Red Hat OpenStack Platform 11

Network Functions Virtualization Configuration Guide

Configuring the Network Functions Virtualization (NFV) OpenStack Deployment

Last Updated: 2019-01-31

Red Hat OpenStack Platform 11 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 11 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-acclerated Open vSwitch (OVS) using the Red Hat OpenStack Platform 11 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 (CSP) to move away from traditional hardware.

NOTE

OVS-DPDK and SR-IOV configuration depends on your hardware and topology. This guide provides examples for CPU assignments, memory allocation, and NIC configurations that may vary from your topology and use case. See the Network Functions Virtualization Product Guide and the Network Functions Virtualization Planning and Prerequisite Guide to understand the hardware and configuration options.

Red Hat OpenStack Platform 11 director allows you to isolate the overcloud networks. Using this feature, you can separate specific network types (for example, external, tenant, internal API and so on) into isolated networks. You can deploy a network on a single network interface or distributed over a multiple host network interface. Open vSwitch allows you to create bonds by assigning multiple interfaces to a single bridge. Network isolation in a Red Hat OpenStack Platform 11 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 three types of template configuration files:

- **`network-environment.yaml`** - this file contains the network details such as, subnets, IP address ranges that are used for the network configuration on the overcloud nodes. In addition, this file also contains the different settings that override the default parameter values for various scenarios.
- Host templates (for example, **`compute.yaml`** , **`controller.yaml`**) - Define the network interface configuration for the overcloud nodes. The values of the network details are provided by the **`network-environment.yaml`** file.
- Post install configuration files (**`first-boot.yaml`**, **`post-install.yaml`**) - Provide various post configuration steps, for example:
  - Grub argument configurations.
  - DPDK configuration.
  - 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. See Performance Tuning Guide for details on this package.

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.

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

NOTE

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

CHAPTER 2. UPGRADING RED HAT OPENSTACK PLATFORM WITH NFV

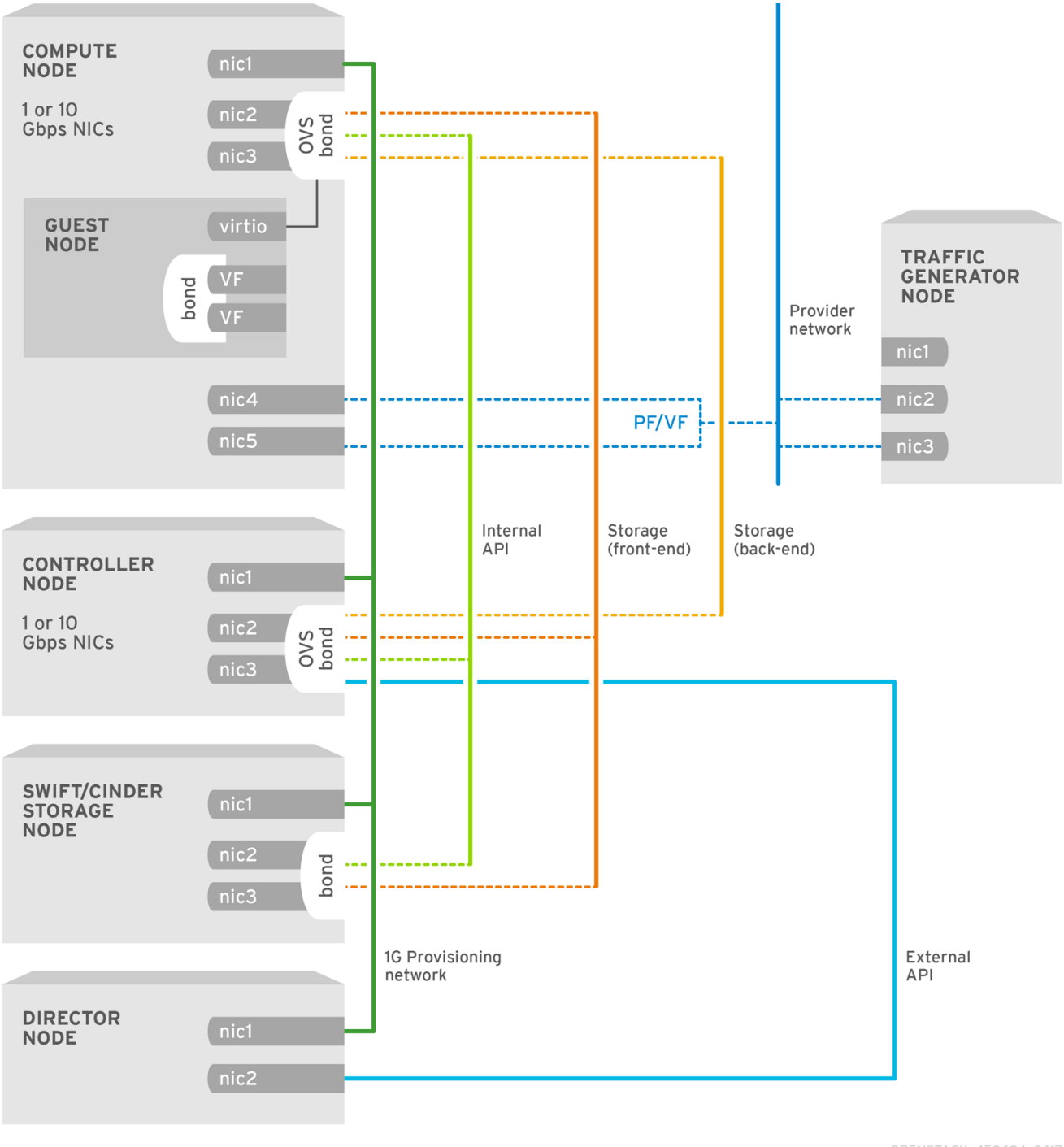
There are additional considerations and steps needed to upgrade Red Hat OpenStack Platform when you have OVS-DPDK configured. The steps are covered in NFV Configuration in the *Upgrading Red Hat OpenStack Platform Guide*.

