Red Hat OpenStack Platform 10

Network Functions Virtualization Configuration Guide

Configuring the Network Functions Virtualization (NFV) OpenStack Deployment

Last Updated: 2020-05-26

Red Hat OpenStack Platform 10 Network Functions Virtualization Configuration Guide

Configuring the Network Functions Virtualization (NFV) OpenStack Deployment

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide describes the configuration procedures for SR-IOV and OVS-DPDK in your Red Hat OpenStack Platform 10 with NFV deployment.

Table of Contents

PREFACE

Red Hat OpenStack Platform provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It offers a massively scalable, fault-tolerant platform for the development of cloud-enabled workloads.

This guide describes the steps to configure SR-IOV and DPDK-accelerated Open vSwitch (OVS) using the Red Hat OpenStack Platform 10 director for NFV deployments.

CHAPTER 1. OVERVIEW

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

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 the Network Functions Virtualization Planning Guide to understand the hardware and configuration options.

Red Hat OpenStack Platform 10 director allows you to isolate the overcloud networks (for example, external, tenant, internal API and so on). You can deploy a network on a single network interface or distributed over a multiple host network interface. Network isolation in a Red Hat OpenStack Platform 10 installation is configured using template files. If you do not provide template files, all the service networks are deployed on the provisioning network. There are multiple types of template configuration files:

- **network-environment.yaml** - Contains the network details such as subnets and IP address ranges that are used to configure the network on the overcloud nodes. This file also contains the different settings that override the default parameter values for various scenarios.
- Host templates (for example, **compute.yaml** , **controller.yaml** and so on) - Define the network interface configuration for the overcloud nodes.
- **first-boot.yaml** - Provides various configuration steps, for example:
  - Grub arguments.
  - DPDK parameters.
  - Tuned installation and configuration. The **tuned** package contains the **tuned** daemon that monitors the use of system components and dynamically tunes system settings based on that monitoring information. To provide proper CPU affinity configuration in OVS-DPDK and SR-IOV deployments, you should use the **tuned-cpu-partitioning** profile.

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

For samples of these heat template files for NFV, see the Sample YAML Files.

NOTE

NFV configuration makes use of YAML files. See YAML in a Nutshell for an introduction to the YAML file format.

The following sections provide more details on how to configure the heat template files for NFV using the Red Hat OpenStack Platform director.

1.1. COMPOSABLE ROLES

With Red Hat OpenStack Platform 10, you can use composable roles to create custom deployment roles for NFV. Composable roles allow you to add or remove services from each role. For more information on Composable Roles, see Composable Roles and Services.

To configure composable roles:

- Copy and modify the **roles-data.yaml** file to add the composable role for OVS-DPDK or SR-IOV.
- Create an OpenStack flavor and assign the appropriate properties to that flavor.
- Associate this new flavor with a node.
- Update the appropriate **network-environment.yaml** file to include parameters for kernel arguments and DPDK or SR-IOV arguments.
- Run the **overcloud_deploy.sh** script to deploy the overcloud with the composable roles.

CHAPTER 2. UPDATING RED HAT OPENSTACK PLATFORM WITH NFV

There are additional considerations and steps needed to update Red Hat OpenStack Platform when you have OVS-DPDK configured. The steps are covered in Director-Based Environments: Performing Updates to Minor Versions in the *Upgrading Red Hat OpenStack Platform Guide*.

CHAPTER 3. CONFIGURE SR-IOV SUPPORT FOR VIRTUAL NETWORKING

This chapter covers the configuration of Single Root Input/Output Virtualization (SR-IOV) within the Red Hat OpenStack Platform 10 environment using the director.

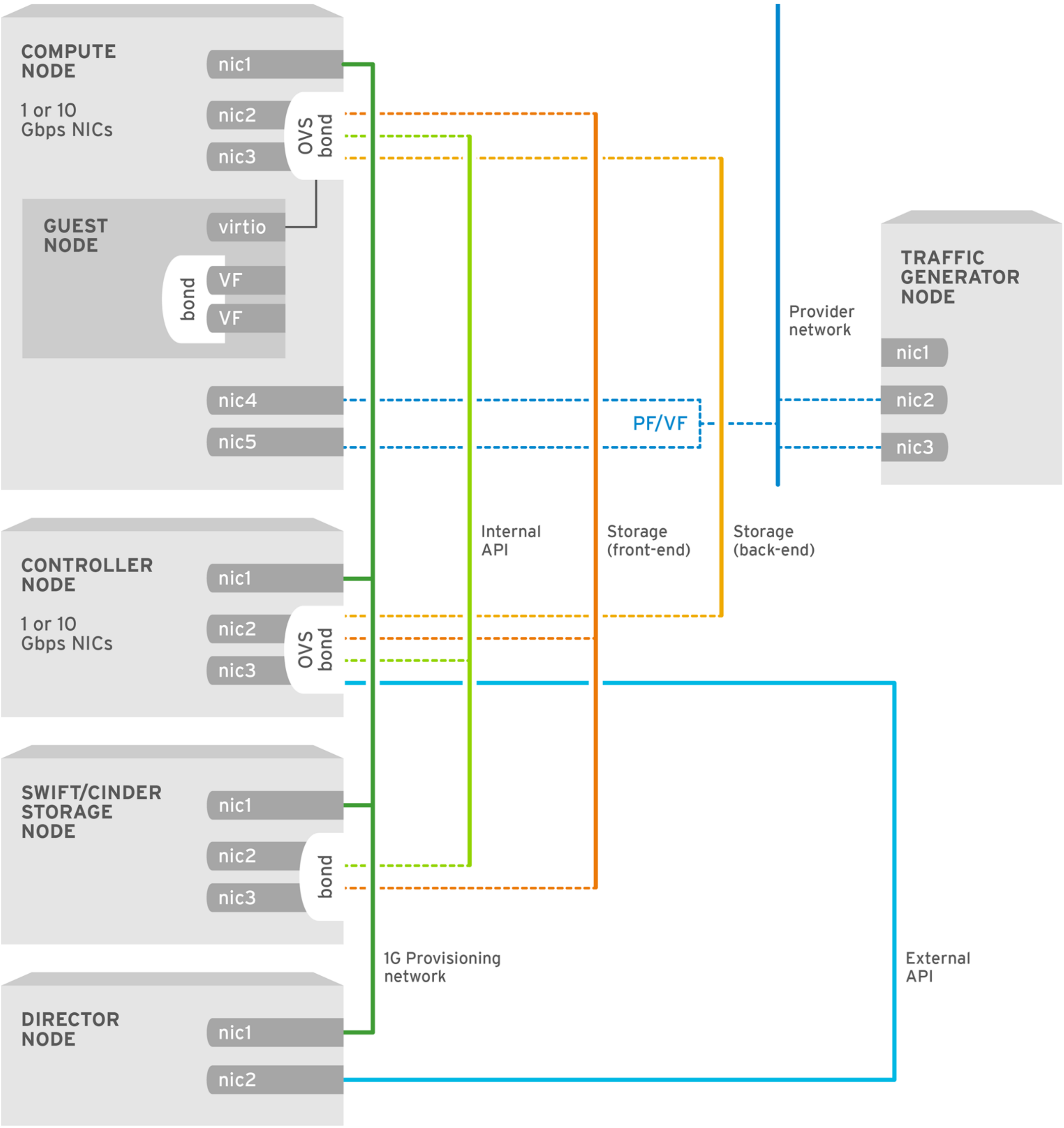
NOTE

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 the Network Functions Virtualization Planning Guide to understand the hardware and configuration options.

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.

In the following procedure, you need to update the **network-environment.yaml** file to include parameters for kernel arguments, SR-IOV driver, PCI passthrough and so on. You must also update the **compute.yaml** file to include the SR-IOV interface parameters, and run the **overcloud_deploy.sh** script to deploy the overcloud with the SR-IOV parameters.



OPENSTACK_450694_0617

3.1. CONFIGURE TWO-PORT SR-IOV WITH VLAN TUNNELLING

This section describes the YAML files you need to modify to configure SR-IOV with two ports that use VLAN tunnelling for your OpenStack environment.

3.1.1. Modify first-boot.yaml

NOTE

If you have included the following lines in the **first-boot.yaml** file in a previous deployment, remove these lines for Red Hat OpenStack Platform 10 with Open vSwitch 2.9.

```
ovs_service_path="/usr/lib/systemd/system/ovs-vswitchd.service"
grep -q "RuntimeDirectoryMode=.*" $ovs_service_path

if [ "$?" -eq 0 ]; then
    sed -i 's/RuntimeDirectoryMode=.*/RuntimeDirectoryMode=0775/' $ovs_service_path
else
    echo "RuntimeDirectoryMode=0775" >> $ovs_service_path
fi

grep -Fxq "Group=qemu" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "Group=qemu" >> $ovs_service_path
fi

grep -Fxq "UMask=0002" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "UMask=0002" >> $ovs_service_path
fi

ovs_ctl_path='/usr/share/openvswitch/scripts/ovs-ctl'
grep -q "umask 0002 \&\& start_daemon \"\$OVS_VSWITCHD_PRIORITY\"" $ovs_ctl_path

if [ ! "$?" -eq 0 ]; then
    sed -i 's/start_daemon \"\$OVS_VSWITCHD_PRIORITY.*/umask 0002 \&\& start_daemon
\"$OVS_VSWITCHD_PRIORITY\" \"$OVS_VSWITCHD_WRAPPER\" \"$@\"/' $ovs_ctl_path
fi
```

1. Set the **tuned** configuration to enable CPU affinity.

```
install_tuned:
  type: OS::Heat::SoftwareConfig
  properties:
    config:
      str_replace:
        template: |
          #!/bin/bash
          FORMAT=$COMPUTE_HOSTNAME_FORMAT
          if [[ -z $FORMAT ]] ; then
            FORMAT="compute" ;
          else
            # Assumption: only %index% and %stackname% are the variables in Host name
format
            FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
          fi
          if [[ $(hostname) == *$FORMAT* ]] ; then
            # Install the tuned package
            yum install -y tuned-profiles-cpu-partitioning

            tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
            if [ -n "$TUNED_CORES" ]; then
              grep -q "^isolated_cores" $tuned_conf_path
              if [ "$?" -eq 0 ]; then
                sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
              else
```

```
                  echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
                fi
                tuned-adm profile cpu-partitioning
              fi
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

2. Set the Kernel arguments:

```
  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name
format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/g' -i /etc/default/grub
;
              grub2-mkconfig -o /etc/grub2.cfg

              sleep 5
              reboot
            fi
          params:
            $KERNEL_ARGS: {get_param: ComputeKernelArgs}
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
```

3.1.2. Modify network-environment.yaml

1. Add **first-boot.yaml** under **resource_registry** to set the CPU tuning.

```
  resource_registry:
    # Specify the relative/absolute path to the config files you want to use for override the
default.
    OS::TripleO::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
    OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
    # First boot and Kernel Args
    OS::TripleO::NodeUserData: first-boot.yaml
```

2. Under **parameter_defaults**, disable the tunnel type (set the value to **""**), and set network type to **vlan**.

```
NeutronTunnelTypes: ''
NeutronNetworkType: 'vlan'
```

3. Under **parameter_defaults**, map the Open vSwitch physical network to the bridge.

   ```
   NeutronBridgeMappings: 'tenant:br-link0'
   ```

4. Under **parameter_defaults**, set the OpenStack Networking ML2 and Open vSwitch VLAN mapping range.

   ```
   NeutronNetworkVLANRanges: 'tenant:22:22,tenant:25:25'
   ```

5. Under **parameter_defaults**, set the SR-IOV configuration parameters.

   a. Enable the SR-IOV mechanism driver (**sriovnicswitch**).

   ```
   NeutronMechanismDrivers: "openvswitch,sriovnicswitch"
   ```

   b. Configure the Compute **pci_passthrough_whitelist** parameter, and set **devname** for the SR-IOV interface. The whitelist sets the PCI devices available to instances.

   ```
   NovaPCIPassthrough:
     - devname: "p7p1"
       physical_network: "tenant"
     - devname: "p7p2"
       physical_network: "tenant"
   ```

   c. Specify the physical network and SR-IOV interface in the format - **PHYSICAL_NETWORK:PHYSICAL DEVICE**.
   All physical networks listed in the **network_vlan_ranges** on the server should have mappings to the appropriate interfaces on each agent.

   ```
   NeutronPhysicalDevMappings: "tenant:p7p1,tenant:p7p2"
   ```

   d. Provide the number of Virtual Functions (VFs) to be reserved for each SR-IOV interface. Red Hat OpenStack Platform supports the number of VFs supported by the NIC vendor. See Deployment Limits for Red Hat OpenStack Platform for other related details. This example reserves 5 VFs for each of the SR-IOV interfaces:

   ```
   NeutronSriovNumVFs: "p7p1:5,p7p2:5"
   ```

   ### NOTE

   Changing the **NeutronSriovNumVFs** 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.

6. Under **parameter_defaults**, reserve the RAM for the host processes.

   ```
   NovaReservedHostMemory: 4096
   ```

7. Under **parameter_defaults**, set a comma-separated list or range of physical CPU cores to reserve for virtual machine processes.

