  0     6      1         0    handler27
3226   OTHER  0     6      1         0    handler28
```

```
3227   OTHER  0     6      2         0    handler31
3228   OTHER  0     6      2         4    handler30
3229   OTHER  0     6      2         5    handler32
3230   OTHER  0     6 953538       431   revalidator29
3231   OTHER  0     6   1424258       976   revalidator33
3232   OTHER  0     6   1424693       836   revalidator34
3233   OTHER  0     6 951678       503   revalidator36
3234   OTHER  0     6   1425128       498   revalidator35
*3235   OTHER  0     4 151123        51        pmd37*
*3236   OTHER  0    20 298967        48        pmd38*
3164   OTHER  0     6  47575         0  dpdk_watchdog3
3165   OTHER  0     6 237634         0   vhost_thread1
3166   OTHER  0     6   3665         0        urcu2
```

CHAPTER 9. TUNING A RED HAT OPENSTACK PLATFORM ENVIRONMENT

9.1. TRUSTED VIRTUAL FUNCTIONS

You can configure physical functions (PFs) to trust virtual functions (VFs) so that VFs can perform some privileged actions. For example, you can use this configuration to allow VFs to enable promiscuous mode or to change a hardware address.

9.1.1. Providing trust

Prerequisites

- An operational installation Red Hat OpenStack Platform director

Procedure

Complete the following steps to deploy the overcloud with the parameters necessary to enable physical function trust of virtual functions:

1. Add the **NeutronPhysicalDevMappings** parameter under the **parameter_defaults** section to make the link between the logical network name and the physical interface.

```
parameter_defaults:
  NeutronPhysicalDevMappings:
    - sriov2:p5p2
```

2. Add the new property "trusted" to the existing parameters related to SR-IOV.

```
parameter_defaults:
  NeutronPhysicalDevMappings:
    - sriov2:p5p2
  NeutronSriovNumVFs: ["p5p2:8"]
  NovaPCIPassthrough:
    - vendor_id: "8086"
      product_id: "1572"
      physical_network: "sriov2"
      trusted: "true"
```

NOTE

You must include quotation marks around the value "true".

IMPORTANT

Complete the following step only in trusted environments. This step will allow non-administrative accounts the ability to bind trusted ports.

3. Modify permissions to allow users the capability of creating and updating port bindings.

```
parameter_defaults:
  NeutronApiPolicies: {
    operator_create_binding_profile: { key: 'create_port:binding:profile', value:
```

```
'rule:admin_or_network_owner'},
    operator_get_binding_profile: { key: 'get_port:binding:profile', value:
'rule:admin_or_network_owner'},
    operator_update_binding_profile: { key: 'update_port:binding:profile', value:
'rule:admin_or_network_owner'}
  }
```

9.1.2. Utilizing trusted virtual functions

Execute the following on a fully deployed overcloud to utilize trusted virtual functions.

Creating a trusted VF network

1. Create a network of type vlan.

```
openstack network create trusted_vf_network  --provider-network-type vlan \
 --provider-segment 111 --provider-physical-network sriov2 \
 --external --disable-port-security
```

2. Create a subnet.

```
openstack subnet create --network trusted_vf_network \
  --ip-version 4 --subnet-range 192.168.111.0/24 --no-dhcp \
 subnet-trusted_vf_network
```

3. Create a port, setting the **vnic-type** option to direct, and the **binding-profile** option to true.

```
openstack port create --network sriov111 \
--vnic-type direct --binding-profile trusted=true \
sriov111_port_trusted
```

4. Create an instance binding it to the previously created trusted port.

```
openstack server create --image rhel --flavor dpdk  --network internal --port
trusted_vf_network_port_trusted --config-drive True --wait rhel-dpdk-sriov_trusted
```

Verify the trusted virtual function configuration on the hypervior

On the compute node that hosts the newly created instance, run the following command:

```
# ip link
7: p5p2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode
DEFAULT group default qlen 1000
    link/ether b4:96:91:1c:40:fa brd ff:ff:ff:ff:ff:ff
    vf 6 MAC fa:16:3e:b8:91:c2, vlan 111, spoof checking off, link-state auto, trust on, query_rss off
    vf 7 MAC fa:16:3e:84:cf:c8, vlan 111, spoof checking off, link-state auto, trust off, query_rss off
```

View the output of the **ip link** command and verify that the trust status of the virtual function is **trust on**. The example output contains details of an environment that contains two ports. Note that **vf 6** contains the text **trust on**.

9.2. CONFIGURING RX/TX QUEUE SIZE

You can experience packet loss at high packet rates above 3.5 million packets per second (mpps) for many reasons, such as:

- a network interrupt
- a SMI
- packet processing latency in the Virtual Network Function

To prevent packet loss, increase the queue size from the default of 512 to a maximum of 1024.

Prerequisites

- To configure RX, ensure that you have libvirt v2.3 and QEMU v2.7.
- To configure TX, ensure that you have libvirt v3.7 and QEMU v2.10.

Procedure

- To increase the RX and TX queue size, include the following lines to the **parameter_defaults:** section of a relevant director role. Here is an example with ComputeOvsDpdk role:

```
parameter_defaults:
  ComputeOvsDpdkParameters:
    -NovaLibvirtRxQueueSize: 1024
    -NovaLibvirtTxQueueSize: 1024
```

Testing

- You can observe the values for RX queue size and TX queue size in the nova.conf file:

```
[libvirt]
rx_queue_size=1024
tx_queue_size=1024
```

- You can check the values for RX queue size and TX queue size in the VM instance XML file generated by libvirt on the compute host.

```
<devices>
   <interface type='vhostuser'>
     <mac address='56:48:4f:4d:5e:6f'/>
     <source type='unix' path='/tmp/vhost-user1' mode='server'/>
     <model type='virtio'/>
     <driver name='vhost' rx_queue_size='1024'   tx_queue_size='1024' />
     <address type='pci' domain='0x0000' bus='0x00' slot='0x10' function='0x0'/>
   </interface>
</devices>
```

  To verify the values for RX queue size and TX queue size, use the following command on a KVM host:

```
$ virsh dumpxml <vm name> | grep queue_size
```

- You can check for improved performance, such as 3.8 mpps/core at 0 frame loss.

9.3. ENABLING RT-KVM FOR NFV WORKLOADS

This section describes the steps to install and configure Red Hat Enterprise Linux 7.5 Real Time KVM (RT-KVM) for the Red Hat OpenStack Platform. Red Hat OpenStack Platform provides real-time capabilities with a new Real-time Compute node role that provisions Red Hat Enterprise Linux for Real-Time, as well as the additional RT-KVM kernel module, and automatic configuration of the Compute node.

9.3.1. Planning for your RT-KVM Compute nodes

You must use Red Hat certified servers for your RT-KVM Compute nodes. See Red Hat Enterprise Linux for Real Time 7 certified servers for details.

See Registering and updating your undercloud for details on how to enable the **rhel-7-server-nfv-rpms** repository for RT-KVM, and ensuring your system is up to date.

NOTE

You will need a separate subscription to a **Red Hat OpenStack Platform for Real Time** SKU before you can access this repository.

Building the real-time image

Use the following steps to build the overcloud image for Real-time Compute nodes:

1. To initialize the stack user to use the director command line tools, run the following command:

```
[stack@undercloud-0 ~]$ source ~/stackrc
```

2. Install the libguestfs-tools package on the undercloud to get the virt-customize tool:

```
(undercloud) [stack@undercloud-0 ~]$ sudo yum install libguestfs-tools
```

IMPORTANT

If you install the **libguestfs-tools** package on the undercloud, disable **iscsid.socket** to avoid port conflicts with the **tripleo_iscsid** service on the undercloud:

```
$ sudo systemctl disable --now iscsid.socket
```

3. Extract the images:

```
(undercloud) [stack@undercloud-0 ~]$ tar -xf /usr/share/rhosp-director-images/overcloud-full.tar
(undercloud) [stack@undercloud-0 ~]$ tar -xf /usr/share/rhosp-director-images/ironic-python-agent.tar
```

4. Copy the default image:

```
(undercloud) [stack@undercloud-0 ~]$ cp overcloud-full.qcow2 overcloud-realtime-compute.qcow2
```

5. Register your image to enable Red Hat repositories relevant to your customizations. Replace **[username]** and **[password]** with valid credentials in the following example.

```
virt-customize -a overcloud-realtime-compute.qcow2 --run-command \
'subscription-manager register --username=[username] --password=[password]'
```

NOTE

Remove credentials from the history file anytime they are used on the command prompt. You can delete individual lines in history using the **history -d** command followed by the line number.

6. Find a list of pool IDs from your account's subscriptions, and attach the appropriate pool ID to your image.

```
sudo subscription-manager list --all --available | less
...
virt-customize -a overcloud-realtime-compute.qcow2 --run-command \
'subscription-manager attach --pool [pool-ID]'
```

7. Add repositories necessary for Red Hat OpenStack Platform with NFV.

```
virt-customize -a overcloud-realtime-compute.qcow2 --run-command \
'subscription-manager repos --enable=rhel-7-server-nfv-rpms \
--enable=rhel-7-server-rpms \
--enable=rhel-7-server-rh-common-rpms \
--enable=rhel-7-server-extras-rpms \
--enable=rhel-7-server-openstack-13-rpms'
```

8. Create a script to configure real-time capabilities on the image.

```
(undercloud) [stack@undercloud-0 ~]$ cat <<'EOF' > rt.sh
  #!/bin/bash

  set -eux

  yum -v -y --setopt=protected_packages= erase kernel.$(uname -m)
  yum -v -y install kernel-rt kernel-rt-kvm tuned-profiles-nfv-host
  EOF
```

9. Run the script to configure the RT image:

```
(undercloud) [stack@undercloud-0 ~]$ virt-customize -a overcloud-realtime-compute.qcow2 -
v --run rt.sh 2>&1 | tee virt-customize.log
```

NOTE

You may see the following error in the **rt.sh** script output: **grubby fatal error: unable to find a suitable template**. You can safely ignore this error.

10. You can check that the packages installed using the **rt.sh** script installed correctly by examining the **virt-customize.log** file that was created from the previous command.

```
(undercloud) [stack@undercloud-0 ~]$ cat virt-customize.log | grep Verifying

  Verifying  : kernel-3.10.0-957.el7.x86_64                              1/1
  Verifying  : 10:qemu-kvm-tools-rhev-2.12.0-18.el7_6.1.x86_64              1/8
  Verifying  : tuned-profiles-realtime-2.10.0-6.el7_6.3.noarch             2/8
  Verifying  : linux-firmware-20180911-69.git85c5d90.el7.noarch             3/8
  Verifying  : tuned-profiles-nfv-host-2.10.0-6.el7_6.3.noarch             4/8
  Verifying  : kernel-rt-kvm-3.10.0-957.10.1.rt56.921.el7.x86_64            5/8
  Verifying  : tuna-0.13-6.el7.noarch                                   6/8
  Verifying  : kernel-rt-3.10.0-957.10.1.rt56.921.el7.x86_64              7/8
  Verifying  : rt-setup-2.0-6.el7.x86_64                                8/8
```

11. Relabel SELinux:

```
(undercloud) [stack@undercloud-0 ~]$ virt-customize -a overcloud-realtime-compute.qcow2 -
-selinux-relabel
```

12. Extract vmlinuz and initrd:

NOTE

The software version in the **vmlinuz** and **initramfs** filenames vary with the kernel version. Use the relevant software version in the filename, for example **image/boot/vmlinuz-3.10.0-862.rt56.804.el7x86_64**, or use the wildcard symbol ***** instead.

```
(undercloud) [stack@undercloud-0 ~]$ mkdir image
(undercloud) [stack@undercloud-0 ~]$ guestmount -a overcloud-realtime-compute.qcow2 -i -
-ro image
(undercloud) [stack@undercloud-0 ~]$ cp image/boot/vmlinuz-*.x86_64 ./overcloud-realtime-
compute.vmlinuz
(undercloud) [stack@undercloud-0 ~]$ cp image/boot/initramfs-*.x86_64.img ./overcloud-
realtime-compute.initrd
(undercloud) [stack@undercloud-0 ~]$ guestunmount image
```

13. Upload the image:

```
(undercloud) [stack@undercloud-0 ~]$ openstack overcloud image upload --update-existing -
-os-image-name overcloud-realtime-compute.qcow2
```

You now have a real-time image you can use with the **ComputeOvsDpdkRT** composable role on select Compute nodes.

Modifying BIOS settings on RT-KVM Compute nodes

To reduce latency on your RT-KVM Compute nodes, you must modify the BIOS settings. You should disable all options for the following in your Compute node BIOS settings:

- Power Management
- Hyper-Threading
- CPU sleep states

- Logical processors

See Setting BIOS parameters for descriptions of these settings and the impact of disabling them. See your hardware manufacturer documentation for complete details on how to change BIOS settings.

9.3.2. Configuring OVS-DPDK with RT-KVM

NOTE

You must determine the best values for the OVS-DPDK parameters that you set in the network-environment.yaml file to optimize your OpenStack network for OVS-DPDK. See Section 8.1, "Deriving DPDK parameters with workflows" for details.

9.3.2.1. Generating the ComputeOvsDpdk composable role

You use the **ComputeOvsDpdkRT** role to specify Compute nodes that use the real-time compute image.

Generate **roles_data.yaml** for the **ComputeOvsDpdkRT** role.