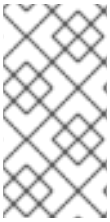
CHAPTER 3. CONFIGURE SR-IOV SUPPORT FOR VIRTUAL NETWORKING

This section describes how to configure Single Root Input/Output Virtualization (SR-IOV) for Red Hat OpenStack.

3.1. UNDERSTANDING SR-IOV CONFIGURABLE PARAMETERS

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.





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 and Prerequisite Guide to understand the hardware and configuration options.

3.2. CONFIGURE SINGLE-NIC SR-IOV COMPOSABLE ROLE WITH VLAN TUNNELLING

This section describes how to configure a composable role for SR-IOV with VLAN tunnelling for your OpenStack environment. The process to create and deploy a composable role includes:

- Define the new role in a local copy of the **`role_data.yaml`** file.
- Modify the **`network_environment.yaml`** file to include this new role.
- Deploy the overcloud with this updated set of roles.

In this example, **ComputeSriov** is a composable role for compute node to enable SR-IOV only on the nodes that have the SR-IOV NICs. The existing set of default roles provided by the Red Hat OpenStack Platform is stored in the **`/home/stack/roles_data.yaml`** file.

3.2.1. Modify roles_data.yaml to Create an SR-IOV Composable Role

Copy the **`roles_data.yaml`** file to your **`/home/stack/templates`** directory and add the new `**ComputeSriov** role.

```
- name: ComputeSriov
  CountDefault: 1
  HostnameFormatDefault: compute-sriov-%index%
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
```

```
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

3.2.2. Modify first-boot.yaml

1. Set the configuration to install **tuned** for CPU affinity.

```
  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
              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

              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}
```

3.2.3. Modify post-install.yaml

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

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      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
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i '/After=.*/s/network.target//g'
$tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                grep -q "Before=.*" $tuned_service
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                else
                  sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                fi
              fi
              systemctl daemon-reload
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

3.2.4. Modify network-environment.yaml

1. Add the custom resources for SR-IOV including the **ComputeSriov** role under **`resource_registry`**.

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

```
  OS::TripleO::Services::NeutronSriovAgent: /usr/share/openstack-
tripleo-heat-templates/puppet/services/neutron-sriov-agent.yaml

  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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 physical network to the logical bridge.

```
NeutronBridgeMappings: 'tenant:br-link'
```

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

```
NeutronNetworkVLANRanges: 'tenant:420:420,tenant:421:421'
```

   This example sets the VLAN ranges on the physical network (**tenant**).

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: "ens2f1"
      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:ens2f1"
```

   This example uses **tenant** as the **physical_network** name.

   d. Provide the number of Virtual Functions (VFs) to be reserved for each SR-IOV interface.

```
NeutronSriovNumVFs: "ens2f1:5"
```

   This example reserves 5 VFs for the SR-IOV interface.

NOTE

Currently, Red Hat OpenStack Platform with NFV supports 30 or fewer VFs.

6. Under **parameter_defaults**, list 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:
['AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabi
litiesFilter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter
','ServerGroupAffinityFilter','PciPassthroughFilter','NUMATopologyFi
lter']
```

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

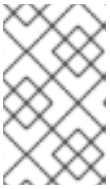
```
NovaReservedHostMemory: 2048
```

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

```
NovaVcpuPinSet:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

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=32 iommu=pt intel_iommu=on"
```



NOTE

You need to add **hw:mem_page_size=1GB** to the flavor you associate with the SR-IOV 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 appended to the **tuned cpu-partitioning** profile and isolated from the host.

```
HostIsolatedCoreList:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

3.2.5. Modify controller.yaml

1. Create the interface for an isolated network.

```
-
  type: interface
  name: ens1f1
  use_dhcp: false
```

```
    dns_servers: {get_param: DnsServers}
```

2. Assign VLANs to this interface.

```
-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: vlan
  vlan_id: {get_param: TenantNetworkVlanID}
  device: ens1f1
  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: ens1f1
  addresses:
  -
    ip_netmask: {get_param: ExternalIpSubnet}
  routes:
    -
      default: true
      next_hop: {get_param: ExternalInterfaceDefaultRoute}
```

3. Create the OVS bridge to the Compute node.

```
-
  type: ovs_bridge
  name: br-link
  use_dhcp: false
  members:
    -
      type: interface
      name: ens2f1
```

3.2.6. Modify compute-sriov.yaml

Create **compute-sriov.yaml** from the default **compute.yaml** file. This is the file that controls the parameters for the Compute nodes that use the **ComputeSriov** composable role.

1. Create the interface for an isolated network.

```
-
  type: interface
  name: ens1f1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
```

2. Assign VLANs to this interface.

```
-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: vlan
  vlan_id: {get_param: TenantNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: StorageIpSubnet}
```

3. Create an interface to the controller node.