```
NovaVcpuPinSet: "1-19,21-39"
```

8. Under **parameter_defaults**, list the the applicable filters.
   Nova scheduler applies these filters in the order they are listed. List the most restrictive filters first to make the filtering process for the nodes more efficient.

```
NovaSchedulerDefaultFilters:
  - "RetryFilter"
  - "AvailabilityZoneFilter"
  - "RamFilter"
  - "ComputeFilter"
  - "ComputeCapabilitiesFilter"
  - "ImagePropertiesFilter"
  - "ServerGroupAntiAffinityFilter"
  - "ServerGroupAffinityFilter"
  - "PciPassthroughFilter"
  - "NUMATopologyFilter"
  - "AggregateInstanceExtraSpecsFilter"
```

9. Under **parameter_defaults**, define the **ComputeKernelArgs** parameters to be included in the default **grub** file at first boot.

```
ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=12
intel_iommu=on iommu=pt"
```

NOTE

You need to add **hw:mem_page_size=1GB** to the flavor you associate with the DPDK instance. If you do not do this, the instance does not get a DHCP allocation.

10. Under **parameter_defaults**, set a list or range of physical CPU cores to be tuned.
    The given argument is appended to the tuned **cpu-partitioning** profile.

```
HostIsolatedCoreList: "1-19,21-39"
```

3.1.3. Modify controller.yaml

1. Create the 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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  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: ovs_bond
    name: bond0
    use_dhcp: true
    members:
      -
        type: interface
        name: nic7
        mtu: 9000
      -
        type: interface
        name: nic8
        mtu: 9000
```

3.1.4. Modify compute.yaml

1. Create the 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
      # force the MAC address of the bridge to this interface
      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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: bond_api
```

```
addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
```

3. Set the two SR-IOV interfaces by adding the following to the **compute.yaml** file.

```
-
 type: interface
 name: p7p1
 mtu: 9000
 use_dhcp: false
 defroute: false
 nm_controlled: true
 hotplug: true

-
 type: interface
 name: p7p2
 mtu: 9000
 use_dhcp: false
 defroute: false
 nm_controlled: true
 hotplug: true
```

3.1.5. Run the overcloud_deploy.sh Script

The following example defines the **openstack overcloud deploy** command for the VLAN environment.

```
openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml \
-e /home/stack/ospd-10-vlan-sriov-two-ports-ctlplane-bonding/network-environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml** is the location of the default **neutron-sriov.yaml** file, which enables the SR-IOV parameters in the Compute node.
- **/home/stack/<relative-directory>/network-environment.yaml** is the path for the **network-environment.yaml** file. The default **neutron-sriov.yaml** values can be overridden in **network-environment.yaml** file.

3.2. CREATE A FLAVOR AND DEPLOY AN INSTANCE FOR SR-IOV

After you have completed configuring SR-IOV for your Red Hat OpenStack Platform deployment with NFV, you need to create a flavor and deploy an instance by performing the following steps.

1. Create an aggregate group and add a host to it for SR-IOV. Define metadata, for example, **"aggregate_instance_extra_specs:sriov"="true"**, that matches flavor metadata.

```
# openstack aggregate create sriov_group
# openstack aggregate set --property \
"aggregate_instance_extra_specs:sriov"="true" sriov_group
```

```
# openstack aggregate add host sriov compute-sriov-0.localdomain
```

2. Create a flavor.

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

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

```
# openstack flavor set --property "aggregate_instance_extra_specs:sriov"="true" \
--property hw:cpu_policy=dedicated \
--property hw:mem_page_size=large <flavor>
```

4. Create the network.

```
# openstack network create net1 --provider-physical-network tenant --provider-network-type
vlan --provider-segment <VLAN-ID>
```

5. Create the subnet.

```
# openstack subnet create subnet1 --network net1 --subnet-range 192.0.2.0/24 --dhcp
```

6. Create the port.

   a. Use **vnic-type direct** to create an SR-IOV VF port.

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

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

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

7. Deploy an instance.

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

You have now deployed an instance for the SR-IOV with NFV use case.

CHAPTER 4. CONFIGURE DPDK ACCELERATED OPEN VSWITCH (OVS) FOR NETWORKING

This chapter covers DPDK with Open vSwitch installation and tuning within the Red Hat OpenStack Platform environment.

See Planning Your OVS-DPDK Deployment to understand the parameters used to configure OVS-DPDK.

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 the Network Functions Virtualization Planning Guide to understand the hardware and configuration options.

NOTE

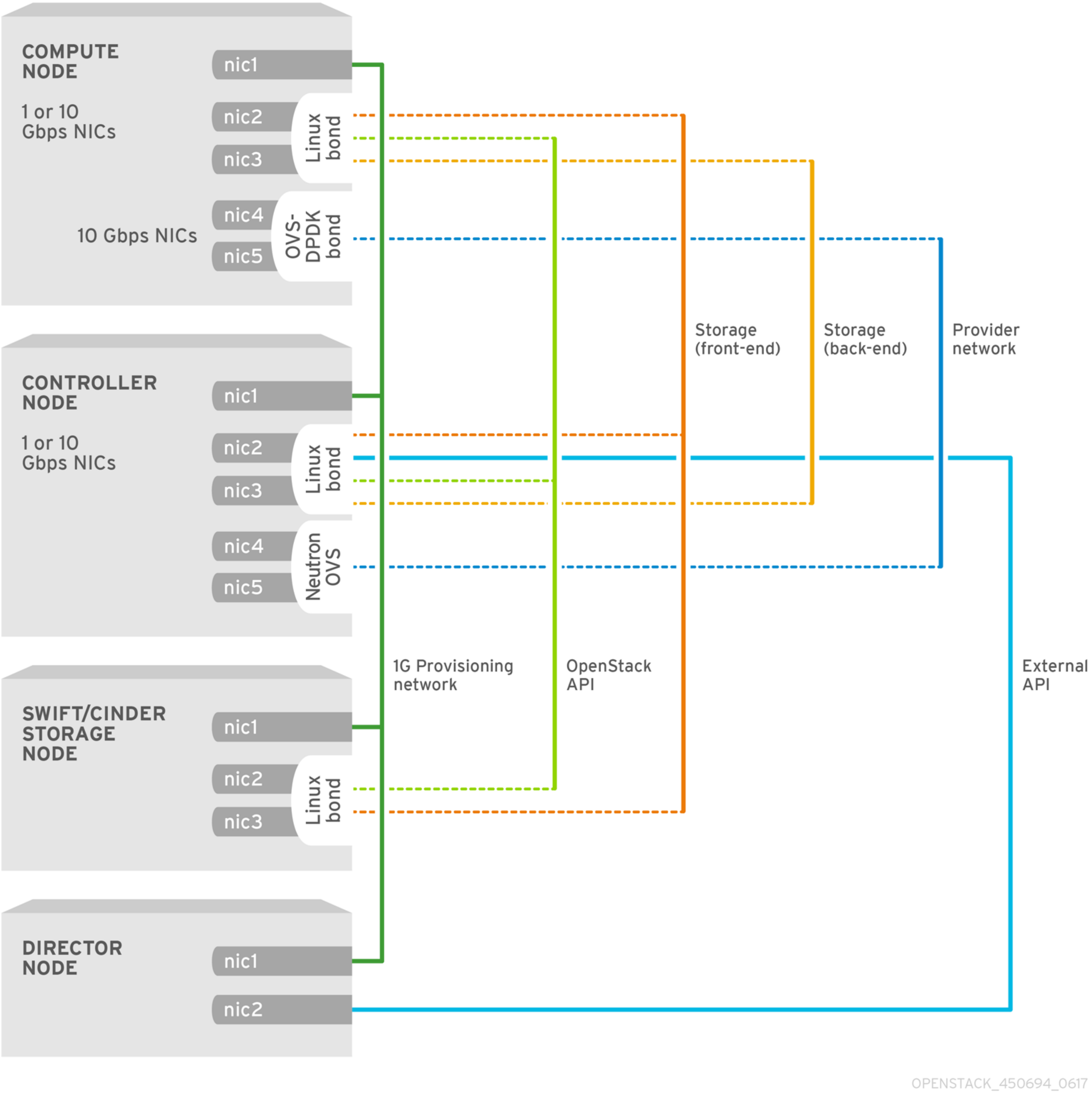
Do not edit or change **isolated_cores** or other values in **etc/tuned/cpu-partitioning-variables.conf** that are modified by these director heat templates.

In the following procedures, you need to:

- 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

For deployments that use hugepages, you also need to configure reserved_huge_pages. See How to set reserved_huge_pages in /etc/nova/nova.conf in Red Hat OpenStack Platform 10 for details.



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

- Red Hat OpenStack Platform 10 with Red Hat Enterprise Linux 7.5
- OVS-DPDK 2.9
- Tested NIC. For a list of tested NICs for NFV, see Tested NICs.

NOTE

Red Hat OpenStack Platform 10 with OVS 2.9 operates in OVS client mode for OVS-DPDK deployments.

4.1. NAMING CONVENTIONS

We recommend that you follow a consistent naming convention when you use custom roles in your OpenStack deployment, especially with multiple nodes. This naming convention can assist you when creating the following files and configurations:

- **instackenv.json** - To differentiate between nodes with different hardware or NIC capabilities.

  ```
  "name":"computeovsdpdk-0"
  ```

- **roles_data.yaml** - To differentiate between compute-based roles that support DPDK.

  ```
  `ComputeOvsDpdk`
  ```

- **network-environment.yaml** - To ensure that you match the custom role to the correct flavor name.

  ```
   `OvercloudComputeOvsDpdkFlavor: computeovsdpdk`
  ```

- **nic-config** file names - To differentiate NIC yaml files for compute nodes that support DPDK interfaces.

- Flavor creation - To help you match a flavor and **capabilities:profile** value to the appropriate bare metal node and custom role.

  ```
  # openstack flavor create --id auto --ram 4096 --disk 40 --vcpus 4 computeovsdpdk
  # openstack flavor set --property "cpu_arch"="x86_64" --property
  "capabilities:boot_option"="local" --property "capabilities:profile"="computeovsdpdk"
  computeovsdpdk
  ```

- Bare metal node - To ensure that you match the bare metal node with the appropriate hardware and **capability:profile** value.

  ```
   # openstack baremetal node update computeovsdpdk-0 add
  properties/capabilities='profile:computeovsdpdk,boot_option:local'
  ```

NOTE

The flavor name does not have to match the **capabilities:profile** value for the flavor, but the flavor **capabilities:profile** value must match the bare metal node **properties/capabilities='profile** value. All three use **computeovsdpdk** in this example.

NOTE

Ensure that all your nodes used for a custom role and profile have the same CPU, RAM, and PCI hardware topology.

4.2. CONFIGURE TWO-PORT OVS-DPDK DATA PLANE BONDING WITH VLAN TUNNELLING

This section covers the procedures to configure and deploy OVS-DPDK with two data plane ports in an OVS-DPDK bond, with control plane Linux bonding for your OpenStack environment.

4.2.1. Modify first-boot.yaml

Modify the **first-boot.yaml** file to set up OVS and DPDK parameters and to configure **tuned** for CPU affinity.