```
# (undercloud) [stack@undercloud-0 ~]$ openstack overcloud roles generate -o roles_data.yaml
Controller ComputeOvsDpdkRT
```

9.3.2.2. Configuring the OVS-DPDK parameters

IMPORTANT

Attempting to deploy Data Plane Development Kit (DPDK) without appropriate values causes the deployment to fail or lead to unstable deployments. You must determine the best values for the OVS-DPDK parameters set in the **network-environment.yaml** file to optimize your OpenStack network for OVS-DPDK. See Section 8.1, "Deriving DPDK parameters with workflows" for details.

1. Add the nic configuration for the OVS-DPDK role you use under **resource_registry**:

```
resource_registry:
  # Specify the relative/absolute path to the config files you want to use for override the
default.
  OS::TripleO::ComputeOvsDpdkRT::Net::SoftwareConfig: nic-configs/compute-ovs-
dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
```

2. Under **parameter_defaults**, Set the OVS-DPDK and RT-KVM parameters:

```
  # DPDK compute node.
  ComputeOvsDpdkRTParameters:
    KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=1-7,17-23,9-15,25-31"
    TunedProfileName: "realtime-virtual-host"
    IsolCpusList: "1-7,17-23,9-15,25-31"
    NovaVcpuPinSet: ['2-7,18-23,10-15,26-31']
    NovaReservedHostMemory: 4096
    OvsDpdkSocketMemory: "1024,1024"
    OvsDpdkMemoryChannels: "4"
    OvsDpdkCoreList: "0,16,8,24"
    OvsPmdCoreList: "1,17,9,25"
    VhostuserSocketGroup: "hugetlbfs"
  ComputeOvsDpdkRTImage: "overcloud-realtime-compute"
```

9.3.2.3. Preparing the container images.

Prepare the container images:

```
(undercloud) [stack@undercloud-0 ~]$ openstack overcloud container image prepare --
namespace=192.0.40.1:8787/rhosp13 --env-file=/home/stack/ospd-13-vlan-dpdk/docker-images.yaml
-e /usr/share/openstack-tripleo-heat-templates/environments/docker.yaml -e /usr/share/openstack-
tripleo-heat-templates/environments/docker-ha.yaml -e /usr/share/openstack-tripleo-heat-
templates/environments/services-docker/neutron-ovs-dpdk.yaml -e /home/stack/ospd-13-vlan-
dpdk/network-environment.yaml --roles-file /home/stack/ospd-13-vlan-dpdk/roles_data.yaml --
prefix=openstack- --tag=2018-03-29.1 --set ceph_namespace=registry.redhat.io/rhceph --set
ceph_image=rhceph-3-rhel7 --set ceph_tag=latest
```

9.3.2.4. Deploying the overcloud

Deploy the overcloud for ML2-OVS:

```
(undercloud) [stack@undercloud-0 ~]$ openstack overcloud deploy \
--templates \
-r /home/stack/ospd-13-vlan-dpdk-ctlplane-bonding-rt/roles_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/host-config-and-reboot.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-ovs-dpdk.yaml
\
-e /home/stack/ospd-13-vxlan-dpdk-data-bonding-rt-hybrid/docker-images.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-data-bonding-rt-hybrid/network-environment.yaml
```

9.3.3. Launching an RT-KVM Instance

To launch an RT-KVM instance on a real-time enabled Compute node:

1. Create an RT-KVM flavor on the overcloud:

   ```
   # openstack flavor create --ram 4096 --disk 20 --vcpus 4 <flavor-name>
   # openstack flavor set --property hw:cpu_policy=dedicated <flavor-name>
   # openstack flavor set --property hw:cpu_realtime=yes <flavor-name>
   # openstack flavor set --property hw:mem_page_size=1GB <flavor-name>
   # openstack flavor set --property hw:cpu_realtime_mask="^0-1" <flavor-name>
   # openstack flavor set --property hw:emulator_threads_policy=isolate <flavor-name>
   ```

2. Launch an RT-KVM instance:

   ```
   # openstack server create  --image <rhel> --flavor <flavor-name> --nic net-id=<dpdk-net>
   test-rt
   ```

3. Optionally, verify that the instance uses the assigned emulator threads:

```
# virsh dumpxml <instance-id> | grep vcpu -A1
<vcpu placement='static'>4</vcpu>
<cputune>
  <vcpupin vcpu='0' cpuset='1'/>
  <vcpupin vcpu='1' cpuset='3'/>
  <vcpupin vcpu='2' cpuset='5'/>
  <vcpupin vcpu='3' cpuset='7'/>
  <emulatorpin cpuset='0-1'/>
  <vcpusched vcpus='2-3' scheduler='fifo'
  priority='1'/>
</cputune>
```

9.4. CONFIGURING A NUMA-AWARE VSWITCH (TECHNOLOGY PREVIEW)

IMPORTANT

This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

Before you implement a NUMA-aware vSwitch, examine the following components of your hardware configuration:

- The number of physical networks.
- The placement of PCI cards.
- The physical architecture of the servers.

Memory-mapped I/O (MMIO) devices, such as PCIe NICs, are associated with specific NUMA nodes. When a VM and the NIC are on different NUMA nodes, there is a significant decrease in performance. To increase performance, align PCIe NIC placement and instance processing on the same NUMA node.

Use this feature to ensure that instances that share a physical network are located on the same NUMA node. To optimize datacenter hardware, you can leverage load-sharing VMs by using multiple networks, different network types, or bonding.

IMPORTANT

To architect NUMA-node load sharing and network access correctly, you must understand the mapping of the PCIe slot and the NUMA node. For detailed information on your specific hardware, refer to your vendor’s documentation.

To prevent a cross-NUMA configuration, place the VM on the correct NUMA node, by providing the location of the NIC to Nova.

Prerequisites

- You have enabled the filter “NUMATopologyFilter”

Procedure

- Set a new **NeutronPhysnetNUMANodesMapping** parameter to map the physical network to the NUMA node that you associate with the physical network.
- If you use tunnels, such as VxLAN or GRE, you must also set the **NeutronTunnelNUMANodes** parameter.

```
parameter_defaults:
  NeutronPhysnetNUMANodesMapping: {<physnet_name>: [<NUMA_NODE>]}
  NeutronTunnelNUMANodes: <NUMA_NODE>,<NUMA_NODE>
```

Here is an example with two physical networks tunneled to NUMA node 0:

- one project network associated with NUMA node 0
- one management network without any affinity

```
parameter_defaults:
  NeutronBridgeMappings:
    - tenant:br-link0
  NeutronPhysnetNUMANodesMapping: {tenant: [1], mgmt: [0,1]}
  NeutronTunnelNUMANodes: 0
```

Testing

- Observe the configuration in the file /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf

```
[neutron_physnet_tenant]
numa_nodes=1
[neutron_tunnel]
numa_nodes=1
```

- Confirm the new configuration with the **lscpu** command:

```
$ lscpu
```

- Launch a VM, with the NIC attached to the appropriate network

9.5. CONFIGURING QUALITY OF SERVICE (QOS) IN AN NFVI ENVIRONMENT

For details on Configuring QoS, see Configuring Real-Time Compute. Support is limited to QoS rule type **bandwidth-limit** on SR-IOV and OVS-DPDK egress interfaces.

9.6. DEPLOYING AN OVERCLOUD WITH HCI AND DPDK

You can deploy your NFV infrastructure with hyper-converged nodes, by co-locating and configuring Compute and Ceph Storage services for optimized resource usage.

For more information about hyper-converged infrastructure (HCI), see: Hyper Converged Infrastructure Guide

Prerequisites

- Red Hat OpenStack Platform 13.12 Maintenance Release 19 December 2019 or newer.
- Ceph 12.2.12-79 (luminous) or newer.
- Ceph-ansible 3.2.38 or newer.

Procedure

1. Install **ceph-ansible** on the undercloud.

   ```
   $ sudo yum install ceph-ansible -y
   ```

2. Generate the **roles_data.yaml** file for the ComputeHCI role.

   ```
   $ openstack overcloud roles generate -o ~/<templates>/roles_data.yaml Controller \
    ComputeHCIOvsDpdk
   ```

3. Create and configure a new flavor with the **openstack flavor create** and **openstack flavor set** commands. For more information about creating a flavor, see Creating a new role in the *Advanced Overcloud Customization Guide*.

4. Deploy the overcloud with the custom **roles_data.yaml** file that you generated.

   ```
   # time openstack overcloud deploy --templates \
    --timeout 360 \
    -r ~/<templates>/roles_data.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
   ansible.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-ovs-
   dpdk.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/host-config-and-reboot.yaml \
    -e ~/<templates>/<custom environment file>
   ```

9.6.1. Example NUMA node configuration

For increased performance, place the tenant network and Ceph object service daemon (OSD)s in one NUMA node, such as NUMA-0, and the VNF and any non-NFV VMs in another NUMA node, such as NUMA-1.

CPU allocation:

NUMA-0	NUMA-1
Number of Ceph OSDs * 4 HT	Guest vCPU for the VNF and non-NFV VMs
DPDK lcore - 2 HT	DPDK lcore - 2 HT
DPDK PMD - 2 HT	DPDK PMD - 2 HT

Example of CPU allocation:

	NUMA-0	NUMA-1
Ceph OSD	32,34,36,38,40,42,76,78,80,82,84,86	
DPDK-lcore	0,44	1,45
DPDK-pmd	2,46	3,47
nova		5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,49,51,53,55,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87

9.6.2. Example ceph configuration file

```
parameter_defaults:
  CephPoolDefaultSize: 3
  CephPoolDefaultPgNum: 64
  CephPools:
    - {"name": backups, "pg_num": 128, "pgp_num": 128, "application": "rbd"}
    - {"name": volumes, "pg_num": 256, "pgp_num": 256, "application": "rbd"}
    - {"name": vms, "pg_num": 64, "pgp_num": 64, "application": "rbd"}
    - {"name": images, "pg_num": 32, "pgp_num": 32, "application": "rbd"}
  CephConfigOverrides:
    osd_recovery_op_priority: 3
    osd_recovery_max_active: 3
    osd_max_backfills: 1
  CephAnsibleExtraConfig:
    nb_retry_wait_osd_up: 60
    delay_wait_osd_up: 20
    is_hci: true
    # 3 OSDs * 4 vCPUs per SSD = 12 vCPUs (list below not used for VNF)
    ceph_osd_docker_cpuset_cpus: "32,34,36,38,40,42,76,78,80,82,84,86" # 1
    # cpu_limit 0 means no limit as we are limiting CPUs with cpuset above
    ceph_osd_docker_cpu_limit: 0                                       # 2
    # numactl preferred to cross the numa boundary if we have to
    # but try to only use memory from numa node0
    # cpuset-mems would not let it cross numa boundary
    # lots of memory so NUMA boundary crossing unlikely
    ceph_osd_numactl_opts: "-N 0 --preferred=0"                        # 3
  CephAnsibleDisksConfig:
    osds_per_device: 1
    osd_scenario: lvm
    osd_objectstore: bluestore
    devices:
      - /dev/sda
      - /dev/sdb
      - /dev/sdc
```

Assign CPU resources for ceph OSD processes with the following parameters. Adjust the values based on the workload and hardware in this hyperconverged environment.

1. ceph_osd_docker_cpuset_cpus: Allocate 4 CPU threads for each OSD for SSD disks, or 1 CPU for each OSD for HDD disks. Include the list of cores and sibling threads from the NUMA node associated with ceph, and the CPUs not found in the three lists: **NovaVcpuPinSet**, **OvsDpdkCoreList**, and **OvsPmdCoreList**.

2. ceph_osd_docker_cpu_limit: Set this value to **0**, to pin the ceph OSDs to the CPU list from **ceph_osd_docker_cpuset_cpus**.

3. ceph_osd_numactl_opts: Set this value to **preferred** for cross-NUMA operations, as a precaution.

9.6.3. Example DPDK configuration file

```
parameter_defaults:
  ComputeHCIParameters:
    KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=240 intel_iommu=on
iommu=pt                              # 1

isolcpus=2,46,3,47,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,49,51,53,55,57,59,61,63,
65,67,69,71,73,75,77,79,81,83,85,87"
    TunedProfileName: "cpu-partitioning"
    IsolCpusList:                          # 2
      ”2,46,3,47,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,49,51,
      53,55,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87"
    VhostuserSocketGroup: hugetlbfs
    OvsDpdkSocketMemory: "4096,4096"                # 3
    OvsDpdkMemoryChannels: "4"

    OvsPmdCoreList: "2,46,3,47"                # 4
    OvsDpdkCoreList: "0,44,1,45"               # 5
    NumDpdkInterfaceRxQueues: 1
```

1. KernelArgs: To calculate **hugepages**, subtract the value of the **NovaReservedHostMemory** parameter from total memory.

2. IsolCpusList: Assign a set of CPU cores that you want to isolate from the host processes with this parameter. Add the value of the **OvsPmdCoreList** parameter to the value of the **NovaVcpuPinSet** parameter to calculate the value for the **IsolCpusList** parameter.

3. OvsDpdkSocketMemory: Specify the amount of memory in MB to pre-allocate from the hugepage pool per NUMA node with the **OvsDpdkSocketMemory** parameter. For more information about calculating OVS-DPDK parameters, see: ovsdpdk parameters

4. OvsPmdCoreList: Specify the CPU cores that are used for the DPDK poll mode drivers (PMD) with this parameter. Choose CPU cores that are associated with the local NUMA nodes of the DPDK interfaces. Allocate 2 HT sibling threads for each NUMA node to calculate the value for the **OvsPmdCoreList** parameter.

5. OvsDpdkCoreList: Specify CPU cores for non-data path OVS-DPDK processes, such as handler, and revalidator threads, with this parameter. Allocate 2 HT sibling threads for each NUMA node to calculate the value for the **OvsDpdkCoreList** parameter.

9.6.4. Example nova configuration file