```
-
  type: interface
  name: ens2f1
  use_dhcp: false
  defroute: false
```

3.2.7. Run the overcloud_deploy.sh Script

The following example defines the **openstack overcloud deploy** Bash script that uses composable roles:

```
#!/bin/bash

openstack overcloud deploy \
--templates \
```

```
-r /home/stack/ospd-11-vlan-sriov-single-port-composable-roles/roles-
data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
-e /home/stack/ospd-11-vlan-sriov-single-port-composable-roles/network-
environment.yaml
```

home/stack/ospd-11-vlan-sriov-single-port-composable-roles/roles-data.yaml is the location of the updated **roles_data.yaml** file, which defines the SR-IOV composable role.

NOTE

Reboot the Compute nodes to enforce the **tuned** profile after the overcloud is deployed.

3.3. CONFIGURE TWO-PORT SR-IOV WITH HCI

This section describes how to configure a composable role for SR-IOV with VLAN tunnelling for your OpenStack HCI environment. The process includes:

- Defining the new role in a local copy of the **role_data.yaml** file.
- Defining the OpenStack flavor to use this new role.
- Modifying the **network_environment.yaml** file to include this new role.
- Deploying the overcloud with this updated set of roles.

In this example, the **Compute** role is modified to enable SR-IOV and Ceph OSD only on the nodes that have the appropriate hardware and NICs. The existing set of default roles provided by Red Hat OpenStack Platform is stored in the **/home/stack/roles_data.yaml** file.

NOTE

See the Hyper-Converged Infrastructure Guide for complete details on configuring HCI. The SR-IOV configuration described here is complementary to the HCI configuration in that guide.

3.3.1. Modify custom-roles.yaml to Create an SR-IOV Composable Role

Copy the **roles_data.yaml** file to your **/home/stack/templates** directory and modify the **Compute** role.

```
- name: Compute
  CountDefault: 1
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
```

```
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

3.3.2. Modify first-boot.yaml

1. Add the NFV **tuned** configuration to the HCI **first-boot.yaml** file to enable CPU affinity.

```
  tuned_config:
    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
              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 initial boot parameters.

```
  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 $KERNEL_ARGS
isolcpus=$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}
```

3.3.3. Modify post-install.yaml

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

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      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
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i '/After=.*/s/network.target//g'
$tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                grep -q "Before=.*" $tuned_service
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                else
                  sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                fi
              fi

              systemctl daemon-reload
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

NOTE

If you have the appropriate hardware, configure Ceph NUMA pinning as well for optimal performance. See Configure Ceph NUMA Pinning.

3.3.4. Modify network-environment.yaml

1. Add the custom resources for SR-IOV under **resource_registry**. This is in addition to any resources configured for HCI. See the Hyper-Converged Infrastructure Guide for complete details on configuring HCI.

```
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
  OS::TripleO::NodeExtraConfigPost: post-install.yamll
```

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 physical network to the logical bridge.

```
NeutronBridgeMappings: 'datacentre:br-isolated,tenant:br-
sriov,tenant2:br-sriov2'
```

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

```
NeutronNetworkVLANRanges:
'datacentre:419:419,tenant:420:420,tenant2:421:421'
```

This example sets the VLAN ranges on the physical network (**tenant**).

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: "ens2f0"
    physical_network: "tenant"
  - devname: "ens2f1"
    physical_network: "tenant2"
```

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:ens2f0,tenant2:ens2f1"
```

This example uses **tenant** as the **physical_network** name.

d. Provide the number of Virtual Functions (VFs) to be reserved for each SR-IOV interface.

```
NeutronSriovNumVFs: "ens2f0:7,ens2f1:7"
```

This example reserves 7 VFs for each of the SR-IOV interfaces.

NOTE

Currently, the Red Hat OpenStack Platform with NFV supports 30 or fewer VFs.

6. Under **parameter_defaults**, list 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:
['AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabi
litiesFilter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter
','ServerGroupAffinityFilter','PciPassthroughFilter']
```

7. 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 SR-IOV instance. If you do not do this, the instance does not get a DHCP allocation.

8. Under **parameter_defaults**, set a list or range of physical CPU cores to be appended to the **tuned cpu-partitioning** profile and isolated from the host.

```
HostIsolatedCoreList:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

3.3.5. Modify controller.yaml

1. Create the bridge and interface for an isolated network.

```
-
  type: ovs_bridge
  name: br-isolated
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      # force the MAC address of the bridge to this interface
      primary: true
```

2. Assign VLANs to this bridge.

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

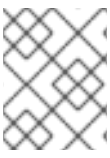
```
      type: vlan
      vlan_id: {get_param: InternalApiNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: InternalApiIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: TenantNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: StorageMgmtNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: StorageMgmtIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: StorageNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: StorageIpSubnet}
```

3. Create the OVS bridges to the Compute node.

```
    -
       type: ovs_bridge
       name: br-sriov
       use_dhcp: false
       members:
        -
           type: interface
           name: nic3
    -
       type: ovs_bridge
       name: br-sriov2
       use_dhcp: false
       members:
        -
           type: interface
           name: nic4
```

3.3.6. Modify compute.yaml

Add the following parameters to the **`compute.yaml`** file you created for your HCI deployment.



NOTE

HCI includes compute parameters not shown in the example YAML files in this guide. See the Configuring Resource Isolation on Hyper-Converged Nodes for details.

1. Create the bridge and interface for an isolated network.