NOTE

If you have included the following lines in the **first-boot.yaml** file in a previous deployment, remove these lines for Red Hat OpenStack Platform 10 with Open vSwitch 2.9.

```
ovs_service_path="/usr/lib/systemd/system/ovs-vswitchd.service"
grep -q "RuntimeDirectoryMode=.*" $ovs_service_path

if [ "$?" -eq 0 ]; then
    sed -i 's/RuntimeDirectoryMode=.*/RuntimeDirectoryMode=0775/' $ovs_service_path
else
    echo "RuntimeDirectoryMode=0775" >> $ovs_service_path
fi

grep -Fxq "Group=qemu" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "Group=qemu" >> $ovs_service_path
fi

grep -Fxq "UMask=0002" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "UMask=0002" >> $ovs_service_path
fi

ovs_ctl_path='/usr/share/openvswitch/scripts/ovs-ctl'
grep -q "umask 0002 \&\& start_daemon \"\$OVS_VSWITCHD_PRIORITY\"" $ovs_ctl_path

if [ ! "$?" -eq 0 ]; then
    sed -i 's/start_daemon \"\$OVS_VSWITCHD_PRIORITY.*/umask 0002 \&\& start_daemon
\"$OVS_VSWITCHD_PRIORITY\" \"$OVS_VSWITCHD_WRAPPER\" \"$@\"/' $ovs_ctl_path
fi
```

1. Add additional resources.

   ```
   resources:
     userdata:
       type: OS::Heat::MultipartMime
       properties:
         parts:
         - config: {get_resource: set_dpdk_params}
         - config: {get_resource: install_tuned}
         - config: {get_resource: compute_kernel_args}
   ```

2. Set the DPDK parameters.

   ```
   set_dpdk_params:
     type: OS::Heat::SoftwareConfig
     properties:
       config:
         str_replace:
           template: |
             #!/bin/bash
   ```

```
        set -x
        get_mask()
        {
          local list=$1
          local mask=0
          declare -a bm
          max_idx=0
          for core in $(echo $list | sed 's/,/ /g')
          do
              index=$(($core/32))
              bm[$index]=0
              if [ $max_idx -lt $index ]; then
                 max_idx=$(($index))
              fi
          done
          for ((i=$max_idx;i>=0;i--));
          do
              bm[$i]=0
          done
          for core in $(echo $list | sed 's/,/ /g')
          do
              index=$(($core/32))
              temp=$((1<<$(($core % 32))))
              bm[$index]=$((${bm[$index]} | $temp))
          done

          printf -v mask "%x" "${bm[$max_idx]}"
          for ((i=$max_idx-1;i>=0;i--));
          do
              printf -v hex "%08x" "${bm[$i]}"
              mask+=$hex
          done
          printf "%s" "$mask"
        }

        FORMAT=$COMPUTE_HOSTNAME_FORMAT
        if [[ -z $FORMAT ]] ; then
          FORMAT="compute" ;
        else
          # Assumption: only %index% and %stackname% are the variables in Host name
format
          FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
        fi
        if [[ $(hostname) == *$FORMAT* ]] ; then
          # 42477 is the kolla hugetlbfs gid value.
          getent group hugetlbfs >/dev/null || \
            groupadd hugetlbfs -g 42477 && groupmod -g 42477 hugetlbfs

          pmd_cpu_mask=$( get_mask $PMD_CORES )
          host_cpu_mask=$( get_mask $LCORE_LIST )
          socket_mem=$(echo $SOCKET_MEMORY | sed s/\'//g )
          ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
          ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-
mem=$socket_mem
          ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-
mask=$pmd_cpu_mask
```

```
       ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-
mask=$host_cpu_mask
      fi
     params:
      $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
      $LCORE_LIST: {get_param: HostCpusList}
      $PMD_CORES: {get_param: NeutronDpdkCoreList}
      $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}
```

3. Set the **tuned** configuration to provide CPU affinity.

```
 install_tuned:
  type: OS::Heat::SoftwareConfig
  properties:
   config:
    str_replace:
     template: |
      #!/bin/bash
      FORMAT=$COMPUTE_HOSTNAME_FORMAT
      if [[ -z $FORMAT ]] ; then
       FORMAT="compute" ;
      else
       # Assumption: only %index% and %stackname% are the variables in Host name
format
       FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
      fi
      if [[ $(hostname) == *$FORMAT* ]] ; then
       # Install the tuned package
       yum install -y tuned-profiles-cpu-partitioning

       tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
       if [ -n "$TUNED_CORES" ]; then
        grep -q "^isolated_cores" $tuned_conf_path
        if [ "$?" -eq 0 ]; then
         sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
        else
         echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
        fi
        tuned-adm profile cpu-partitioning
       fi
      fi
     params:
      $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
      $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

4. Set the kernel arguments.

```
 compute_kernel_args:
  type: OS::Heat::SoftwareConfig
  properties:
   config:
    str_replace:
     template: |
      #!/bin/bash
      FORMAT=$COMPUTE_HOSTNAME_FORMAT
```

```
          if [[ -z $FORMAT ]] ; then
            FORMAT="compute" ;
          else
            # Assumption: only %index% and %stackname% are the variables in Host name
format
            FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
          fi
          if [[ $(hostname) == *$FORMAT* ]] ; then
            sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS
isolcpus=$TUNED_CORES"/g' -i /etc/default/grub ;
            grub2-mkconfig -o /etc/grub2.cfg
            reboot
          fi
        params:
          $KERNEL_ARGS: {get_param: ComputeKernelArgs}
          $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
          $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

4.2.2. Modify network-environment.yaml

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::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
    OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
    OS::TripleO::NodeUserData: first-boot.yaml
```

2. Under **parameter_defaults**, disable the tunnel type (set the value to **""**), and set the network type to **vlan**.

```
NeutronTunnelTypes: ''
NeutronNetworkType: 'vlan'
```

3. Under **parameter_defaults**, map the physical network to the virtual bridge.

```
NeutronBridgeMappings: 'tenant:br-link0'
```

4. Under **parameter_defaults**, set the OpenStack Networking ML2 and Open vSwitch VLAN mapping range.

```
NeutronNetworkVLANRanges: 'tenant:22:22,tenant:25:25'
```

This example sets the VLAN ranges on the physical network.

5. Under **parameter_defaults**, set the OVS-DPDK configuration parameters.

NOTE

NeutronDPDKCoreList and **NeutronDPDKMemoryChannels** are the **required** settings for this procedure. Attempting to deploy DPDK without appropriate values causes the deployment to fail or lead to unstable deployments.

a. Provide a list of cores that can be used as DPDK poll mode drivers (PMDs) in the format - **[allowed_pattern: "'[0-9,-]+'"]**.

```
NeutronDpdkCoreList: "'2,22,3,23'"
```

NOTE

You must assign at least one CPU (with sibling thread) on each NUMA node with or without DPDK NICs present for DPDK PMD to avoid failures in creating guest instances.

To optimize OVS-DPDK performance, consider the following options:

- Select CPUs associated with the NUMA node of the DPDK interface. Use **cat /sys/class/net/<interface>/device/numa_node** to list the NUMA node associated with an interface and use **lscpu** to list the CPUs associated with that NUMA node.
- Group CPU siblings together (in case of hyper-threading). Use **cat /sys/devices/system/cpu/<cpu>/topology/thread_siblings_list** to find the sibling of a CPU.
- Reserve CPU 0 for the host process.
- Isolate CPUs assigned to PMD so that the host process does not use these CPUs.
- Use **NovaVcpuPinset** to exclude CPUs assigned to PMD from Compute scheduling.

  a. Provide the number of memory channels in the format - **[allowed_pattern: "[0-9]+"]**.

  ```
  NeutronDpdkMemoryChannels: "4"
  ```

  b. Set the memory pre-allocated from the hugepage pool for each socket.

  ```
  NeutronDpdkSocketMemory: "'3072,1024'"
  ```

  This is a comma-separated string, in ascending order of the CPU socket. This example assumes a 2 NUMA node configuration and sets socket 0 to pre-allocate 1024 MB of huge pages, and sets socket 1 to pre-allocate 1024 MB. If you have a single NUMA node system, set this value to *1024,0*.

  c. Set the DPDK driver type for OVS bridges.

  ```
  NeutronDpdkDriverType: "vfio-pci"
  ```

6. Under **parameter_defaults**, set the vhost-user socket directory for OVS.

```
NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

7. Under **parameter_defaults**, reserve the RAM for the host processes.

```
NovaReservedHostMemory: 4096
```

8. Under **parameter_defaults**, set a comma-separated list or range of physical CPU cores to reserve for virtual machine processes.

```
NovaVcpuPinSet: "4-19,24-39"
```

9. Under **parameter_defaults**, list the the applicable filters.
   Nova scheduler applies these filters in the order they are listed. List the most restrictive filters first to make the filtering process for the nodes more efficient.

```
NovaSchedulerDefaultFilters:
  - "RetryFilter"
  - "AvailabilityZoneFilter"
  - "RamFilter"
  - "ComputeFilter"
  - "ComputeCapabilitiesFilter"
  - "ImagePropertiesFilter"
  - "ServerGroupAntiAffinityFilter"
  - "ServerGroupAffinityFilter"
  - "PciPassthroughFilter"
  - "NUMATopologyFilter"
  - "AggregateInstanceExtraSpecsFilter"
```

10. Under **parameter_defaults**, add the **ComputeKernelArgs** parameters to add these parameters to the default **grub** file at first boot.

```
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32
iommu=pt intel_iommu=on"
```

NOTE

These huge pages are consumed by the virtual machines, and also by OVS-DPDK using the **NeutronDpdkSocketMemory** parameter as shown in this procedure. The number of huge pages available for the virtual machines is the **boot** parameter minus the **NeutronDpdkSocketMemory**.

You need to add **hw:mem_page_size=1GB** to the flavor you associate with the DPDK instance. If you do not do this, the instance does not get a DHCP allocation.

11. Under **parameter_defaults**, set a list or range of physical CPU cores to be tuned.
    The given argument is appended to the tuned **cpu-partitioning** profile.

```
HostIsolatedCoreList: "2-19,22-39"
```

12. Under **parameters_default**, set the logical OVS-DPDK cores list. These cores must be mutually exclusive from the list of cores in **NeutronDpdkCoreList** and **NovaVcpuPinSet**.

```
HostCpusList: "'0,20,1,21'"
```

4.2.3. Modify controller.yaml

1. Create a separate provisioning interface.

```
network_config:
  -
    type: interface
    name: nic1
    use_dhcp: false
    defroute: false
  -
    type: interface
    name: nic2
    addresses:
      -
        ip_netmask:
          list_join:
            - '/'
            - - {get_param: ControlPlaneIp}
              - {get_param: ControlPlaneSubnetCidr}
    routes:
      -
        ip_netmask: 169.254.169.254/32
        next_hop: {get_param: EC2MetadataIp}
```

2. 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
```

3. 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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  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}
```

4. 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: ovs_bond
      name: bond0
      use_dhcp: true
      members:
        -
          type: interface
          name: nic7
          mtu: 9000
        -
          type: interface
          name: nic8
          mtu: 9000
```

4.2.4. Modify compute.yaml

Modify the default **compute.yaml** file and make the following changes:

1. Create a separate provisioning interface.

```
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}
```

2. 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
```

3. 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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
```

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

4. Set a bridge with two DPDK ports in an OVS-DPDK data plane bond to link to the controller.

```
-
  type: ovs_user_bridge
  name: br-link
  use_dhcp: false
  members:
    -
      type: ovs_dpdk_bond
      name: dpdkbond0
      mtu: 9000
      ovs_extra:
        - set interface dpdk0 mtu_request=$MTU
        - set interface dpdk1 mtu_request=$MTU
        - set interface dpdk0 options:n_rxq=2
        - set interface dpdk1 options:n_rxq=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.

4.2.5. Run the overcloud_deploy.sh Script

The following example defines the **openstack overcloud deploy** command for the OVS-DPDK environment within a bash script:

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ovs-dpdk-permissions.yaml \
-e /home/stack/ospd-10-vlan-dpdk-two-ports-ctlplane-dataplane-bonding/network-environment.yaml
```

- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml** is the location of the default **neutron-ovs-dpdk.yaml** file, which enables the OVS-DPDK parameters for the Compute role.

- **/home/stack/<relative-directory>/network-environment.yaml** is the path for the **network-environment.yaml** file. Use this file to overwrite the default values from the **neutron-ovs-dpdk.yaml** file.

NOTE

This configuration of OVS-DPDK does not support security groups and live migrations.

4.3. CONFIGURE SINGLE-PORT OVS-DPDK WITH VXLAN TUNNELLING

This section covers the procedures to configure single-port OVS-DPDK with control plane Linux bonding and VXLAN tunnelling for your OpenStack environment.

4.3.1. Modify first-boot.yaml

Modify the **first-boot.yaml** file to set up OVS and DPDK parameters and to configure **tuned** for CPU affinity.

NOTE

If you have included the following lines in the **first-boot.yaml** file in a previous deployment, remove these lines for Red Hat OpenStack Platform 10 with Open vSwitch 2.9.

```
ovs_service_path="/usr/lib/systemd/system/ovs-vswitchd.service"
grep -q "RuntimeDirectoryMode=.*" $ovs_service_path

if [ "$?" -eq 0 ]; then
   sed -i 's/RuntimeDirectoryMode=.*/RuntimeDirectoryMode=0775/' $ovs_service_path
else
   echo "RuntimeDirectoryMode=0775" >> $ovs_service_path
fi

grep -Fxq "Group=qemu" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
```

```
    echo "Group=qemu" >> $ovs_service_path
fi

grep -Fxq "UMask=0002" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "UMask=0002" >> $ovs_service_path
fi

ovs_ctl_path='/usr/share/openvswitch/scripts/ovs-ctl'
grep -q "umask 0002 \&\& start_daemon \"\$OVS_VSWITCHD_PRIORITY\"" $ovs_ctl_path

if [ ! "$?" -eq 0 ]; then
    sed -i 's/start_daemon \"\$OVS_VSWITCHD_PRIORITY.*/umask 0002 \&\& start_daemon
\"$OVS_VSWITCHD_PRIORITY\" \"$OVS_VSWITCHD_WRAPPER\" \"$@\"/' $ovs_ctl_path
fi
```

1. Add additional resources.

```
resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}
```

2. Set the DPDK parameters.