```
parameter_defaults:
  ComputeHCIExtraConfig:
    nova::cpu_allocation_ratio: 16 # 2
    NovaReservedHugePages:                              # 1
        - node:0,size:1GB,count:4
        - node:1,size:1GB,count:4
  NovaReservedHostMemory: 123904                          # 2
  # All left over cpus from NUMA-1
  NovaVcpuPinSet:                                 # 3
  ['5','7','9','11','13','15','17','19','21','23','25','27','29','31','33','35','37','39','41','43','49','51','|
  53','55','57','59','61','63','65','67','69','71','73','75','77','79','81','83','85','87
```

1. NovaReservedHugePages: Pre-allocate memory in MB from the hugepage pool with the **NovaReservedHugePages** parameter. It is the same memory total as the value for the **OvsDpdkSocketMemory** parameter.

2. NovaReservedHostMemory: Reserve memory in MB for tasks on the host with the **NovaReservedHostMemory** parameter. Use the following guidelines to calculate the amount of memory that you must reserve:

    - 5 GB for each OSD.
    - 0.5 GB overhead for each VM.
    - 4GB for general host processing. Ensure that you allocate sufficient memory to prevent potential performance degradation caused by cross-NUMA OSD operation.

3. NovaVcpuPinSet: List the CPUs not found in **OvsPmdCoreList**, **OvsDpdkCoreList**, or **Ceph_osd_docker_cpuset_cpus** with the **NovaVcpuPinSet** parameter. The CPUs must be in the same NUMA node as the DPDK NICs.

9.6.5. Recommended configuration for HCI-DPDK deployments

Table 9.1. Tunable parameters for HCI deployments

Block Device Type	OSDs, Memory, vCPUs per device
NVMe	Memory : 5GB per OSD OSDs per device: 4 vCPUs per device: 3
SSD	Memory : 5GB per OSD OSDs per device: 1 vCPUs per device: 4
HDD	Memory : 5GB per OSD OSDs per device: 1 vCPUs per device: 1

Use the same NUMA node for the following functions:

- Disk controller
- Storage networks
- Storage CPU and memory

Allocate another NUMA node for the following functions of the DPDK provider network:

- NIC
- PMD CPUs
- Socket memory

CHAPTER 10. EXAMPLE: CONFIGURING OVS-DPDK AND SR-IOV WITH VXLAN TUNNELLING

This section describes how to deploy Compute nodes with both OVS-DPDK and SR-IOV interfaces. The cluster will be installed with ML2/OVS and VXLAN tunnelling.

IMPORTANT

You must determine the best values for the OVS-DPDK parameters that you set in the **network-environment.yaml** file to optimize your OpenStack network for OVS-DPDK. See Deriving DPDK parameters with workflows for details.

10.1. CONFIGURING ROLES

Configure a custom role by copying and editing the default **roles_data.yaml** file found in /usr/share/openstack-tripleo-heat-templates.

For the purposes of this example, the **ComputeOvsDpdkSriov** role is created. For information on creating roles in Red Hat OpenStack Platform, see Advanced Overcloud Customization. For details on the specific role used for this example, see roles_data.yaml.

10.2. CONFIGURING OVS-DPDK PARAMETERS

IMPORTANT

You must determine the best values for the OVS-DPDK parameters that you set in the **network-environment.yaml** file to optimize your OpenStack network for OVS-DPDK. See Network Functions Virtualization Planning and Configuration for details.

1. Add the custom resources for OVS-DPDK under **resource_registry**:

```
  resource_registry:
    # Specify the relative/absolute path to the config files you want to use for override the default.
    OS::TripleO::ComputeOvsDpdkSriov::Net::SoftwareConfig:
      nic-configs/computeovsdpdksriov.yaml
    OS::TripleO::Controller::Net::SoftwareConfig:
      nic-configs/controller.yaml
```

2. Under **parameter_defaults**, set the tunnel type and network type to **vxlan**.:

```
NeutronTunnelTypes: 'vxlan'
NeutronNetworkType: 'vxlan'
```

3. Under **parameters_defaults**, set the bridge mapping:

```
# The OVS logical->physical bridge mappings to use.
NeutronBridgeMappings:
  - dpdk-mgmt:br-link0
```

4. Under **parameter_defaults**, set the role-specific parameters for the **ComputeOvsDpdkSriov** role:

```
  #########################
  # OVS DPDK configuration #
  # ########################
  ComputeOvsDpdkSriovParameters:
    KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=2-19,22-39"
    TunedProfileName: "cpu-partitioning"
    IsolCpusList: "2-19,22-39"
    NovaVcpuPinSet: ['4-19,24-39']
    NovaReservedHostMemory: 2048
    OvsDpdkSocketMemory: "3072,1024"
    OvsDpdkMemoryChannels: "4"
    OvsDpdkCoreList: "0,20,1,21"
    OvsPmdCoreList: "2,22,3,23"
    NovaLibvirtRxQueueSize: 1024
    NovaLibvirtTxQueueSize: 1024
```

NOTE

To prevent failures during guest creation, assign at least one CPU with sibling thread on each NUMA node. In the example, the values for the OvsPmdCoreList parameter denote cores 2 and 22 from NUMA 0, and cores 3 and 23 from NUMA 1.

NOTE

Huge pages are consumed by virtual machines, as well as OVS-DPDK using the **OvsDpdkSocketMemory** parameter. To calculate the number of huge pages available to the virtual machine, subtract the **OvsDpdkSocketMemory** value from the boot parameter value. You must also add **hw:mem_page_size=1GB** to the flavor you associate with the DPDK instance.

NOTE

OvsDPDKCoreList and **OvsDpdkMemoryChannels** are required settings for this procedure and must be set correctly to prevent failures.

5. Configure the role-specific parameters for SR-IOV:

```
  #########################
  #  SR-IOV configuration  #
  #########################
  NeutronMechanismDrivers: ['openvswitch','sriovnicswitch']
  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','Im
agePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthrough
Filter','NUMATopologyFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthrough_filter.PciPassthrough
Filter"]
  NovaPCIPassthrough:
```

```
 - vendor_id: "8086"
   product_id: "1528"
   address: "0000:06:00.0"
   trusted: "true"
   physical_network: "sriov-1"
 - vendor_id: "8086"
   product_id: "1528"
   address: "0000:06:00.1"
   trusted: "true"
   physical_network: "sriov-2"
```

10.3. CONFIGURING THE CONTROLLER NODE

1. Create the control plane Linux bond for an isolated network.

```
- type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  members:
  - type: interface
    name: nic2
    primary: true
```

2. Assign VLANs to this Linux bond.

```
- type: vlan
  vlan_id:
    get_param: InternalApiNetworkVlanID
  device: bond_api
  addresses:
  - ip_netmask:
      get_param: InternalApiIpSubnet

- type: vlan
  vlan_id:
    get_param: StorageNetworkVlanID
  device: bond_api
  addresses:
  - ip_netmask:
      get_param: StorageIpSubnet

- type: vlan
  vlan_id:
    get_param: StorageMgmtNetworkVlanID
  device: bond_api
  addresses:
  - ip_netmask:
      get_param: StorageMgmtIpSubnet

- type: vlan
  vlan_id:
    get_param: ExternalNetworkVlanID
  device: bond_api
  addresses:
  - ip_netmask:
      get_param: ExternalIpSubnet
  routes:
  - default: true
    next_hop:
      get_param: ExternalInterfaceDefaultRoute
```

3. Create the OVS bridge for access to neutron-dhcp-agent and neutron-metadata-agent services.

```
- type: ovs_bridge
  name: br-link0
  use_dhcp: false
  mtu: 9000
  members:
  - type: interface
    name: nic3
    mtu: 9000
  - type: vlan
    vlan_id:
      get_param: TenantNetworkVlanID
    mtu: 9000
    addresses:
    - ip_netmask:
        get_param: TenantIpSubnet
```

10.4. CONFIGURING THE COMPUTE NODE FOR DPDK AND SR-IOV

Create the **computeovsdpdksriov.yaml** file from the default **compute.yaml** file and make the following changes:

1. Create the control plane Linux bond for an isolated network.