```
-
  type: ovs_bridge
  name: br-isolated
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: ens1f1
      # force the MAC address of the bridge to this interface
      primary: true
```

2. Assign VLANs to this bridge.

```
-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: vlan
  vlan_id: {get_param: TenantNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: StorageMgmtIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: StorageIpSubnet}
```

3. Create interfaces to the controller node.

```
-
  type: interface
  name: ens2f0
  use_dhcp: false
  defroute: false

-
  type: interface
  name: ens2f1
  use_dhcp: false
  defroute: false
```

3.3.7. Run the overcloud_deploy.sh Script

The following example defines the **`openstack overcloud deploy`** Bash script that uses composable roles.

NOTE

This example shows SR-IOV only. You need to add the appropriate templates and YAML files to support HCI. See the Deploying Pure HCI for a sample HCI overcloud deploy command.

```
#!/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/storage-
environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-
sriov.yaml \
-r /home/stack/ospd-11-vlan-ovs-sriov-two-ports-and-hci/custom-roles.yaml
\
-e /home/stack/ospd-11-vlan-ovs-sriov-two-ports-and-hci/network-
environment.yaml
```

NOTE

Reboot the Compute nodes to enforce the **`tuned`** profile after the overcloud is deployed.

3.4. 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.

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

2. Create a flavor.

   ```
   # openstack flavor create compute --ram 4096 --disk 150 --vcpus 4
   ```

   compute is the flavor name, **`4096`** is the memory size in MB, **`150`** is the disk size in GB (default 0G), and **`4`** is the number of vCPUs.

3. Set additional flavor properties.

   ```
   # openstack flavor set --property hw:cpu_policy=dedicated --property
   hw:mem_page_size=large compute
   ```

   compute is the flavor name and the remaining parameters set the other properties for the flavor.

4. Create the network.

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

5. 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
   ```

6. Deploy an instance.

   ```
   # openstack server create --flavor compute --availability-zone sriov
   --image rhel_7.3 --nic port-id=sriov_port sriov_vm
   ```

   Where:

   - **compute** is the flavor name or ID.
   - **`sriov`** is the availability zone for the server.
   - **`rhel_7.3`** is the image (name or ID) used to create an instance.
   - **`sriov_port`** is the NIC on the server.
   - **`sriov_vm`** is the name of the instance.

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

CHAPTER 4. CONFIGURE OVS-DPDK SUPPORT FOR VIRTUAL NETWORKING

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

See Planning your OVS-DPDK Deployment 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.

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.