```
set_dpdk_params:
  type: OS::Heat::SoftwareConfig
  properties:
    config:
      str_replace:
        template: |
          #!/bin/bash
          set -x
          get_mask()
          {
            local list=$1
            local mask=0
            declare -a bm
            max_idx=0
            for core in $(echo $list | sed 's/,/ /g')
            do
                index=$(($core/32))
                bm[$index]=0
                if [ $max_idx -lt $index ]; then
                   max_idx=$(($index))
                fi
            done
            for ((i=$max_idx;i>=0;i--));
            do
                bm[$i]=0
```

```
       done
       for core in $(echo $list | sed 's/,/ /g')
       do
         index=$(($core/32))
         temp=$((1<<$(($core % 32))))
         bm[$index]=$((${bm[$index]} | $temp))
       done

       printf -v mask "%x" "${bm[$max_idx]}"
       for ((i=$max_idx-1;i>=0;i--));
       do
         printf -v hex "%08x" "${bm[$i]}"
         mask+=$hex
       done
       printf "%s" "$mask"
      }

      FORMAT=$COMPUTE_HOSTNAME_FORMAT
      if [[ -z $FORMAT ]] ; then
       FORMAT="compute" ;
      else
       # Assumption: only %index% and %stackname% are the variables in Host name
format
       FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
      fi
      if [[ $(hostname) == *$FORMAT* ]] ; then
       # 42477 is the kolla hugetlbfs gid value.
       getent group hugetlbfs >/dev/null || \
        groupadd hugetlbfs -g 42477 && groupmod -g 42477 hugetlbfs

       pmd_cpu_mask=$( get_mask $PMD_CORES )
       host_cpu_mask=$( get_mask $LCORE_LIST )
       socket_mem=$(echo $SOCKET_MEMORY | sed s/\'//g )
       ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
       ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-
mem=$socket_mem
       ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-
mask=$pmd_cpu_mask
       ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-
mask=$host_cpu_mask
      fi
     params:
      $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
      $LCORE_LIST: {get_param: HostCpusList}
      $PMD_CORES: {get_param: NeutronDpdkCoreList}
      $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}
```

3. Set the **tuned** configuration to provide CPU affinity.

```
install_tuned:
 type: OS::Heat::SoftwareConfig
 properties:
  config:
   str_replace:
    template: |
     #!/bin/bash
```

```
          FORMAT=$COMPUTE_HOSTNAME_FORMAT
          if [[ -z $FORMAT ]] ; then
            FORMAT="compute" ;
          else
            # Assumption: only %index% and %stackname% are the variables in Host name
format
            FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
          fi
          if [[ $(hostname) == *$FORMAT* ]] ; then
            # Install the tuned package
            yum install -y tuned-profiles-cpu-partitioning

            tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
            if [ -n "$TUNED_CORES" ]; then
              grep -q "^isolated_cores" $tuned_conf_path
              if [ "$?" -eq 0 ]; then
                sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
              else
                echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
              fi
              tuned-adm profile cpu-partitioning
            fi
          fi
        params:
          $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
          $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

4. Set the kernel arguments.

```
compute_kernel_args:
  type: OS::Heat::SoftwareConfig
  properties:
    config:
      str_replace:
        template: |
          #!/bin/bash
          FORMAT=$COMPUTE_HOSTNAME_FORMAT
          if [[ -z $FORMAT ]] ; then
            FORMAT="compute" ;
          else
            # Assumption: only %index% and %stackname% are the variables in Host name
format
            FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
          fi
          if [[ $(hostname) == *$FORMAT* ]] ; then
            sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS
isolcpus=$TUNED_CORES"/g' -i /etc/default/grub ;
            grub2-mkconfig -o /etc/grub2.cfg
            reboot
          fi
        params:
          $KERNEL_ARGS: {get_param: ComputeKernelArgs}
          $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
          $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

4.3.2. Modify network-environment.yaml

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::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
   OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
   OS::TripleO::NodeUserData: first-boot.yaml
```

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

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

3. Under **parameter_defaults**, set the OVS-DPDK configuration parameters.

NOTE

NeutronDPDKCoreList and **NeutronDPDKMemoryChannels** are the **required** settings for this procedure. Attempting to deploy DPDK without appropriate values causeS the deployment to fail or lead to unstable deployments.

a. Provide a list of cores that can be used as DPDK poll mode drivers (PMDs) in the format - **[allowed_pattern: "'[0-9,-]+'"]**.

```
NeutronDpdkCoreList: "'2,22,3,23'"
```

NOTE

You must assign at least one CPU (with sibling thread) on each NUMA node with or without DPDK NICs present for DPDK PMD to avoid failures in creating guest instances.

To optimize OVS-DPDK performance, consider the following options:

- Select CPUs associated with the NUMA node of the DPDK interface. Use **cat /sys/class/net/<interface>/device/numa_node** to list the NUMA node associated with an interface and use **lscpu** to list the CPUs associated with that NUMA node.
- Group CPU siblings together (in case of hyper-threading). Use **cat /sys/devices/system/cpu/<cpu>/topology/thread_siblings_list** to find the sibling of a CPU.
- Reserve CPU 0 for the host process.
- Isolate CPUs assigned to PMD so that the host process does not use these CPUs.
- Use **NovaVcpuPinset** to exclude CPUs assigned to PMD from Compute scheduling.

a. Provide the number of memory channels in the format - **[allowed_pattern: "[0-9]+"]**.

```
NeutronDpdkMemoryChannels: "4"
```

b. Set the memory pre-allocated from the hugepage pool for each socket.

```
NeutronDpdkSocketMemory: "'3072,1024'"
```

This is a comma-separated string, in ascending order of the CPU socket. If you have a single NUMA node system, set this value to *3072,0*.

c. Set the DPDK driver type for OVS bridges.

```
NeutronDpdkDriverType: "vfio-pci"
```

4. Under **parameter_defaults**, set the vhost-user socket directory for OVS.

```
NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

5. Under **parameter_defaults**, reserve the RAM for the host processes.

```
NovaReservedHostMemory: 4096
```

6. Under **parameter_defaults**, set a comma-separated list or range of physical CPU cores to reserve for virtual machine processes.

```
  NovaVcpuPinSet: "4-19,24-39"
```

7. Under **parameter_defaults**, list the the applicable filters.
Nova scheduler applies these filters in the order they are listed. List the most restrictive filters first to make the filtering process for the nodes more efficient.

```
  NovaSchedulerDefaultFilters:
    - "RetryFilter"
    - "AvailabilityZoneFilter"
    - "RamFilter"
    - "ComputeFilter"
    - "ComputeCapabilitiesFilter"
    - "ImagePropertiesFilter"
    - "ServerGroupAntiAffinityFilter"
    - "ServerGroupAffinityFilter"
    - "PciPassthroughFilter"
    - "NUMATopologyFilter"
    - "AggregateInstanceExtraSpecsFilter"
```

8. Under **parameter_defaults**, add the **ComputeKernelArgs** parameters to add these parameters to the default **grub** file at first boot.

```
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32
iommu=pt intel_iommu=on"
```

NOTE

These huge pages are consumed by the virtual machines, and also by OVS-DPDK using the **NeutronDpdkSocketMemory** parameter as shown in this procedure. The number of huge pages available for the virtual machines is the **boot** parameter minus the **NeutronDpdkSocketMemory**.

You need to add **hw:mem_page_size=1GB** to the flavor you associate with the DPDK instance. If you do not do this, the instance does not get a DHCP allocation.

9. Under **parameter_defaults**, set a list or range of physical CPU cores to be tuned. The given argument is appended to the tuned **cpu-partitioning** profile.

```
HostIsolatedCoreList: "2-19,22-39"
```

10. Under **parameters_default**, set the logical OVS-DPDK cores list. These cores must be mutually exclusive from the list of cores in **NeutronDpdkCoreList** and **NovaVcpuPinSet**.

```
HostCpusList: "'0,20,22-39'"
```

4.3.3. Modify controller.yaml

1. Create a separate provisioning interface.

```
network_config:
  -
    type: interface
    name: nic1
    use_dhcp: false
    defroute: false
  -
    type: interface
    name: nic2
    addresses:
      -
        ip_netmask:
          list_join:
            - '/'
            - - {get_param: ControlPlaneIp}
              - {get_param: ControlPlaneSubnetCidr}
    routes:
      -
        ip_netmask: 169.254.169.254/32
        next_hop: {get_param: EC2MetadataIp}
```

2. 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
```

3. 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}
```

4. 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: ovs_bond
      name: bond0
      use_dhcp: true
      members:
        -
          type: interface
          name: nic7
          mtu: 9000
        -
          type: interface
          name: nic8
          mtu: 9000
    -
      type: vlan
      vlan_id: {get_param: TenantNetworkVlanID}
      device: bond0
      mtu: 9000
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
```

4.3.4. Modify compute.yaml

Create the **compute-ovs-dpdk.yaml** file from the default **compute.yaml** file and make the following changes:

1. Create a separate provisioning interface.

```
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}
```

2. 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
```

3. 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}
```

4. 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
      ovs_extra:
        - set interface dpdk0 mtu_request=$MTU
        - set interface dpdk1 mtu_request=$MTU
        - set interface dpdk0 options:n_rxq=2
  - set interface dpdk1 options:n_rxq=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.

4.3.5. Run the overcloud_deploy.sh Script

The following example defines the **openstack overcloud deploy** command for the OVS-DPDK environment within a Bash script:

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ovs-dpdk-permissions.yaml \
-e /home/stack/ospd-10-vxlan-dpdk-single-port-ctlplane-bonding/network-environment.yaml
```

- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml** is the location of the default **neutron-ovs-dpdk.yaml** file, which enables the OVS-DPDK parameters for the Compute role.

- **/home/stack/<relative-directory>/network-environment.yaml** is the path for the **network-environment.yaml** file. Use this file to overwrite the default values from the **neutron-ovs-dpdk.yaml** file.

NOTE

This configuration of OVS-DPDK does not support security groups and live migrations.

4.4. SET THE MTU VALUE FOR OVS-DPDK INTERFACES

Red Hat OpenStack Platform supports jumbo frames for OVS-DPDK. To set the 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:
  # Global MTU configuration on Neutron
  NeutronGlobalPhysnetMtu: 9000
```

NOTE

Ensure that the NeutronDpdkSocketMemory value in the **network-environment.yaml** file is large enough to support jumbo frames. See 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
  mtu: 9000
  members:
    -
      type: ovs_bond
      name: bond0
      use_dhcp: true
      members:
        -
          type: interface
```

```
              name: nic7
              mtu: 9000
            -
              type: interface
              name: nic8
              mtu: 9000
        -
          type: vlan
          vlan_id: {get_param: TenantNetworkVlanID}
          device: bond0
          mtu: 9000
          addresses:
            -
              ip_netmask: {get_param: TenantIpSubnet}
```

To set the MTU values for the OVS-DPDK interfaces and bonds in the **compute.yaml** file:

```
-
  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
      ovs_extra:
        - set interface dpdk0 mtu_request=$MTU
        - set interface dpdk1 mtu_request=$MTU
        - set interface dpdk0 options:n_rxq=2
        - set interface dpdk1 options:n_rxq=2
      members:
        -
          type: ovs_dpdk_port
          name: dpdk0
          members:
            -
              type: interface
              name: nic7
        -
          type: ovs_dpdk_port
          name: dpdk1
          members:
            -
              type: interface
              name: nic8
```

4.5. SET MULTIQUEUE FOR OVS-DPDK INTERFACES

To set the number of queues for an OVS-DPDK port on the Compute node, modify the **compute.yaml** file as follows:

```
-
  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
      ovs_extra:
        - set interface dpdk0 mtu_request=$MTU
        - set interface dpdk1 mtu_request=$MTU
        - set interface dpdk0 options:n_rxq=2
        - set interface dpdk1 options:n_rxq=2
      members:
        -
          type: ovs_dpdk_port
          name: dpdk0
          members:
            -
              type: interface
              name: nic7
        -
          type: ovs_dpdk_port
          name: dpdk1
          members:
            -
              type: interface
              name: nic8
```

4.6. KNOWN LIMITATIONS

There are certain limitations when configuring OVS-DPDK with Red Hat OpenStack Platform 10 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 will appear but not function.

- There is a performance degradation of services that use tap devices, because these devices do not support DPDK. For example, services such as DVR, FWaaS, and LBaaS use tap devices.
    - With OVS-DPDK, you can enable DVR with **netdev datapath**, but this has poor performance and is not suitable for a production environment. DVR uses kernel namespace and tap devices to perform the routing.
    - To ensure the DVR routing performs well with OVS-DPDK, you need to use a controller such as ODL which implements routing as OpenFlow rules. With OVS-DPDK, OpenFlow routing removes the bottleneck introduced by the Linux kernel interfaces so that the full performance of datapath is maintained.
- When using OVS-DPDK, **all** bridges should be of type **ovs_user_bridge** on the Compute node. The director may accept the configuration, but Red Hat OpenStack Platform does not support mixing **ovs_bridge** and **ovs_user_bridge**.

4.7. CREATE A FLAVOR AND DEPLOY AN INSTANCE FOR OVS-DPDK

After you have completed configuring 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 a host to it for OVS-DPDK. Define metadata, for example, **"aggregate_instance_extra_specs:dpdk"="true"**, that matches flavor metadata.

    ```
    # openstack aggregate create dpdk_group
    # openstack aggregate set --property \
    "aggregate_instance_extra_specs:dpdk"="true" dpdk_group
    # openstack aggregate add host dpdk compute-ovs-dpdk-0.localdomain
    ```

2. Create a flavor.

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

3. Set additional flavor properties. Note that the defined metadata, **"aggregate_instance_extra_specs:dpdk"=true"**, matches the defined metadata on the DPDK aggregate.

    ```
    # openstack flavor set --property "aggregate_instance_extra_specs:dpdk"="true" \
    --property hw:cpu_policy=dedicated  \
    --property hw:mem_page_size=large <flavor>
    ```

4. Create the network.

    ```
    # openstack network create net1 --provider-physical-network tenant --provider-network-type
    vlan --provider-segment <VLAN-ID>
    ```

5. Create the subnet.

    ```
    # openstack subnet create subnet1 --network net1 --subnet-range 192.0.2.0/24 --dhcp
    ```

6. Deploy an instance.

    ```
    # openstack server create --flavor <flavor> --image <glance_image> --nic net-id=net1
    <name>
    ```

You have now deployed an instance for the OVS-DPDK with NFV use case.

4.7.1. Optimizing Performance with Emulator Thread Pinning

To improve performance, you can pin the Qemu emulator thread to an alternate core.

1. Determine which cores are used as vCPUs for your instance.