```
- type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  members:
  - type: interface
    name: nic3
    primary: true
  - type: interface
    name: nic4
```

2. Assign VLANs to this Linux bond.

```
- type: vlan
  vlan_id:
    get_param: InternalApiNetworkVlanID
  device: bond_api
```

```
    addresses:
    - ip_netmask:
        get_param: InternalApiIpSubnet

  - type: vlan
    vlan_id:
      get_param: StorageNetworkVlanID
    device: bond_api
    addresses:
    - ip_netmask:
        get_param: StorageIpSubnet
```

3. Set a bridge with a DPDK port to link to the controller.

```
  - type: ovs_user_bridge
    name: br-link0
    use_dhcp: false
    ovs_extra:
      - str_replace:
          template: set port br-link0 tag=_VLAN_TAG_
          params:
            _VLAN_TAG_:
               get_param: TenantNetworkVlanID
    addresses:
      - ip_netmask:
          get_param: TenantIpSubnet
    members:
      - type: ovs_dpdk_bond
        name: dpdkbond0
        mtu: 9000
        rx_queue: 2
        members:
          - type: ovs_dpdk_port
            name: dpdk0
            members:
              - type: interface
                name: nic7
          - type: ovs_dpdk_port
            name: dpdk1
            members:
              - type: interface
                name: nic8
```

NOTE

To include multiple DPDK devices, repeat the **type** code section for each DPDK device you want to add.

NOTE

When using OVS-DPDK, **all** bridges on the same Compute node should be of type **ovs_user_bridge**. The director may accept the configuration, but Red Hat OpenStack Platform does not support mixing **ovs_bridge** and **ovs_user_bridge** on the same node.

10.5. DEPLOYING THE OVERCLOUD

Run the **overcloud_deploy.sh** script to deploy the overcloud.

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-r /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/roles_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/host-config-and-reboot.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/ovs-dpdk-
permissions.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/overcloud_images.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/network-
environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

CHAPTER 11. UPGRADING RED HAT OPENSTACK PLATFORM WITH NFV

For more information about upgrading Red Hat OpenStack Platform (RHOSP) with OVS-DPDK configured, see Preparing network functions virtualization (NFV) in the *Framework for Upgrades (13 to 16.1)* Guide. For more information about upgrading RHOSP, see the Upgrading Red Hat OpenStack Platform Guide.

CHAPTER 12. PERFORMANCE

Red Hat OpenStack Platform director configures the Compute nodes to enforce resource partitioning and fine tuning to achieve line rate performance for the guest virtual network functions (VNFs). The key performance factors in the network functions virtualization (NFV) use case are throughput, latency and jitter.

Data plane development kit (DPDK) accelerated Open vSwitch (OVS) enables high performance packet switching between physical NICs and virtual machines. OVS 2.7 embeds support for DPDK 16.11 and includes support for **vhost-user** multiqueue, allowing scalable performance. OVS-DPDK provides line rate performance for guest VNFs.

Single root I/O virtualization (SR-IOV) networking provides enhanced performance characteristics, including improved throughput for specific networks and virtual machines.

Other important features for performance tuning include huge pages, NUMA alignment, host isolation and CPU pinning. VNF flavors require huge pages and emulator thread isolation for better performance. Host isolation and CPU pinning improve NFV performance and prevent spurious packet loss.

See NFV Performance Considerations and Configure Emulator Threads to run on a Dedicated Physical CPU for a high-level introduction to CPUs and NUMA topology.

CHAPTER 13. FINDING MORE INFORMATION

The following table includes additional Red Hat documentation for reference:

The Red Hat OpenStack Platform documentation suite can be found here: Red Hat OpenStack Platform Documentation Suite

Table 13.1. List of Available Documentation

Component	Reference
Red Hat Enterprise Linux	Red Hat OpenStack Platform is supported on Red Hat Enterprise Linux 7.4. For information on installing Red Hat Enterprise Linux, see the corresponding installation guide at: Red Hat Enterprise Linux Documentation Suite.
Red Hat OpenStack Platform	To install OpenStack components and their dependencies, use the Red Hat OpenStack Platform director. The director uses a basic OpenStack installation as the undercloud to install, configure and manage the OpenStack nodes in the final overcloud. Be aware that you will need one extra host machine for the installation of the undercloud, in addition to the environment necessary for the deployed overcloud. For detailed instructions, see Red Hat OpenStack Platform Director Installation and Usage. For information on configuring advanced features for a Red Hat OpenStack Platform enterprise environment using the Red Hat OpenStack Platform director such as network isolation, storage configuration, SSL communication, and general configuration method, see Advanced Overcloud Customization.
NFV Documentation	For a high level overview of the NFV concepts, see the Network Functions Virtualization Product Guide.

APPENDIX A. SAMPLE DPDK SRIOV YAML FILES

This section provides sample YAML files as a reference for adding single root I/O virtualization (SR-IOV) and Data Plane Development Kit (DPDK) interfaces on the same compute node.

NOTE

These templates are from a fully configured environment and include parameters unrelated to NFV, that may not be relevant or appropriate for your deployment.

A.1. SAMPLE VXLAN DPDK SR-IOV YAML FILES

A.1.1. roles_data.yaml

```
###############################################################################
# File generated by TripleO                                                   #
###############################################################################
###############################################################################
# Role: Controller                                                            #
###############################################################################
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
  # For systems with both IPv4 and IPv6, you may specify a gateway network for
  # each, such as ['ControlPlane', 'External']
  default_route_networks: ['External']
  HostnameFormatDefault: 'controller-%index%'
  # Deprecated & backward-compatible values (FIXME: Make parameters consistent)
  # Set uses_deprecated_params to True if any deprecated params are used.
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::BarbicanBackendSimpleCrypto
- OS::TripleO::Services::BarbicanBackendDogtag
- OS::TripleO::Services::BarbicanBackendKmip
- OS::TripleO::Services::BarbicanBackendPkcs11Crypto
- OS::TripleO::Services::CACerts
- OS::TripleO::Services::CeilometerApi
- OS::TripleO::Services::CeilometerCollector
- OS::TripleO::Services::CeilometerExpirer
- OS::TripleO::Services::CeilometerAgentCentral
- OS::TripleO::Services::CeilometerAgentNotification
- OS::TripleO::Services::CephExternal
- OS::TripleO::Services::CephMds
- OS::TripleO::Services::CephMgr
- OS::TripleO::Services::CephMon
- OS::TripleO::Services::CephRbdMirror
- OS::TripleO::Services::CephRgw
- OS::TripleO::Services::CertmongerUser
- OS::TripleO::Services::CinderApi
- OS::TripleO::Services::CinderBackendDellPs
- OS::TripleO::Services::CinderBackendDellSc
- OS::TripleO::Services::CinderBackendDellEMCUnity
- OS::TripleO::Services::CinderBackendDellEMCVMAXISCSI
- OS::TripleO::Services::CinderBackendDellEMCVNX
- OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI
- OS::TripleO::Services::CinderBackendNetApp
- OS::TripleO::Services::CinderBackendScaleIO
- OS::TripleO::Services::CinderBackendVRTSHyperScale
- OS::TripleO::Services::CinderBackup
- OS::TripleO::Services::CinderHPELeftHandISCSI
- OS::TripleO::Services::CinderScheduler
- OS::TripleO::Services::CinderVolume
- OS::TripleO::Services::Clustercheck
- OS::TripleO::Services::Collectd
- OS::TripleO::Services::Congress
- OS::TripleO::Services::Docker
- OS::TripleO::Services::Ec2Api
- OS::TripleO::Services::Etcd
- OS::TripleO::Services::ExternalSwiftProxy
- OS::TripleO::Services::Fluentd
- OS::TripleO::Services::GlanceApi
- OS::TripleO::Services::GlanceRegistry
- OS::TripleO::Services::GnocchiApi
- OS::TripleO::Services::GnocchiMetricd
- OS::TripleO::Services::GnocchiStatsd
- OS::TripleO::Services::HAproxy
- OS::TripleO::Services::HeatApi
- OS::TripleO::Services::HeatApiCloudwatch
- OS::TripleO::Services::HeatApiCfn
- OS::TripleO::Services::HeatEngine
- OS::TripleO::Services::Horizon
- OS::TripleO::Services::Ipsec
- OS::TripleO::Services::IronicApi
- OS::TripleO::Services::IronicConductor
- OS::TripleO::Services::IronicPxe
- OS::TripleO::Services::Iscsid
- OS::TripleO::Services::Keepalived
- OS::TripleO::Services::Kernel
```