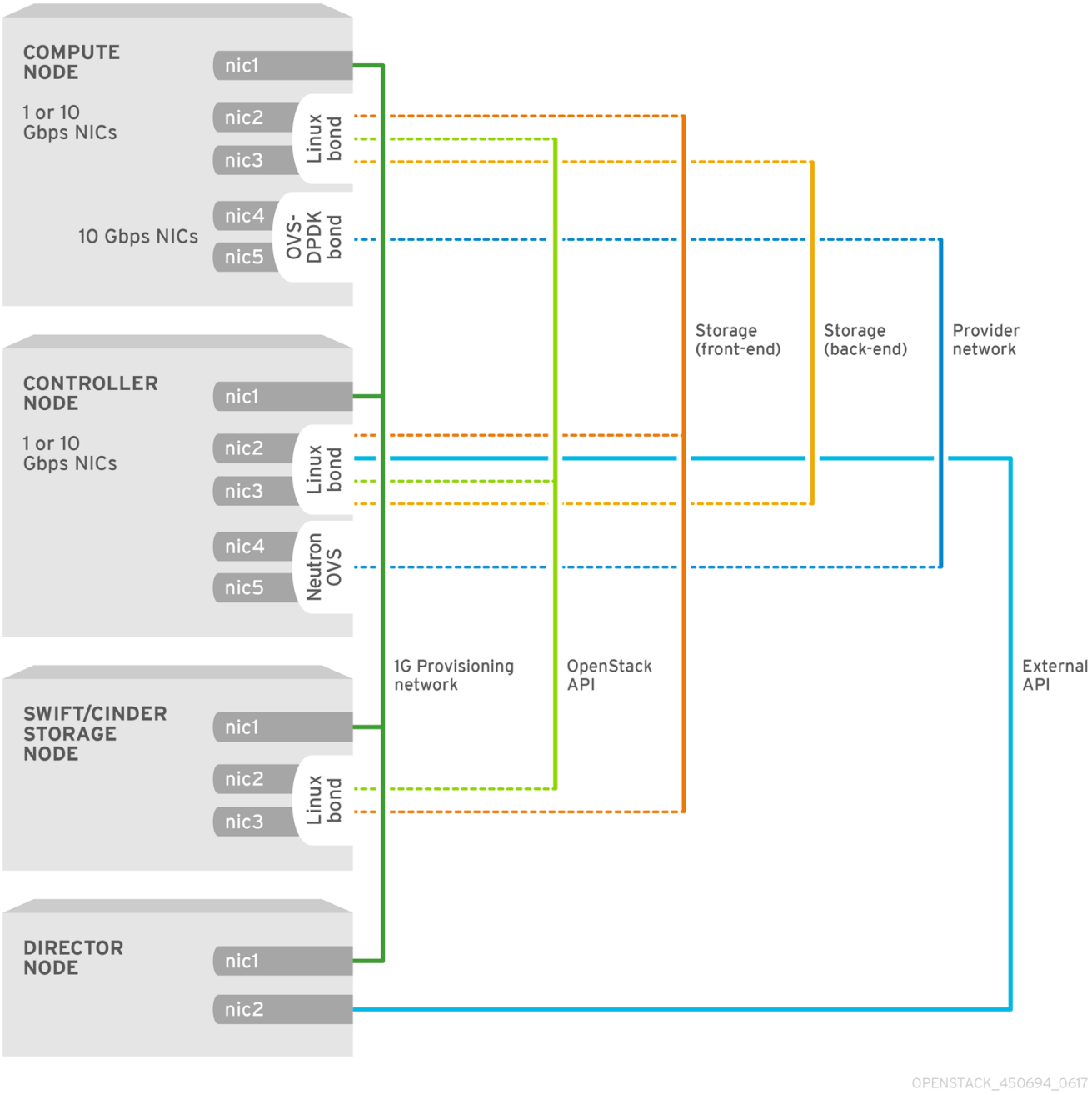
NOTE

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

4.2. OVS-DPDK AND COMPOSABLE ROLES

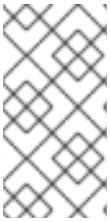
With Red Hat OpenStack Platform 11, you can create custom deployment roles, using the composable roles feature, adding or removing services from each role. For more information on Composable Roles, see Composable Roles and Services.

This image shows a sample OVS-DPDK topology with two bonded ports for the control plane and data plane:



Configuring OVS-DPDK comprises the following tasks:

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



NOTE

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

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

- Red Hat OpenStack Platform 11 with Red Hat Enterprise Linux 7.3
- OVS 2.6
- DPDK 16.04
- Tested NIC. For a list of tested NICs for NFV, see Tested NICs.

NOTE

Red Hat OpenStack Platform 11 operates in OVS client mode for OVS-DPDK deployments.

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

This section describes how to configure OVS-DPDK with one data plane port.

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.

1. Add additional resources.

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

2. Determine the **`NeutronVhostUserSocketDir`** setting.

```
set_ovs_socket_config:
  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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}
```

3. Set the OVS configuration.

```
  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then

ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-
vswitchd.service ]; then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

4. 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
              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}
```

5. 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
              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}
```

6. 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 post-install.yaml

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

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g'
$tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep
-q neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch .
other_config:pmd-cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

4.3.3. Modify network-environment.yaml

See Planning your OVS-DPDK Deployment 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.

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-
ovs-dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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: 'dpdk_data:br-link0'
```

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

```
NeutronNetworkVLANRanges: 'dpdk_data:22:22'
```

-

This example sets the VLAN ranges on the physical network (**dpdk_data**).

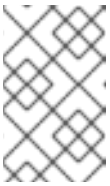
5. Under **parameter_defaults**, set the OVS-DPDK configuration parameters.

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: "1,17,9,25"
```



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: "2048,2048"
```

This is a comma-separated string, in ascending order of the CPU socket. This example assumes a 2 NUMA node configuration and pre-allocates 2048 MB of huge pages to socket 0, and pre-allocates 2048 MB to socket 0. If you have a single NUMA node system, set this value to *1024,0*.

c. Set the DPDK driver type and the datapath type for OVS bridges.

```
NeutronDpdkDriverType: "vfio-pci"
NeutronDatapathType: "netdev"
```

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: 2048
```

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

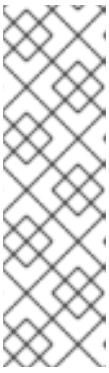
```
NovaVcpuPinSet:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
```

9. Under **parameter_defaults**, list 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesF
ilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

10. 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=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 isolated from the host.
The given argument is appended to the **tuned cpu-partitioning** profile.

```
HostIsolatedCoreList:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

12. Under **parameter_defaults**, set a list of logical cores used by OVS-DPDK processes for **dpdk-lcore-mask**. These cores must be mutually exclusive from the list of cores in **NeutronDpdkCoreList** and **NovaVcpuPinSet**.

```
HostCpusList: "0,16,8,24"
```

4.3.4. Modify controller.yaml

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

```
-
  type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      primary: true
    -
      type: interface
      name: nic3
```

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 to the Compute node.

```
-
  type: ovs_bridge
  name: br-link0
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic4
```

4.3.5. Modify compute.yaml

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

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

```
-
  type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      primary: true
    -
      type: interface
      name: nic3
```

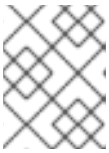
2. Assign VLANs to this Linux bond.

```
-
  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 a bridge with a DPDK port to link to the controller.

```
    -
      type: ovs_user_bridge
      name: br-link0
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      members:
        -
          type: ovs_dpdk_port
          name: dpdk0
          members:
            -
              type: interface
              name: nic4
```



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.

4.3.6. Run the overcloud_deploy.sh Script

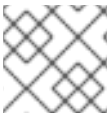
The following example defines the **openstack overcloud deploy** command for the OVS-DPDK environment within a bash script:

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
```

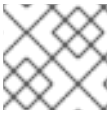
```
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-
dpdk.yaml \
-e /home/stack/ospd-11-vlan-dpdk-single-port-ctlplane-bonding/network-
environment.yaml
```

- **`/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

Reboot the Compute nodes to enforce the **`tuned`** profile after the overcloud is deployed.