```
# virsh dumpxml dpdk_vm | grep cpuset
    <vcpupin vcpu='0' cpuset='2'/>
    <vcpupin vcpu='1' cpuset='18'/>
    <vcpupin vcpu='2' cpuset='1'/>
    <vcpupin vcpu='3' cpuset='17'/>
    <emulatorpin cpuset='1-2,17-18'/>
```

2. Select the core you want to pin the emulator thread to. Ensure the selected core is from the NovaVcpuPinSet.

```
#virsh emulatorpin <vm-name> --cpulist 2
```

NOTE

The pCPU associated with the emulator pin thread consumes one vCPU (two threads if hyperthreading is enabled) from the **NovaVcpuPinSet**.

4.8. TROUBLESHOOTING THE CONFIGURATION

This section describes the steps to troubleshoot the DPDK-OVS configuration.

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

```
# 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. Review the OVS service by confirming that the **neutron-ovs-agent** is configured to start automatically.

```
# systemctl status neutron-openvswitch-agent.service
neutron-openvswitch-agent.service - OpenStack Neutron Open vSwitch Agent
Loaded: loaded (/usr/lib/systemd/system/neutron-openvswitch-agent.service; enabled;
vendor preset: disabled)
Active: active (running) since Mon 2015-11-23 14:49:31 AEST; 25min ago
```

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

```
# journalctl -t neutron-openvswitch-agent.service
```

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 the 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 5. CONFIGURING SR-IOV AND DPDK INTERFACES ON THE SAME COMPUTE NODE

This section describes how to deploy SR-IOV and DPDK interfaces on the same Compute node.

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 the Network Functions Virtualization Planning Guide to understand the hardware and configuration options.

The process to create and deploy SR-IOV and DPDK interfaces on the same Compute node includes:

- Set the parameters for SR-IOV role and OVS-DPDK in the **network-environment.yaml** file.
- Configure the **compute.yaml** file with an SR-IOV interface and a DPDK interface.
- Deploy the overcloud with this updated set of roles.
- Create the appropriate OpenStack flavor, networks, and ports to support these interface types.

We recommend the following network settings:

- Use floating IP addresses for the guest instances.
- Create a router and attach it to the DPDK VXLAN network (the management network).
- Use SR-IOV for the provider network.
- Boot the guest instance with two ports attached. We recommend you use **cloud-init** for the guest instance to set the default route for the management network.
- Add the floating IP address to booted guest instance.

NOTE

If needed, use SR-IOV bonding for the guest instance and ensure both SR-IOV interfaces exist on the same NUMA node for optimum performance.

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

NOTE

Ensure that you create an OpenStack flavor that match this custom role.

5.1. MODIFYING THE FIRST-BOOT.YAML FILE

Modify the first-boot.yaml file to set up OVS and DPDK parameters and to configure **tuned** for CPU affinity.

NOTE

If you have included the following lines in the **first-boot.yaml** file in a previous deployment, remove these lines for Red Hat OpenStack Platform 10 with Open vSwitch 2.9.

```
ovs_service_path="/usr/lib/systemd/system/ovs-vswitchd.service"
grep -q "RuntimeDirectoryMode=.*" $ovs_service_path

if [ "$?" -eq 0 ]; then
    sed -i 's/RuntimeDirectoryMode=.*/RuntimeDirectoryMode=0775/' $ovs_service_path
else
    echo "RuntimeDirectoryMode=0775" >> $ovs_service_path
fi

grep -Fxq "Group=qemu" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "Group=qemu" >> $ovs_service_path
fi

grep -Fxq "UMask=0002" $ovs_service_path

if [ ! "$?" -eq 0 ]; then
    echo "UMask=0002" >> $ovs_service_path
fi

ovs_ctl_path='/usr/share/openvswitch/scripts/ovs-ctl'
grep -q "umask 0002 \&\& start_daemon \"\$OVS_VSWITCHD_PRIORITY\"" $ovs_ctl_path

if [ ! "$?" -eq 0 ]; then
    sed -i 's/start_daemon \"\$OVS_VSWITCHD_PRIORITY.*/umask 0002 \&\& start_daemon
\"$OVS_VSWITCHD_PRIORITY\" \"$OVS_VSWITCHD_WRAPPER\" \"$@\"/' $ovs_ctl_path
fi
```

1. Add additional resources.

```
resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}
```

2. Set the DPDK parameters.

```
set_dpdk_params:
  type: OS::Heat::SoftwareConfig
  properties:
    config:
      str_replace:
        template: |
          #!/bin/bash
        set -x
        get_mask()
        {
          local list=$1
          local mask=0
          declare -a bm
          max_idx=0
          for core in $(echo $list | sed 's/,/ /g')
          do
              index=$(($core/32))
              bm[$index]=0
              if [ $max_idx -lt $index ]; then
                 max_idx=$(($index))
              fi
          done
          for ((i=$max_idx;i>=0;i--));
          do
              bm[$i]=0
          done
          for core in $(echo $list | sed 's/,/ /g')
          do
              index=$(($core/32))
              temp=$((1<<$(($core % 32))))
              bm[$index]=$((${bm[$index]} | $temp))
          done

          printf -v mask "%x" "${bm[$max_idx]}"
          for ((i=$max_idx-1;i>=0;i--));
          do
              printf -v hex "%08x" "${bm[$i]}"
              mask+=$hex
          done
          printf "%s" "$mask"
        }

        FORMAT=$COMPUTE_HOSTNAME_FORMAT
        if [[ -z $FORMAT ]] ; then
          FORMAT="compute" ;
        else
          # Assumption: only %index% and %stackname% are the variables in Host name
format
          FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
        fi
        if [[ $(hostname) == *$FORMAT* ]] ; then
          # 42477 is the kolla hugetlbfs gid value.
          getent group hugetlbfs >/dev/null || \
            groupadd hugetlbfs -g 42477 && groupmod -g 42477 hugetlbfs

          pmd_cpu_mask=$( get_mask $PMD_CORES )
          host_cpu_mask=$( get_mask $LCORE_LIST )
          socket_mem=$(echo $SOCKET_MEMORY | sed s/\'//g )
          ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
          ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-
mem=$socket_mem
          ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-
mask=$pmd_cpu_mask
```

```
              ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-
mask=$host_cpu_mask
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
            $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}
```

3. Set the **tuned** configuration to provide CPU affinity.

```
  install_tuned:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name
format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              # Install the tuned package
              yum install -y tuned-profiles-cpu-partitioning

              tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
              if [ -n "$TUNED_CORES" ]; then
                grep -q "^isolated_cores" $tuned_conf_path
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
                else
                  echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
                fi
                tuned-adm profile cpu-partitioning
              fi
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

4. Set the kernel arguments.

```
  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name
format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS
isolcpus=$TUNED_CORES"/g' -i /etc/default/grub ;
              grub2-mkconfig -o /etc/grub2.cfg
              reboot
            fi
          params:
            $KERNEL_ARGS: {get_param: ComputeKernelArgs}
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}
```

5.2. CONFIGURING OPENVSWITCH FOR SECURITY GROUPS (TECHNOLOGY PREVIEW)

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

Controlplane interfaces can be configured by setting the **NeutronOVSFirewallDriver** parameter **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
```

NOTE

Openvswitch is a technology preview and should only be used in testing environments. The only supported value for the **NeutronOVSFirewallDriver** parameter is **noop**.

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}
```

5.3. DEFINING THE SR-IOV AND OVS-DPDK PARAMETERS

Modify the network-environment.yaml file to configure SR-IOV and OVS-DPDK role-specific parameters:

1. Add the resource mapping for the OVS-DPDK and SR-IOV services to the **network-environment.yaml** file along with the network configuration for these nodes:

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

2. Define the flavors:

```
OvercloudControlFlavor: controller
OvercloudComputeFlavor: compute
```

3. Define the tunnel type:

```
# The tunnel type for the tenant network (vxlan or gre). Set to '' to disable tunneling.
NeutronTunnelTypes: 'vxlan'
# The tenant network type for Neutron (vlan or vxlan).
NeutronNetworkType: 'vlan'
```

4. Configure the parameters for SR-IOV. You can obtain the PCI vendor and device values as seen in the **NeutronSupportedPCIVendorDevs** parameter by running **lspci -nv**.

NOTE

The OpenvSwitch firewall driver, as seen in the following example, is a Technology Preview and should be used for control plane interfaces only. The only supported value for the **NeutronOVSFirewallDriver** parameter is **noop**. See Configuring openvswitch for security groups for details.

```
NeutronSupportedPCIVendorDevs: ['8086:154d', '8086:10ed']
NovaPCIPassthrough:
  - devname: "p5p2"
    physical_network: "tenant"

NeutronPhysicalDevMappings: "tenant:p5p2"
NeutronSriovNumVFs: "p5p2:5"
# Global MTU.
NeutronGlobalPhysnetMtu: 9000
# Configure the classname of the firewall driver to use for implementing security groups.
NeutronOVSFirewallDriver: openvswitch
```

5. Configure the parameters for OVS-DPDK:

```
########################
# OVS DPDK configuration
## NeutronDpdkCoreList and NeutronDpdkMemoryChannels are REQUIRED settings.
## Attempting to deploy DPDK without appropriate values will cause deployment to fail or lead to unstable deployments.
# List of cores to be used for DPDK Poll Mode Driver
NeutronDpdkCoreList: "'2,22,3,23'"
# Number of memory channels to be used for DPDK
NeutronDpdkMemoryChannels: "4"
# NeutronDpdkSocketMemory
```

```
  NeutronDpdkSocketMemory: "'3072,1024'"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ########################
  # Additional settings
  ########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 4096
  # A list or range of physical CPU cores to reserve for virtual machine processes.
  # Example: NovaVcpuPinSet: ['4-12','^8'] will reserve cores from 4-12 excluding 8
  NovaVcpuPinSet: "4-19,24-39"
  # An array of filters used by Nova to filter a node.These filters will be applied in the order they are listed,
  # so place your most restrictive filters first to make the filtering process more efficient.
  NovaSchedulerDefaultFilters:
    - "RetryFilter"
    - "AvailabilityZoneFilter"
    - "RamFilter"
    - "ComputeFilter"
    - "ComputeCapabilitiesFilter"
    - "ImagePropertiesFilter"
    - "ServerGroupAntiAffinityFilter"
    - "ServerGroupAffinityFilter"
    - "PciPassthroughFilter"
    - "NUMATopologyFilter"
    - "AggregateInstanceExtraSpecsFilter"
  # Kernel arguments for Compute node
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt intel_iommu=on"
  # A list or range of physical CPU cores to be tuned.
  # The given args will be appended to the tuned cpu-partitioning profile.
  HostIsolatedCoreList: "2-19,22-39"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
  HostCpusList: "'2-19,22-39'"
```

NOTE

You must assign at least one CPU (with sibling thread) on each NUMA node with or without DPDK NICs present for DPDK PMD to avoid failures in creating guest instances.

6. Configure the remainder of the **network-environment.yaml** file to override the default parameters from the **neutron-ovs-dpdk-agent.yaml** and **neutron-sriov-agent.yaml** files as needed for your OpenStack deployment.

See the Network Functions Virtualization Planning Guide for details on how to 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.

5.4. CONFIGURING THE COMPUTE NODE FOR SR-IOV AND DPDK INTERFACES

This example uses the sample the compute.yaml file to support SR-IOV and DPDK interfaces.

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
ovs_extra:
  -
    str_replace:
      template: set port br-link0 tag=_VLAN_TAG_
      params:
        _VLAN_TAG_: {get_param: TenantNetworkVlanID}
addresses:
  -
    ip_netmask: {get_param: TenantIpSubnet}
use_dhcp: false
members:
  -
    type: ovs_dpdk_port
    name: dpdk0
    mtu: 9000
    ovs_extra:
```

```
- set interface $DEVICE mtu_request=$MTU
- set interface $DEVICE optoins:n_rxq=2
members:
  -
    type: interface
    name: nic7
    primary: true
```

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.