```
    - OS::TripleO::Services::Keystone
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::ManilaApi
    - OS::TripleO::Services::ManilaBackendCephFs
    - OS::TripleO::Services::ManilaBackendIsilon
    - OS::TripleO::Services::ManilaBackendNetapp
    - OS::TripleO::Services::ManilaBackendUnity
    - OS::TripleO::Services::ManilaBackendVNX
    - OS::TripleO::Services::ManilaBackendVMAX
    - OS::TripleO::Services::ManilaScheduler
    - OS::TripleO::Services::ManilaShare
    - OS::TripleO::Services::Memcached
    - OS::TripleO::Services::MistralApi
    - OS::TripleO::Services::MistralEngine
    - OS::TripleO::Services::MistralExecutor
    - OS::TripleO::Services::MistralEventEngine
    - OS::TripleO::Services::MongoDb
    - OS::TripleO::Services::MySQL
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronApi
    - OS::TripleO::Services::NeutronBgpVpnApi
    - OS::TripleO::Services::NeutronSfcApi
    - OS::TripleO::Services::NeutronCorePlugin
    - OS::TripleO::Services::NeutronDhcpAgent
    - OS::TripleO::Services::NeutronL2gwAgent
    - OS::TripleO::Services::NeutronL2gwApi
    - OS::TripleO::Services::NeutronL3Agent
    - OS::TripleO::Services::NeutronLbaasv2Agent
    - OS::TripleO::Services::NeutronLbaasv2Api
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronMetadataAgent
    - OS::TripleO::Services::NeutronML2FujitsuCfab
    - OS::TripleO::Services::NeutronML2FujitsuFossw
    - OS::TripleO::Services::NeutronOvsAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaApi
    - OS::TripleO::Services::NovaConductor
    - OS::TripleO::Services::NovaConsoleauth
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::NovaMetadata
    - OS::TripleO::Services::NovaPlacement
    - OS::TripleO::Services::NovaScheduler
    - OS::TripleO::Services::NovaVncProxy
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OctaviaApi
    - OS::TripleO::Services::OctaviaDeploymentConfig
    - OS::TripleO::Services::OctaviaHealthManager
    - OS::TripleO::Services::OctaviaHousekeeping
    - OS::TripleO::Services::OctaviaWorker
    - OS::TripleO::Services::OVNDBs
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::SkydiveAnalyzer
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::SwiftProxy
    - OS::TripleO::Services::SwiftDispersion
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::Ptp
###############################################################################
# Role: ComputeOvsDpdkSriov                                                   #
###############################################################################
- name: ComputeOvsDpdkSriov
  description: |
    Compute OvS DPDK+SR-IOV Role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
  HostnameFormatDefault: 'compute-%index%'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsDpdk
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::NeutronSriovHostConfig
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
  # Currently when attempting to deploy firewall service it notifies that it was specified multiple times
  # - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Ptp
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
```

A.1.2. network-environment.yaml

```
resource_registry:
  # Specify the relative/absolute path to the config files you want to use for override the default.
  OS::TripleO::ComputeOvsDpdkSriov::Net::SoftwareConfig: nic-configs/computeovsdpdksriov.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml

parameter_defaults:
  # Customize all these values to match the local environment
  InternalApiNetCidr: 10.10.10.0/24
  TenantNetCidr: 10.10.2.0/24
  StorageNetCidr: 10.10.3.0/24
  StorageMgmtNetCidr: 10.10.4.0/24
  ExternalNetCidr: 172.20.12.112/28
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  InternalApiAllocationPools: [{'start': '10.10.10.10', 'end': '10.10.10.200'}]
  TenantAllocationPools: [{'start': '10.10.2.100', 'end': '10.10.2.200'}]
  StorageAllocationPools: [{'start': '10.10.3.100', 'end': '10.10.3.200'}]
  StorageMgmtAllocationPools: [{'start': '10.10.4.100', 'end': '10.10.4.200'}]
  # Use an External allocation pool which will leave room for floating IPs
  ExternalAllocationPools: [{'start': '172.20.12.114', 'end': '172.20.12.125'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 172.20.12.126
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.168.24.1
  # Generally the IP of the Undercloud
  EC2MetadataIp: 192.168.24.1
  InternalApiNetworkVlanID: 10
  TenantNetworkVlanID: 11
  StorageNetworkVlanID: 12
  StorageMgmtNetworkVlanID: 13
  ExternalNetworkVlanID: 14
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["8.8.8.8","8.8.4.4"]
  # May set to br-ex if using floating IPs only on native VLAN on bridge br-ex
  NeutronExternalNetworkBridge: "''"
  # The tunnel type for the project network (vxlan or gre). Set to '' to disable tunneling.
  NeutronTunnelTypes: 'vxlan'
  # The project network type for Neutron (vlan or vxlan).
  NeutronNetworkType: 'vxlan,vlan'
  # The OVS logical->physical bridge mappings to use.
  NeutronBridgeMappings: 'dpdk-mgmt:br-link0'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'dpdk-mgmt:22:22,dpdk-mgmt:25:25,sriov-1:600:600,sriov-
2:601:601'
  # Nova flavor to use.
  OvercloudControllerFlavor: controller
  OvercloudComputeOvsDpdkSriovFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 3
  ComputeOvsDpdkSriovCount: 2
  # NTP server configuration.
  NtpServer: clock.redhat.com
  # MTU global configuration
  NeutronGlobalPhysnetMtu: 9000
  # Configure the classname of the firewall driver to use for implementing security groups.
  NeutronOVSFirewallDriver: openvswitch
  SshServerOptions:
    UseDns: 'no'