NOTE

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

4.4. CONFIGURE TWO-PORT OVS-DPDK WITH VLAN TUNNELLING

This section describes how to configure OVS-DPDK with two data plane ports.

4.4.1. Modify first-boot.yaml

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

1. Add additional resources.

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

2. Determine the **`NeutronVhostUserSocketDir`** setting.

```
  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}
```

3. Set the OVS configuration.

```
  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then

ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-
vswitchd.service ]; then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

4. 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
              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}
```

5. 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
              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}
```

6. 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}
```

4.4.2. Modify post-install.yaml

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

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g'
$tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep
-q neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch .
other_config:pmd-cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

4.4.3. Modify network-environment.yaml

See Planning your OVS-DPDK Deployment 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.

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-
ovs-dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.yaml
```

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

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

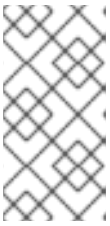
3. Under **parameter_defaults**, map the physical network to the virtual bridge.

```
NeutronBridgeMappings: 'dpdk_mgmt:br-link0,dpdk_data:br-link1'
```

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

```
NeutronNetworkVLANRanges: 'dpdk_mgmt:22:22,dpdk_data:25:28'
```

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

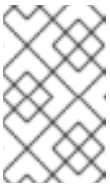


NOTE

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: "1,17,9,25"
```



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: "2048,2048"
```

This is a comma-separated string, in ascending order of the CPU socket. This example assumes a 2 NUMA node configuration and pre-allocates 2048 MB of huge pages to socket 0, and pre-allocates 2048 MB to socket 0. If you have a single NUMA node system, set this value to *1024,0*.

c. Set the DPDK driver type and the datapath type for OVS bridges.

```
NeutronDpdkDriverType: "vfio-pci"
NeutronDatapathType: "netdev"
```

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: 2048
```

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

```
NovaVcpuPinSet:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
```

9. Under **parameter_defaults**, list 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesF
ilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

10. 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=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 get a DHCP allocation.

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

```
HostIsolatedCoreList:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

12. Under **parameter_defaults**, set a list of logical cores used by OVS-DPDK processes for **dpdk-lcore-mask**. These cores must be mutually exclusive from the list of cores in **NeutronDpdkCoreList** and **NovaVcpuPinSet**.

```
HostCpusList: "0,16,8,24"
```

4.4.4. Modify controller.yaml

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

```
-
  type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      primary: true
    -
      type: interface
      name: nic3
```

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 two OVS bridges to the Compute node.

```
-
  type: ovs_bridge
  name: br-link0
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic4
-
  type: ovs_bridge
  name: br-link1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic5
```

4.4.5. Modify compute.yaml

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

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

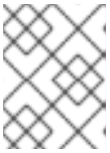
```
-
  type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      primary: true
    -
      type: interface
      name: nic3
```

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 two bridges with a DPDK port each to link to the controller.

```
-
  type: ovs_user_bridge
  name: br-link0
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: ovs_dpdk_port
      name: dpdk0
      members:
        -
          type: interface
          name: nic4
-
  type: ovs_user_bridge
  name: br-link1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: ovs_dpdk_port
      name: dpdk1
      members:
        -
          type: interface
          name: nic5
```



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.

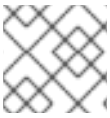
4.4.6. Run the overcloud_deploy.sh Script

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 /home/stack/ospd-11-vlan-dpdk-two-ports-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

Reboot the Compute nodes to enforce the **tuned** profile after the overcloud is deployed.

NOTE

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

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

This section describes how to configure OVS-DPDK with two data plane ports in an OVS-DPDK bond.

4.5.1. Modify first-boot.yaml

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

1. Add additional resources.

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

2. Determine the **NeutronVhostUserSocketDir** setting.

```
  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}
```

3. Set the OVS configuration.

```
  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then

ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-
vswitchd.service ]; then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

4. 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
              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}
```

5. 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
              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}
```

6. 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}
```

4.5.2. Modify post-install.yaml

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

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g'
$tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
```

```
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep
-q neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch .
other_config:pmd-cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

4.5.3. Modify network-environment.yaml

See Planning your OVS-DPDK Deployment 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.

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
  OS::TripleO::NodeExtraConfigPost: post-install.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.

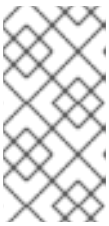
```
NeutronBridgeMappings: 'dpdk_mgmt:br-link'
```

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

```
NeutronNetworkVLANRanges: 'dpdk_mgmt:22:22'
```

   This example sets the VLAN ranges on the physical network (**dpdk_mgmt**).

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: "1,17,9,25"
```

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: "1024,1024"
  ```

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

  c. Set the DPDK driver type and the datapath type for OVS bridges.

  ```
  NeutronDpdkDriverType: "vfio-pci"
  NeutronDatapathType: "netdev"
  ```

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: 2048
   ```

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

   ```
   NovaVcpuPinSet:
   "2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
   ```

9. Under **`parameter_defaults`**, list 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesF
ilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

10. 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=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 isolated from the host.
The given argument is appended to the tuned **cpu-partitioning** profile.

```
HostIsolatedCoreList:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

12. Under **parameter_defaults**, set a list of logical cores used by OVS-DPDK processes for **dpdk-lcore-mask**. These cores must be mutually exclusive from the list of cores in **NeutronDpdkCoreList** and **NovaVcpuPinSet**.