4. Create the SR-IOV interface to the Controller:

```
- type: interface
  name: p7p2
  mtu: 9000
  use_dhcp: false
  defroute: false
  nm_controlled: true
  hotplug: true
```

5.5. DEPLOYING THE OVERCLOUD

The following example defines the overcloud_deploy.sh Bash script that deploys both OVS-DPDK and SR-IOV:

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml \
-e /home/stack/ospd-10-vxlan-vlan-dpdk-sriov-ctlplane-bonding/network-environment.yaml
```

5.6. CREATING A FLAVOR AND DEPLOYING AN INSTANCE WITH SR-IOV AND DPDK INTERFACES

With a successful deployment, you can now begin populating the overcloud. Start by sourcing the newly created **overcloudrc** file in the **/home/stack** directory. Then, create a flavor and deploy an instance.

1. Create a flavor:

```
# source overcloudrc
# openstack flavor create --vcpus 6 --ram 4096 --disk 40 compute
```

Where:

- **compute** is the flavor name.
- **4096** is the memory size in MB.
- **40** is the disk size in GB (default 0G).
- **6** is the number of vCPUs.

2. Set the flavor for large pages:

```
# openstack flavor set compute --property hw:mem_page_size=1GB
```

3. Create the external network:

```
# openstack network create --share --external \
--provider-physical-network <net-mgmt-physnet> \
--provider-network-type <flat|vlan> external
```

4. Create the networks for SR-IOV and DPDK:

```
# openstack network create net-dpdk
# openstack network create net-sriov
# openstack subnet create --subnet-range <cidr/prefix> --network net-dpdk  net-dpdk-subnet
# openstack subnet create --subnet-range <cidr/prefix> --network net-sriov  net-sriov-subnet
```

5. Create the SR-IOV port.

   a. Use **vnic-type direct** to create an SR-IOV VF port:

   ```
   # openstack port create --network net-sriov --vnic-type direct sriov_port
   ```

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

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

6. Create a router and attach to the DPDK VXLAN network:

```
# openstack router create router1
# openstack router add subnet router1 net-dpdk-subnet
```

7. Create a floating IP address and associate it with the guest instance port:

```
# openstack floating ip create --floating-ip-address FLOATING-IP external
```

8. Deploy an instance:

```
# openstack server create --flavor compute --image rhel_7.3 --nic port-id=sriov_port --nic
net-id=NET_DPDK_ID vm1
```

Where:

- **compute** is the flavor name or ID.
- **rhel_7.3** is the image (name or ID) used to create an instance.
- **sriov_port** is the name of the port created in the previous step.
- *NET_DPDK_ID* is the DPDK network ID.
- **vm1** is the name of the instance.

You have now deployed an instance that uses an SR-IOV interface and a DPDK interface on the same Compute node.

NOTE

For instances with more interfaces, you can use **cloud-init**. See Table 3.1 in Create an Instance for details.

CHAPTER 6. 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 10 Documentation Suite

Table 6.1. List of Available Documentation

Component	Reference
Red Hat Enterprise Linux	Red Hat OpenStack Platform is supported on Red Hat Enterprise Linux 7.3. 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. You can also manually install the Red Hat OpenStack Platform components, see Manual Installation Procedures.
NFV Documentation	For a high level overview of the NFV concepts, see the Network Functions Virtualization Product Guide. For more details on planning your Red Hat OpenStack Platform deployment with NFV, see Network Function Virtualization Planning Guide.

APPENDIX A. SAMPLE SR-IOV YAML FILES

This section provides sample configuration files for single root I/O virtualization (SR-IOV) as a reference for Network Functions Virtualization infrastructure (NFVi).

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 VLAN SR-IOV YAML FILES

A.1.1. network-environment.yaml

```
resource_registry:
  # Specify the relative/absolute path to the config files you want to use for override the default.
  OS::TripleO::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
  # First boot and Kernel Args
  OS::TripleO::NodeUserData: first-boot.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 tenant network (vxlan or gre). Set to '' to disable tunneling.
  NeutronTunnelTypes: ''
  # The tenant network type for Neutron (vlan or vxlan).
  NeutronNetworkType: 'vlan'
  # The OVS logical->physical bridge mappings to use.
  NeutronBridgeMappings: 'tenant:br-link0'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'tenant:22:22,tenant:25:25'
  # Nova flavor to use.
  OvercloudControlFlavor: controller
  OvercloudComputeFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 1

  # Sets overcloud nodes custom names
  # http://docs.openstack.org/developer/tripleo-
docs/advanced_deployment/node_placement.html#custom-hostnames
  ControllerHostnameFormat: 'controller-%index%'
  ComputeHostnameFormat: 'compute-%index%'
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'

  #######################
  # SRIOV configuration #
  #######################
  # The mechanism drivers for the Neutron tenant network.
  NeutronMechanismDrivers: "openvswitch,sriovnicswitch"
  # List of PCI Passthrough whitelist parameters.
  # Use ONE of the following examples.
  # Example 1:
  # NovaPCIPassthrough:
  #   - vendor_id: "8086"
  #     product_id: "154c"
  #     address: "0000:05:00.0" - (optional)
  #     physical_network: "datacentre"
  #
  # Example 2:
  # NovaPCIPassthrough:
  #   - devname: "p6p1"
  #     physical_network: "tenant"
  NovaPCIPassthrough:
    - devname: "p7p1"
      physical_network: "tenant"
    - devname: "p7p2"
      physical_network: "tenant"
  # List of supported pci vendor devices in the format VendorID:ProductID.
  # Not merged into RHOSP10 refer BZ 1448919
  NeutronSupportedPCIVendorDevs: ['8086:154c', '8086:154d', '8086:10ed']
  # List of <physical_network>:<physical device>
  # All physical networks listed in network_vlan_ranges on the server
  # should have mappings to appropriate interfaces on each agent.
  NeutronPhysicalDevMappings: "tenant:p7p1,tenant:p7p2"
  # Provide the list of VFs to be reserved for each SR-IOV interface.
  # Format "<interface_name1>:<numvfs1>","<interface_name2>:<numvfs2>"
  # Example "eth1:4096","eth2:128"
  NeutronSriovNumVFs: "p7p1:5,p7p2:5"

  ########################
  # Additional settings
  ########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 4096
  # A list or range of physical CPU cores to reserve for virtual machine processes.
  # Example: NovaVcpuPinSet: ['4-12','^8'] will reserve cores from 4-12 excluding 8
  NovaVcpuPinSet: "1-19,21-39"
    # List of scheduler available filters
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter"]
  # An array of filters used by Nova to filter a node.These filters will be applied in the order they are
listed,
  # so place your most restrictive filters first to make the filtering process more efficient.
  NovaSchedulerDefaultFilters:
['AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','ImagePropertiesFilter','
ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter']
  # Kernel arguments for Compute node
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=12 intel_iommu=on
iommu=pt"
  # A list or range of physical CPU cores to be tuned.
  # The given args will be appended to the tuned cpu-partitioning profile.
  HostIsolatedCoreList: "1-19,21-39"

  # Set backend for overcloud
  GlanceBackend: 'file'
  # Global MTU
  NeutronGlobalPhysnetMtu: 9000

  SshServerOptions:
    UseDns: 'no'
```

A.1.2. first-boot.yaml

```
heat_template_version: 2014-10-16

description: >
  This is an example showing how you can do firstboot configuration
  of the nodes via cloud-init.  To enable this, replace the default
  mapping of OS::TripleO::NodeUserData in ../overcloud_resource_registry*

parameters:
  ComputeKernelArgs:
    description: >
      Space seprated list of Kernel args to be update to grub.
      The given args will be appended to existing args of GRUB_CMDLINE_LINUX in file
/etc/default/grub
      Example: "intel_iommu=on default_hugepagesz=1GB hugepagesz=1G hugepages=1"
    type: string
    default: ""
  ComputeHostnameFormat:
    type: string
    default: ""
  HostIsolatedCoreList:
    description: >
      A list or range of physical CPU cores to be tuned as isolated_cores.
      The given args will be appended to the tuned cpu-partitioning profile.
      Ex. HostIsolatedCoreList: '4-12' will tune cores from 4-12
    type: string
    default: ""

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

  install_tuned:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              # Install the tuned package
              yum install -y tuned-profiles-cpu-partitioning

              tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
              if [ -n "$TUNED_CORES" ]; then
                grep -q "^isolated_cores" $tuned_conf_path
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
                else
                  echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
                fi
                tuned-adm profile cpu-partitioning
              fi
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

  # Verify the logs on /var/log/cloud-init.log on the overcloud node
  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/g' -i /etc/default/grub ;
              grub2-mkconfig -o /etc/grub2.cfg

              sleep 5
              reboot
            fi
          params:
            $KERNEL_ARGS: {get_param: ComputeKernelArgs}
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

outputs:
  # This means get_resource from the parent template will get the userdata, see:
  # http://docs.openstack.org/developer/heat/template_guide/composition.html#making-your-template-resource-more-transparent
  # Note this is new-for-kilo, an alternative is returning a value then using
  # get_attr in the parent template instead.
  OS::stack_id:
    value: {get_resource: userdata}
```

A.1.3. controller.yaml

```
heat_template_version: 2015-04-30

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
  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
  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
  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
  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::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
```

```
network_config:
  -
    type: interface
    name: nic1
    use_dhcp: false
    defroute: false
  -
    type: interface
    name: nic2
    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: 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: TenantNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: TenantIpSubnet}
  -
    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: ovs_bond
         name: bond0
         use_dhcp: true
         members:
          -
           type: interface
           name: nic7
           mtu: 9000
          -
           type: interface
           name: nic8
           mtu: 9000

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

A.1.4. compute.yaml

```
heat_template_version: 2015-04-30

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
StorageNetworkVlanID:
  default: 30
  description: Vlan ID for the storage network traffic.
  type: number
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 externalheat stack-list network
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              defroute: false
            -
              type: interface
              name: nic2
              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
                  # force the MAC address of the bridge to this interface
                  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: TenantNetworkVlanID}
              device: bond_api
              addresses:
                -
                  ip_netmask: {get_param: TenantIpSubnet}
            -
              type: vlan
              vlan_id: {get_param: StorageNetworkVlanID}
              device: bond_api
              addresses:
                -
                  ip_netmask: {get_param: StorageIpSubnet}
            -
              type: interface
              name: p7p1
              mtu: 9000
              use_dhcp: false
              defroute: false
              nm_controlled: true
              hotplug: true

            -
              type: interface
              name: p7p2
              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 \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml \
-e /home/stack/ospd-10-vlan-sriov-two-ports-ctlplane-bonding/network-environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

APPENDIX B. SAMPLE OVS-DPDK YAML FILES

This section provides sample configuration files for Open vSwitch with Data Plane Development Kit (OVS-DPDK) as a reference for Network Functions Virtualization infrastructure (NFVi).

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.

B.1. SAMPLE VLAN OVS-DPDK DATA PLANE BONDING YAML FILES

B.1.1. first-boot.yaml

```
heat_template_version: 2014-10-16

description: >
  This is an example showing how you can do firstboot configuration
  of the nodes via cloud-init.  To enable this, replace the default
  mapping of OS::TripleO::NodeUserData in ../overcloud_resource_registry*

parameters:
  ComputeKernelArgs:
    description: >
      Space seprated list of Kernel args to be update to grub.
      The given args will be appended to existing args of GRUB_CMDLINE_LINUX in file
/etc/default/grub
      Example: "intel_iommu=on default_hugepagesz=1GB hugepagesz=1G hugepages=1"
    type: string
    default: ""
  ComputeHostnameFormat:
    type: string
    default: ""
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. Its mandatory parameter.
    type: string
  NeutronDpdkSocketMemory:
    description: Memory allocated for each socket
    default: ""
    type: string
    constraints:
      - allowed_pattern: "'[0-9,]+'"
  NeutronVhostuserSocketDir:
    description: The vhost-user socket directory for OVS.
    default: ""
    type: string
  HostIsolatedCoreList:
    description: >
      A list or range of physical CPU cores to be tuned as isolated_cores.
      The given args will be appended to the tuned cpu-partitioning profile.
      Ex. HostIsolatedCoreList: '4-12' will tune cores from 4-12
    type: string
    default: ""
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
    type: string
    constraints:
      - allowed_pattern: "'[0-9,]+'"

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  # Verify the logs on /var/log/cloud-init.log on the overcloud node
  set_dpdk_params:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            get_mask()
            {
              local list=$1
              local mask=0
              declare -a bm
              max_idx=0
              for core in $(echo $list | sed 's/,/ /g')
              do
                  index=$(($core/32))
                  bm[$index]=0
                  if [ $max_idx -lt $index ]; then
                     max_idx=$(($index))
                  fi
              done
              for ((i=$max_idx;i>=0;i--));
              do
                  bm[$i]=0
              done
              for core in $(echo $list | sed 's/,/ /g')
              do
                  index=$(($core/32))
                  temp=$((1<<$(($core % 32))))
                  bm[$index]=$((${bm[$index]} | $temp))
              done

              printf -v mask "%x" "${bm[$max_idx]}"
              for ((i=$max_idx-1;i>=0;i--));
              do
                  printf -v hex "%08x" "${bm[$i]}"
                  mask+=$hex
              done
      printf "%s" "$mask"
     }