  ##########################
  # OVS DPDK configuration #
  # ########################
  ComputeOvsDpdkSriovParameters:
    KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=2-19,22-39"
    TunedProfileName: "cpu-partitioning"
    IsolCpusList: "2-19,22-39"
    NovaVcpuPinSet: ['4-19,24-39']
    NovaReservedHostMemory: 2048
    OvsDpdkSocketMemory: "3072,1024"
    OvsDpdkMemoryChannels: "4"
    OvsDpdkCoreList: "0,20,1,21"
    OvsPmdCoreList: "2,22,3,23"
    NovaLibvirtRxQueueSize: 1024
    NovaLibvirtTxQueueSize: 1024

  ##########################
  #  SR-IOV configuration  #
  ##########################
  NeutronMechanismDrivers: ['openvswitch','sriovnicswitch']
  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','ImageProp
ertiesFilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter','NUMATop
ologyFilter']
  NovaSchedulerAvailableFilters:
```

```
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter"]
  NovaPCIPassthrough:
    - vendor_id: "8086"
      product_id: "1528"
      address: "0000:06:00.0"
      physical_network: "sriov-1"
    - vendor_id: "8086"
      product_id: "1528"
      address: "0000:06:00.1"
      physical_network: "sriov-2"
  NeutronPhysicalDevMappings:
    - sriov1:p7p3
    - sriov2:p7p4
  NeutronSriovNumVFs: "p7p3:5,p7p4:5"
```

A.1.3. controller.yaml

```
heat_template_version: queens

description: >
  Software Config to drive os-net-config to configure VLANs for the
  controller role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  ExternalNetworkVlanID:
    default: ''
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: ''
    description: Vlan ID for the internal_api network traffic.
    type: number
  TenantNetworkVlanID:
    default: ''
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 23
    description: Vlan ID for the management network traffic.
    type: number
  ExternalInterfaceDefaultRoute:
    default: ''
    description: default route for the external network
    type: string
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
            addresses:
            - ip_netmask:
                list_join:
                - /
                - - get_param: ControlPlaneIp
                  - get_param: ControlPlaneSubnetCidr
            routes:
            - ip_netmask: 169.254.169.254/32
              next_hop:
                get_param: EC2MetadataIp

          - type: linux_bond
            name: bond_api
            bonding_options: "mode=active-backup"
            use_dhcp: false
            dns_servers:
              get_param: DnsServers
            members:
            - type: interface
              name: nic2
              primary: true

          - type: vlan
            vlan_id:
              get_param: InternalApiNetworkVlanID
            device: bond_api
            addresses:
            - ip_netmask:
                get_param: InternalApiIpSubnet

          - type: vlan
            vlan_id:
              get_param: StorageNetworkVlanID
            device: bond_api
            addresses:
            - ip_netmask:
                get_param: StorageIpSubnet

          - type: vlan
            vlan_id:
              get_param: StorageMgmtNetworkVlanID
            device: bond_api
            addresses:
            - ip_netmask:
                get_param: StorageMgmtIpSubnet

          - type: vlan
            vlan_id:
              get_param: ExternalNetworkVlanID
            device: bond_api
            addresses:
            - ip_netmask:
                get_param: ExternalIpSubnet
            routes:
            - default: true
              next_hop:
                get_param: ExternalInterfaceDefaultRoute

            - type: ovs_bridge
              name: br-link0
              use_dhcp: false
              mtu: 9000
              members:
              - type: interface
                name: nic3
                mtu: 9000
              - type: vlan
                vlan_id:
                  get_param: TenantNetworkVlanID
                mtu: 9000
                addresses:
                - ip_netmask:
                    get_param: TenantIpSubnet

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

A.1.4. compute-ovs-dpdk.yaml

```
heat_template_version: queens

description: >
  Software Config to drive os-net-config to configure VLANs for the
  compute role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  InternalApiNetworkVlanID:
    default: ''
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: ''
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 23
    description: Vlan ID for the management network traffic.
    type: number
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
  ExternalInterfaceDefaultRoute:
    default: ''
    description: default route for the external network
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh
          params:
```

```
$network_config:
  network_config:
  - type: interface
    name: nic1
    use_dhcp: false
    defroute: false

  - type: interface
    name: nic2
    use_dhcp: false
    addresses:
    - ip_netmask:
        list_join:
        - /
        - - get_param: ControlPlaneIp
          - get_param: ControlPlaneSubnetCidr
    routes:
    - ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: EC2MetadataIp
    - default: true
      next_hop:
        get_param: ControlPlaneDefaultRoute

  - type: linux_bond
    name: bond_api
    bonding_options: "mode=active-backup"
    use_dhcp: false
    dns_servers:
      get_param: DnsServers
    members:
    - type: interface
      name: nic3
      primary: true
    - type: interface
      name: nic4

  - type: vlan
    vlan_id:
      get_param: InternalApiNetworkVlanID
    device: bond_api
    addresses:
    - ip_netmask:
        get_param: InternalApiIpSubnet

  - type: vlan
    vlan_id:
      get_param: StorageNetworkVlanID
    device: bond_api
    addresses:
    - ip_netmask:
        get_param: StorageIpSubnet

  - type: ovs_user_bridge
    name: br-link0
    use_dhcp: false
```

```
                ovs_extra:
                  - str_replace:
                      template: set port br-link0 tag=_VLAN_TAG_
                      params:
                        _VLAN_TAG_:
                          get_param: TenantNetworkVlanID
                addresses:
                  - ip_netmask:
                      get_param: TenantIpSubnet
                members:
                  - type: ovs_dpdk_bond
                    name: dpdkbond0
                    mtu: 9000
                    rx_queue: 2
                    members:
                      - type: ovs_dpdk_port
                        name: dpdk0
                        members:
                          - type: interface
                            name: nic7
                      - type: ovs_dpdk_port
                        name: dpdk1
                        members:
                          - type: interface
                            name: nic8

              - type: interface
                name: p7p3
                mtu: 9000
                use_dhcp: false
                defroute: false
                nm_controlled: true
                hotplug: true

              - type: interface
                name: p7p4
                mtu: 9000
                use_dhcp: false
                defroute: false
                nm_controlled: true
                hotplug: true

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

A.1.5. overcloud_deploy.sh

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-r /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/roles_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/host-config-and-reboot.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/ovs-dpdk-
permissions.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/overcloud_images.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/network-
environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

APPENDIX B. REVISION HISTORY

Revision 1.6-0 March 07 2019
Revision of Chapter 6 for more detail on SR-IOV configuration.

Revision 1.5-0 January 14 2019
Sample templates are replaced with DPDK/SR-IOV with OVS/ML2

Revision 1.4-0 August 23 2018
Fixed parameter alignment for step 4 of `Configuring SR-IOV with OVS Hardware Offload with VLAN`.

Revision 1.3-0 August 20 2018
Added note about SKU requirement for RT-KVM repository.

Revision 1.2-0 July 31 2018
Updated network creation steps to use OSC parameters. Added description of BIOS settings.

Revision 1.1-0 July 12 2018
Added sample DPDK ODL yaml files and procedures.

Revision 1.0-0 June 27 2018
Initial version for Red Hat OpenStack 13 GA release. Includes procedures for RT-KVM and OVS HW offload. Supports ovs 2.9.