```
HostCpusList: "0,16,8,24"
```

4.5.4. Modify controller.yaml

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

```
-
  type: linux_bond
  name: bond1
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      primary: true
```

```
      -
        type: interface
        name: nic3
```

2. Assign VLANs to this Linux bond.

```
    -
      type: vlan
      vlan_id: {get_param: InternalApiNetworkVlanID}
      device: bond1
      addresses:
        -
          ip_netmask: {get_param: InternalApiIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: ExternalNetworkVlanID}
      device: bond1
      addresses:
        -
          ip_netmask: {get_param: ExternalIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: TenantNetworkVlanID}
      device: bond1
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
```

3. Create the OVS bridge to the Compute node.

```
    -
      type: ovs_bridge
      name: br-link
      use_dhcp: false
      members:
        -
          type: interface
          name: nic4
```

4.5.5. Modify compute.yaml

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

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

```
      -
        type: interface
        name: nic3
```

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}
```

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

```
  -
    type: ovs_user_bridge
    name: br-link
    use_dhcp: false
    members:
      -
        type: ovs_dpdk_bond
        name: bond_dpdk0
        members:
          -
            type: ovs_dpdk_port
            name: dpdk0
            members:
              -
                type: interface
                name: nic4
          -
            type: ovs_dpdk_port
            name: dpdk1
            members:
              -
                type: interface
                name: nic5
```

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.

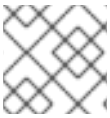
4.5.6. Run the overcloud_deploy.sh Script

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 /home/stack/ospd-11-vlan-ovs-dpdk-bonding-dataplane-bonding-
ctlplane/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

Reboot the Compute nodes to enforce the **tuned** profile after the overcloud is deployed.

NOTE

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

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

This section describes how to configure OVS-DPDK with VXLAN tunnelling.

4.6.1. Modify first-boot.yaml

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

1. Add additional resources.

```
resources:
  userdata:
    type: OS::Heat::MultipartMime
```

```
    properties:
      parts:
      - config: {get_resource: set_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}
```

2. Determine the **`NeutronVhostUserSocketDir`** setting.

```
  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}
```

3. Set the OVS configuration.

```
set_ovs_config:
  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
            if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then

ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
            elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service
]; then
              ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
            fi
            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
          fi
        params:
          $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

4. 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
              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}
```

5. 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
              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}
```

6. 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}
```

4.6.2. Modify post-install.yaml

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

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      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
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i '/After=.*/s/network.target//g'
$tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                grep -q "Before=.*" $tuned_service
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                else
                  sed -i '/After/i Before=network.target
```

```
openvswitch.service' $tuned_service
                fi
              fi
              systemctl daemon-reload
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
```

4.6.3. Modify network-environment.yaml

See Planning your OVS-DPDK Deployment 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.

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-
ovs-dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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: "1,17,9,25"
```

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: "2048,2048"
```

This is a comma-separated string, in ascending order of the CPU socket. This example assumes a 2 NUMA node configuration and pre-allocates 2048 MB of huge pages to socket 0, and pre-allocates 2048 MB to socket 0. If you have a single NUMA node system, set this value to *1024,0*.

c. Set the DPDK driver type and the datapath type for OVS bridges.

```
NeutronDpdkDriverType: "vfio-pci"
NeutronDatapathType: "netdev"
```

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: 2048
```

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

```
NovaVcpuPinSet:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
```

7. Under **parameter_defaults**, list 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesF
ilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

8. 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=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 isolated from the host.
The given argument is appended to the tuned **cpu-partitioning** profile.

```
HostIsolatedCoreList:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,
29,30,31"
```

10. Under **parameter_defaults**, set a list of logical cores used by OVS-DPDK processes for **dpdk-lcore-mask**. These cores must be mutually exclusive from the list of cores in **NeutronDpdkCoreList** and **NovaVcpuPinSet**.

```
HostCpusList: "0,16,8,24"
```

4.6.4. Modify controller.yaml

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

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

```
      -
        type: interface
        name: nic3
```

2. Assign VLANs to this Linux bond.

```
  -
    type: vlan
    vlan_id: {get_param: InternalApiNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: InternalApiIpSubnet}
  -
    type: vlan
    vlan_id: {get_param: StorageNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: StorageIpSubnet}
  -
    type: vlan
    vlan_id: {get_param: StorageMgmtNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: StorageMgmtIpSubnet}
  -
    type: vlan
    vlan_id: {get_param: ExternalNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: ExternalIpSubnet}
```

3. Create the OVS bridge to the Compute node.

```
-
  type: ovs_bridge
  name: br-link
  use_dhcp: false
  members:
    -
      type: interface
      name: nic4
    -
      type: vlan
      vlan_id: {get_param: TenantNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
```

4.6.5. Modify compute.yaml

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

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

```
-
  type: linux_bond
  name: bond_api
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic2
      primary: true
    -
      type: interface
      name: nic3
```

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

```
        name: dpdk0
        members:
          -
            type: interface
            name: nic4
```

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.