     FORMAT=$COMPUTE_HOSTNAME_FORMAT
     if [[ -z $FORMAT ]] ; then
      FORMAT="compute" ;
     else
      # Assumption: only %index% and %stackname% are the variables in Host name format
      FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
     fi
     if [[ $(hostname) == *$FORMAT* ]] ; then
      # 42477 is the kolla hugetlbfs gid value.
      getent group hugetlbfs >/dev/null || \
       groupadd hugetlbfs -g 42477 && groupmod -g 42477 hugetlbfs

      pmd_cpu_mask=$( get_mask $PMD_CORES )
      host_cpu_mask=$( get_mask $LCORE_LIST )
      socket_mem=$(echo $SOCKET_MEMORY | sed s/\'//g )
      ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
      ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem=$socket_mem
      ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-mask=$pmd_cpu_mask
      ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=$host_cpu_mask
     fi
    params:
     $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
     $LCORE_LIST: {get_param: HostCpusList}
     $PMD_CORES: {get_param: NeutronDpdkCoreList}
     $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}

install_tuned:
 type: OS::Heat::SoftwareConfig
 properties:
  config:
   str_replace:
    template: |
     #!/bin/bash
     FORMAT=$COMPUTE_HOSTNAME_FORMAT
     if [[ -z $FORMAT ]] ; then
      FORMAT="compute" ;
     else
      # Assumption: only %index% and %stackname% are the variables in Host name format
      FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
     fi
     if [[ $(hostname) == *$FORMAT* ]] ; then
      # Install the tuned package
      yum install -y tuned-profiles-cpu-partitioning

      tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
      if [ -n "$TUNED_CORES" ]; then
       grep -q "^isolated_cores" $tuned_conf_path
       if [ "$?" -eq 0 ]; then
        sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
       else
        echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
       fi
       tuned-adm profile cpu-partitioning
```

```
              fi
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS
isolcpus=$TUNED_CORES"/g' -i /etc/default/grub ;
              grub2-mkconfig -o /etc/grub2.cfg
              reboot
            fi
          params:
            $KERNEL_ARGS: {get_param: ComputeKernelArgs}
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

outputs:
  # This means get_resource from the parent template will get the userdata, see:
  # http://docs.openstack.org/developer/heat/template_guide/composition.html#making-your-template-
resource-more-transparent
  # Note this is new-for-kilo, an alternative is returning a value then using
  # get_attr in the parent template instead.
  OS::stack_id:
    value: {get_resource: userdata}
```

B.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::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.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 tenant network (vxlan or gre). Set to '' to disable tunneling.
  NeutronTunnelTypes: ''
  # The tenant network type for Neutron (vlan or vxlan).
  NeutronNetworkType: 'vlan'
  # The OVS logical->physical bridge mappings to use.
  NeutronBridgeMappings: 'tenant:br-link0'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'tenant:22:22,tenant:25:25'
  # Nova flavor to use.
  OvercloudControlFlavor: controller
  OvercloudComputeFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 1
  # NTP server configuration.
  NtpServer: clock.redhat.com

  # Sets overcloud nodes custom names
  # http://docs.openstack.org/developer/tripleo-
docs/advanced_deployment/node_placement.html#custom-hostnames
  ControllerHostnameFormat: 'controller-%index%'
  ComputeHostnameFormat: 'compute-%index%'
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'

  #########################
  # OVS DPDK configuration
  ## NeutronDpdkCoreList and NeutronDpdkMemoryChannels are REQUIRED settings.
  ## Attempting to deploy DPDK without appropriate values will cause deployment to fail or lead to
unstable deployments.
  # List of cores to be used for DPDK Poll Mode Driver
  NeutronDpdkCoreList: "'2,22,3,23'"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "'3072,1024'"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ########################
  # Additional settings
  ########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 4096
  # A list or range of physical CPU cores to reserve for virtual machine processes.
  NovaVcpuPinSet: "4-19,24-39"
  # An array of filters used by Nova to filter a node.These filters will be applied in the order they are
listed,
  # so place your most restrictive filters first to make the filtering process more efficient.
  NovaSchedulerDefaultFilters:
    - "RetryFilter"
    - "AvailabilityZoneFilter"
    - "RamFilter"
    - "ComputeFilter"
    - "ComputeCapabilitiesFilter"
    - "ImagePropertiesFilter"
    - "ServerGroupAntiAffinityFilter"
    - "ServerGroupAffinityFilter"
    - "PciPassthroughFilter"
    - "NUMATopologyFilter"
    - "AggregateInstanceExtraSpecsFilter"
  # Kernel arguments for Compute node
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on"
  # A list or range of physical CPU cores to be tuned.
  # The given args will be appended to the tuned cpu-partitioning profile.
  HostIsolatedCoreList: "2-19,22-39"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
  HostCpusList: "'0,20,1,21'"
  NovaLibvirtRxQueueSize: 1024
  NovaLibvirtTxQueueSize: 1024

  # MTU global configuration
  NeutronGlobalPhysnetMtu: 9000
  # Set the storage backend of the overcloud
  GlanceBackend: 'file'
  # Configure the classname of the firewall driver to use for implementing security groups.
  NeutronOVSFirewallDriver: openvswitch

  SshServerOptions:
    UseDns: 'no'
```

B.1.3. controller.yaml

```
heat_template_version: 2015-04-30
```

```
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
  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
  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
  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
  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::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              defroute: false
            -
              type: interface
              name: nic2
              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: 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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  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: ovs_bond
      name: bond0
      use_dhcp: true
      members:
        -
          type: interface
          name: nic7
```

```
                      mtu: 9000
                    -
                      type: interface
                      name: nic8
                      mtu: 9000


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

B.1.4. compute-ovs-dpdk.yaml

```
heat_template_version: 2015-04-30

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
  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
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  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::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  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
  members:
    -
      type: ovs_dpdk_bond
      name: dpdkbond0
      mtu: 9000
      ovs_extra:
        - set interface dpdk0 mtu_request=$MTU
        - set interface dpdk1 mtu_request=$MTU
        - set interface dpdk0 options:n_rxq=2
        - set interface dpdk1 options:n_rxq=2
      members:
        -
```

```
                      type: ovs_dpdk_port
                      name: dpdk0
                      members:
                        -
                          type: interface
                          name: nic7
                    -
                      type: ovs_dpdk_port
                      name: dpdk1
                      members:
                        -
                          type: interface
                          name: nic8


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

B.1.5. overcloud_deploy.sh

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ovs-dpdk-permissions.yaml \
-e /home/stack/ospd-10-vlan-dpdk-two-ports-ctlplane-dataplane-bonding/network-environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

B.2. SAMPLE VXLAN OVS-DPDK DATA PLANE BONDING YAML FILES

B.2.1. first-boot.yaml

```
heat_template_version: 2014-10-16

description: >
  This is an example showing how you can do firstboot configuration
  of the nodes via cloud-init.  To enable this, replace the default
  mapping of OS::TripleO::NodeUserData in ../overcloud_resource_registry*

parameters:
  ComputeKernelArgs:
    description: >
      Space seprated list of Kernel args to be update to grub.
      The given args will be appended to existing args of GRUB_CMDLINE_LINUX in file
/etc/default/grub
      Example: "intel_iommu=on default_hugepagesz=1GB hugepagesz=1G hugepages=1"
    type: string
    default: ""
  ComputeHostnameFormat:
    type: string
    default: ""
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. Its mandatory parameter.
    type: string
  NeutronDpdkSocketMemory:
    description: Memory allocated for each socket
    default: ""
    type: string
    constraints:
      - allowed_pattern: "'[0-9,]+'"
  NeutronVhostuserSocketDir:
    description: The vhost-user socket directory for OVS.
    default: ""
    type: string
  HostIsolatedCoreList:
    description: >
      A list or range of physical CPU cores to be tuned as isolated_cores.
      The given args will be appended to the tuned cpu-partitioning profile.
      Ex. HostIsolatedCoreList: '4-12' will tune cores from 4-12
    type: string
    default: ""
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
    type: string
    constraints:
      - allowed_pattern: "'[0-9,]+'"

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  # Verify the logs on /var/log/cloud-init.log on the overcloud node
  set_dpdk_params:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            get_mask()
            {
              local list=$1
              local mask=0
              declare -a bm
              max_idx=0
              for core in $(echo $list | sed 's/,/ /g')
              do
        index=$(($core/32))
        bm[$index]=0
        if [ $max_idx -lt $index ]; then
          max_idx=$(($index))
        fi
      done
      for ((i=$max_idx;i>=0;i--));
      do
        bm[$i]=0
      done
      for core in $(echo $list | sed 's/,/ /g')
      do
        index=$(($core/32))
        temp=$((1<<$(($core % 32))))
        bm[$index]=$((${bm[$index]} | $temp))
      done

      printf -v mask "%x" "${bm[$max_idx]}"
      for ((i=$max_idx-1;i>=0;i--));
      do
        printf -v hex "%08x" "${bm[$i]}"
        mask+=$hex
      done
      printf "%s" "$mask"
     }

     FORMAT=$COMPUTE_HOSTNAME_FORMAT
     if [[ -z $FORMAT ]] ; then
      FORMAT="compute" ;
     else
      # Assumption: only %index% and %stackname% are the variables in Host name format
      FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
     fi
     if [[ $(hostname) == *$FORMAT* ]] ; then
      # 42477 is the kolla hugetlbfs gid value.
      getent group hugetlbfs >/dev/null || \
       groupadd hugetlbfs -g 42477 && groupmod -g 42477 hugetlbfs

      pmd_cpu_mask=$( get_mask $PMD_CORES )
      host_cpu_mask=$( get_mask $LCORE_LIST )
      socket_mem=$(echo $SOCKET_MEMORY | sed s/\'//g )
      ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
      ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem=$socket_mem
      ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-mask=$pmd_cpu_mask
      ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=$host_cpu_mask
     fi
    params:
     $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
     $LCORE_LIST: {get_param: HostCpusList}
     $PMD_CORES: {get_param: NeutronDpdkCoreList}
     $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}

install_tuned:
 type: OS::Heat::SoftwareConfig
 properties:
  config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              # Install the tuned package
              yum install -y tuned-profiles-cpu-partitioning

              tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
              if [ -n "$TUNED_CORES" ]; then
                grep -q "^isolated_cores" $tuned_conf_path
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
                else
                  echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
                fi
                tuned-adm profile cpu-partitioning
              fi
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS
isolcpus=$TUNED_CORES"/g' -i /etc/default/grub ;
              grub2-mkconfig -o /etc/grub2.cfg
              reboot
            fi
          params:
            $KERNEL_ARGS: {get_param: ComputeKernelArgs}
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

outputs:
  # This means get_resource from the parent template will get the userdata, see:
```

```
  # http://docs.openstack.org/developer/heat/template_guide/composition.html#making-your-template-resource-more-transparent
  # Note this is new-for-kilo, an alternative is returning a value then using
  # get_attr in the parent template instead.
  OS::stack_id:
    value: {get_resource: userdata}
```

B.2.2. network-environment.yaml

```
resource_registry:
  # Specify the relative/absolute path to the config files you want to use for override the default.
  OS::TripleO::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.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 tenant network (vxlan or gre). Set to '' to disable tunneling.
  NeutronTunnelTypes: 'vxlan'
  # The tenant network type for Neutron (vlan or vxlan).
  NeutronNetworkType: 'vxlan'
  # The OVS logical->physical bridge mappings to use.
  NeutronBridgeMappings: 'tenant:br-link0'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'tenant:22:22'
  # Nova flavor to use.
  OvercloudControlFlavor: controller
  OvercloudComputeFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 1
  # NTP server configuration.
  NtpServer: clock.redhat.com

  # Sets overcloud nodes custom names
  # http://docs.openstack.org/developer/tripleo-docs/advanced_deployment/node_placement.html#custom-hostnames
  ControllerHostnameFormat: 'controller-%index%'
  ComputeHostnameFormat: 'compute-%index%'
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'

  ########################
  # OVS DPDK configuration
  ## NeutronDpdkCoreList and NeutronDpdkMemoryChannels are REQUIRED settings.
  ## Attempting to deploy DPDK without appropriate values will cause deployment to fail or lead to unstable deployments.
  # List of cores to be used for DPDK Poll Mode Driver
  NeutronDpdkCoreList: "'2,22,3,23'"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "'3072,1024'"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ########################
  # Additional settings
  ########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 4096
  # A list or range of physical CPU cores to reserve for virtual machine processes.
  NovaVcpuPinSet: "4-19,24-39"
  # An array of filters used by Nova to filter a node.These filters will be applied in the order they are listed,
  # so place your most restrictive filters first to make the filtering process more efficient.
  NovaSchedulerDefaultFilters:
    - "RetryFilter"
    - "AvailabilityZoneFilter"
    - "RamFilter"
    - "ComputeFilter"
    - "ComputeCapabilitiesFilter"
    - "ImagePropertiesFilter"
    - "ServerGroupAntiAffinityFilter"
    - "ServerGroupAffinityFilter"
    - "PciPassthroughFilter"
    - "NUMATopologyFilter"
    - "AggregateInstanceExtraSpecsFilter"
  # Kernel arguments for Compute node
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt intel_iommu=on"
  # A list or range of physical CPU cores to be tuned.
  # The given args will be appended to the tuned cpu-partitioning profile.
  HostIsolatedCoreList: "2-19,22-39"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
  HostCpusList: "'0,20,1,21'"
  NovaLibvirtRxQueueSize: 1024
  NovaLibvirtTxQueueSize: 1024

  # MTU global configuration
  NeutronGlobalPhysnetMtu: 9000
  # Set the storage backend of the overcloud
  GlanceBackend: 'file'
  # Configure the classname of the firewall driver to use for implementing security groups.
  NeutronOVSFirewallDriver: openvswitch

  SshServerOptions:
    UseDns: 'no'
```

B.2.3. controller.yaml

```
heat_template_version: 2015-04-30

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
  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
  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
  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
  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::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              defroute: false
            -
              type: interface
  name: nic2
  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: 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: 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: ovs_bond
                  name: bond0
                  use_dhcp: true
                  members:
                    -
                      type: interface
                      name: nic7
                      mtu: 9000
                    -
                      type: interface
                      name: nic8
                      mtu: 9000
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  device: bond0
                  mtu: 9000
                  addresses:
                    -
                      ip_netmask: {get_param: TenantIpSubnet}


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

B.2.4. compute-ovs-dpdk.yaml

```
heat_template_version: 2015-04-30

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
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage 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
  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::StructuredConfig
properties:
  group: os-apply-config
  config:
    os_net_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
                  ovs_extra:
                    - set interface dpdk0 mtu_request=$MTU
                    - set interface dpdk1 mtu_request=$MTU
                    - set interface dpdk0 options:n_rxq=2
                    - set interface dpdk1 options:n_rxq=2
                  members:
                    -
                      type: ovs_dpdk_port
                      name: dpdk0
                      members:
                        -
                          type: interface
                          name: nic7
                    -
                      type: ovs_dpdk_port
                      name: dpdk1
                      members:
                        -
                          type: interface
                          name: nic8

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

B.2.5. overcloud_deploy.sh

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ovs-dpdk-permissions.yaml \
-e /home/stack/ospd-10-vxlan-dpdk-single-port-ctlplane-bonding/network-environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

APPENDIX C. DIFFERENT INTERFACES ON SAME COMPUTE NODE YAML FILES

This section provides sample YAML files as a reference for adding SR-IOV and 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.

C.1. SAMPLE SR-IOV AND DPDK ON THE SAME COMPUTE NODE YAML FILES

This section provides sample DPDK and SR-IOV YAML files as a reference.

C.1.1. first-boot.yaml

```
heat_template_version: 2014-10-16

description: >
  This is an example showing how you can do firstboot configuration
  of the nodes via cloud-init.  To enable this, replace the default
  mapping of OS::TripleO::NodeUserData in ../overcloud_resource_registry*

parameters:
  ComputeKernelArgs:
    description: >
      Space seprated list of Kernel args to be update to grub.
      The given args will be appended to existing args of GRUB_CMDLINE_LINUX in file
/etc/default/grub
      Example: "intel_iommu=on default_hugepagesz=1GB hugepagesz=1G hugepages=1"
    type: string
    default: ""
  ComputeHostnameFormat:
    type: string
    default: ""
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. Its mandatory parameter.
    type: string
  NeutronDpdkSocketMemory:
    description: Memory allocated for each socket
    default: ""
    type: string
    constraints:
      - allowed_pattern: "'[0-9,]+'"
  NeutronVhostuserSocketDir:
    description: The vhost-user socket directory for OVS.
    default: ""
    type: string
  HostIsolatedCoreList:
    description: >
```

```
      A list or range of physical CPU cores to be tuned as isolated_cores.
      The given args will be appended to the tuned cpu-partitioning profile.
      Ex. HostIsolatedCoreList: '4-12' will tune cores from 4-12
    type: string
    default: ""
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
    type: string
    constraints:
      - allowed_pattern: "'[0-9,]+'"

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  # Verify the logs on /var/log/cloud-init.log on the overcloud node
  set_dpdk_params:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            get_mask()
            {
              local list=$1
              local mask=0
              declare -a bm
              max_idx=0
              for core in $(echo $list | sed 's/,/ /g')
              do
                  index=$(($core/32))
                  bm[$index]=0
                  if [ $max_idx -lt $index ]; then
                     max_idx=$(($index))
                  fi
              done
              for ((i=$max_idx;i>=0;i--));
              do
                  bm[$i]=0
              done
              for core in $(echo $list | sed 's/,/ /g')
              do
                  index=$(($core/32))
                  temp=$((1<<$(($core % 32))))
                  bm[$index]=$((${bm[$index]} | $temp))
              done

              printf -v mask "%x" "${bm[$max_idx]}"
        for ((i=$max_idx-1;i>=0;i--));
        do
          printf -v hex "%08x" "${bm[$i]}"
          mask+=$hex
        done
        printf "%s" "$mask"
       }

       FORMAT=$COMPUTE_HOSTNAME_FORMAT
       if [[ -z $FORMAT ]] ; then
        FORMAT="compute" ;
       else
        # Assumption: only %index% and %stackname% are the variables in Host name format
        FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
       fi
       if [[ $(hostname) == *$FORMAT* ]] ; then
        # 42477 is the kolla hugetlbfs gid value.
        getent group hugetlbfs >/dev/null || \
         groupadd hugetlbfs -g 42477 && groupmod -g 42477 hugetlbfs

        pmd_cpu_mask=$( get_mask $PMD_CORES )
        host_cpu_mask=$( get_mask $LCORE_LIST )
        socket_mem=$(echo $SOCKET_MEMORY | sed s/\'//g )
        ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
        ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem=$socket_mem
        ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-mask=$pmd_cpu_mask
        ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=$host_cpu_mask
       fi
      params:
       $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
       $LCORE_LIST: {get_param: HostCpusList}
       $PMD_CORES: {get_param: NeutronDpdkCoreList}
       $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}

  install_tuned:
   type: OS::Heat::SoftwareConfig
   properties:
    config:
     str_replace:
      template: |
       #!/bin/bash
       FORMAT=$COMPUTE_HOSTNAME_FORMAT
       if [[ -z $FORMAT ]] ; then
        FORMAT="compute" ;
       else
        # Assumption: only %index% and %stackname% are the variables in Host name format
        FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
       fi
       if [[ $(hostname) == *$FORMAT* ]] ; then
        # Install the tuned package
        yum install -y tuned-profiles-cpu-partitioning

        tuned_conf_path="/etc/tuned/cpu-partitioning-variables.conf"
        if [ -n "$TUNED_CORES" ]; then
         grep -q "^isolated_cores" $tuned_conf_path
         if [ "$?" -eq 0 ]; then
```

```
                  sed -i 's/^isolated_cores=.*/isolated_cores=$TUNED_CORES/' $tuned_conf_path
                else
                  echo "isolated_cores=$TUNED_CORES" >> $tuned_conf_path
                fi
                tuned-adm profile cpu-partitioning
              fi
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else
              # Assumption: only %index% and %stackname% are the variables in Host name format
              FORMAT=$(echo $FORMAT | sed  's/\%index\%//g' | sed 's/\%stackname\%//g') ;
            fi
            if [[ $(hostname) == *$FORMAT* ]] ; then
              sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS
isolcpus=$TUNED_CORES"/g' -i /etc/default/grub ;
              grub2-mkconfig -o /etc/grub2.cfg
              reboot
            fi
          params:
            $KERNEL_ARGS: {get_param: ComputeKernelArgs}
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $TUNED_CORES: {get_param: HostIsolatedCoreList}

outputs:
  # This means get_resource from the parent template will get the userdata, see:
  # http://docs.openstack.org/developer/heat/template_guide/composition.html#making-your-template-
resource-more-transparent
  # Note this is new-for-kilo, an alternative is returning a value then using
  # get_attr in the parent template instead.
  OS::stack_id:
    value: {get_resource: userdata}
```

C.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::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.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 tenant network (vxlan or gre). Set to '' to disable tunneling.
  NeutronTunnelTypes: 'vxlan'
  # The tenant network type for Neutron (vlan or vxlan).
  NeutronNetworkType: 'vlan'
  # The OVS logical->physical bridge mappings to use.
  NeutronBridgeMappings: 'dpdk_mgmt:br-link0,tenant:br-link1'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'tenant:22:22,tenant:25:25'
  # Nova flavor to use.
  OvercloudControlFlavor: controller
  OvercloudComputeFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 1
  # NTP server configuration.
  NtpServer: clock.redhat.com

  # Sets overcloud nodes custom names
  # http://docs.openstack.org/developer/tripleo-
docs/advanced_deployment/node_placement.html#custom-hostnames
  ControllerHostnameFormat: 'controller-%index%'
  ComputeHostnameFormat: 'compute-%index%'
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'

  ########################
  # OVS DPDK configuration
  ## NeutronDpdkCoreList and NeutronDpdkMemoryChannels are REQUIRED settings.
  ## Attempting to deploy DPDK without appropriate values will cause deployment to fail or lead to
unstable deployments.
  # List of cores to be used for DPDK Poll Mode Driver
  NeutronDpdkCoreList: "'2,22,3,23'"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "'3072,1024'"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ########################
  # Additional settings
  ########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 4096
  # A list or range of physical CPU cores to reserve for virtual machine processes.
  NovaVcpuPinSet: "4-19,24-39"
  # An array of filters used by Nova to filter a node.These filters will be applied in the order they are
listed,
  # so place your most restrictive filters first to make the filtering process more efficient.
  NovaSchedulerDefaultFilters:
    - "RetryFilter"
    - "AvailabilityZoneFilter"
    - "RamFilter"
    - "ComputeFilter"
    - "ComputeCapabilitiesFilter"
    - "ImagePropertiesFilter"
    - "ServerGroupAntiAffinityFilter"
    - "ServerGroupAffinityFilter"
    - "PciPassthroughFilter"
    - "NUMATopologyFilter"
    - "AggregateInstanceExtraSpecsFilter"
  # Kernel arguments for Compute node
  ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on"
  # A list or range of physical CPU cores to be tuned.
  # The given args will be appended to the tuned cpu-partitioning profile.
  HostIsolatedCoreList: "2-19,22-39"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-mask)
  HostCpusList: "'0,20,1,21'"
  NovaLibvirtRxQueueSize: 1024
  NovaLibvirtTxQueueSize: 1024

  # List of supported pci vendor devices in the format VendorID:ProductID.
  # Not merged into RHOSP10 refer BZ 1448919
  NeutronSupportedPCIVendorDevs: ['8086:154c', '8086:154d', '8086:10ed']
  NovaPCIPassthrough:
    - devname: "p5p2"
      physical_network: "tenant"

  NeutronPhysicalDevMappings: "tenant:p5p2"
  NeutronSriovNumVFs: "p5p2:5"
  # Global MTU.
  NeutronGlobalPhysnetMtu: 9000
  # Configure the classname of the firewall driver to use for implementing security groups.
  NeutronOVSFirewallDriver: openvswitch

  SshServerOptions:
    UseDns: 'no'
```

C.1.3. controller.yaml

```
heat_template_version: 2015-04-30

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
  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
  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
  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
  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
  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
  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::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              defroute: false
            -
              type: interface
              name: nic2
              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: 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: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  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
              members:
                -
                   type: interface
                   name: nic7
                   mtu: 9000
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  mtu: 9000
                  addresses:
                    -
                      ip_netmask: {get_param: TenantIpSubnet}
            -
              type: ovs_bridge
              name: br-link1
              use_dhcp: false
              members:
                -
                   type: interface
                   name: nic8
                   mtu: 9000

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

C.1.4. compute.yaml

```
heat_template_version: 2015-04-30

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
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  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
  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::StructuredConfig
  properties:
    group: os-apply-config
    config:
      os_net_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
              ovs_extra:
                -
                  str_replace:
                    template: set port br-link0 tag=_VLAN_TAG_
                    params:
                      _VLAN_TAG_: {get_param: TenantNetworkVlanID}
              addresses:
                -
                  ip_netmask: {get_param: TenantIpSubnet}
              use_dhcp: false
              members:
                -
                  type: ovs_dpdk_port
                  name: dpdk0
                  mtu: 9000
                  ovs_extra:
                  - set interface $DEVICE mtu_request=$MTU
                  - set interface $DEVICE options:n_rxq=2
                  members:
                    -
                      type: interface
                      name: nic7
                      primary: true

            - type: interface
              name: p7p2
              mtu: 9000
              use_dhcp: false
              defroute: false
              nm_controlled: true
              hotplug: true

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

C.1.5. overcloud_deploy.sh

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ovs-dpdk-permissions.yaml \
-e /home/stack/ospd-10-vxlan-vlan-dpdk-sriov-ctlplane-bonding/network-environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

APPENDIX D. REVISION HISTORY

Revision 10.3-0 July 31 2018

Updated network creation steps to use OSC parameters.

Revision 10.2-0 July 24 2018

Removed section 'Configure OVS-DPDK Composable Role'.

Revision 10.1-0 June 27 2018

Updates for 10zasync release with OVS 2.9 support.

Revision 10.0-0 April 11 2018

Updates for 10z7 release.