4.6.6. Run the overcloud_deploy.sh Script

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

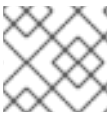
```
#!/bin/bash

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 /home/stack/ospd-11-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

Reboot the Compute nodes to enforce the **tuned** profile after the overcloud is deployed.



NOTE

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

4.7. CONFIGURE OVS-DPDK COMPOSABLE ROLE

This section describes how to configure composable roles for OVS-DPDK and SR-IOV with VLAN tunnelling. The process to create and deploy a composable role includes:

- Defining the new role in a local copy of the **`role_data.yaml`** file.
- Creating the OpenStack flavor that uses this new role.
- Modifying the **`network_environment.yaml`** file to include this new role.
- Deploying the overcloud with this updated set of roles.

In this example, the **ComputeOvsDpdk** and **ComputeSriov** are composable role for compute nodes to enable DPDK or SR-IOV only on the nodes that have the SR-IOV NICs. The existing set of default roles provided by the Red Hat OpenStack Platform is stored in the **`/home/stack/roles_data.yaml`** file.

4.7.1. Modify roles_data.yaml to Create Composable Roles

Copy the **`roles_data.yaml`** file to your **`/home/stack/templates`** directory and add the new **ComputeOvsDpdk** and `**ComputeSriov** roles.

1. Define the OVS-DPDK composable role.

```
- name: ComputeOvsDpdk
  CountDefault: 1
  HostnameFormatDefault: compute-ovs-dpdk-%index%
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsDpdkAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

   In this example, all the services of the new **`ComputeOvsDpdk`** role are same as that of a regular Compute role, with the exception of **`ComputeNeutronOvsAgent`**. Replace **`ComputeNeutronOvsAgent`** with **`ComputeNeutronOvsDpdkAgent`** to map to the OVS-DPDK service.

2. Define the SR-IOV composable role.

```
- name: ComputeSriov
  CountDefault: 1
  HostnameFormatDefault: compute-sriov-%index%
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

4.7.2. Modify network-environment.yaml for the New Composable Roles

Add the resource mapping for the OVS-DPDK and SR-IOV services to the **network-environment.yaml** file along with the network configuration for this node as follows:

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

  OS::TripleO::Services::ComputeNeutronOvsDpdkAgent: /usr/share/openstack-
tripleo-heat-templates/puppet/services/neutron-ovs-dpdk-agent.yaml
  OS::TripleO::Services::NeutronSriovAgent: /usr/share/openstack-tripleo-
heat-templates/puppet/services/neutron-sriov-agent.yaml

  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.yaml
```

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 Planning your OVS-DPDK Deployment 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.

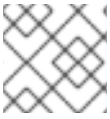
4.7.3. Run the overcloud_deploy.sh Script

The following example defines the **openstack overcloud deploy** Bash script that uses composable roles:

```
# #!/bin/bash

openstack overcloud deploy \
--templates \
-r /home/stack/ospd-11-vlan-dpdk-sriov-single-port-composable-roles/roles-
data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
-e /home/stack/ospd-11-vlan-dpdk-sriov-single-port-composable-
roles/network-environment.yaml
```

/home/stack/templates/roles_data.yaml is the location of the updated **roles_data.yaml** file, which defines the OVS-DPDK and the SR-IOV composable roles.



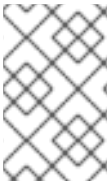
NOTE

Reboot the Compute nodes to enforce the **tuned** profile after the overcloud is deployed.

4.8. 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.

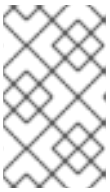
NOTE

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: 2000
```



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
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic4
      mtu: 2000
-
  type: ovs_bridge
  name: br-link1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic5
      mtu: 2000
```

3. Set the MTU value on the OVS-DPDK interfaces in the **compute.yaml** file.

```
-
  type: ovs_user_bridge
  name: br-link0
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: ovs_dpdk_port
      name: dpdk0
```

```
          mtu: 2000
          ovs_extra:
          - set interface $DEVICE options:n_rxq=2
          - set interface $DEVICE mtu_request=$MTU
          members:
            -
              type: interface
              name: nic4
    -
      type: ovs_user_bridge
      name: br-link1
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      members:
        -
          type: ovs_dpdk_port
          name: dpdk1
          mtu: 2000
          ovs_extra:
          - set interface $DEVICE options:n_rxq=2
          - set interface $DEVICE mtu_request=$MTU
          members:
            -
              type: interface
              name: nic5
```

4.9. SET MULTIQUEUE FOR OVS-DPDK INTERFACES

To set set same number of queues for interfaces in OVS-DPDK on the Compute node, modify the **`compute.yaml`** file as follows:

```
  -
    type: ovs_user_bridge
    name: br-link0
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    members:
      -
        type: ovs_dpdk_port
        name: dpdk0
        mtu: 2000
        ovs_extra:
        - set interface $DEVICE options:n_rxq=2
        - set interface $DEVICE mtu_request=$MTU
        members:
          -
            type: interface
            name: nic4
  -
    type: ovs_user_bridge
    name: br-link1
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    members:
      -
```

```
        type: ovs_dpdk_port
        name: dpdk1
        mtu: 2000
        ovs_extra:
        - set interface $DEVICE options:n_rxq=2
        - set interface $DEVICE mtu_request=$MTU
        members:
          -
            type: interface
            name: nic5
```

4.10. KNOWN LIMITATIONS

There are certain limitations when configuring OVS-DPDK with Red Hat OpenStack Platform 11 for the NFV use case:

- Use Linux bonds for control plane networks. Ensure both PCI devices used in the bond are on the same NUMA node for optimum performance. Neutron Linux bridge configuration is not supported by Red Hat.
- Huge pages are required for every instance running on the hosts with OVS-DPDK. If huge pages are not present in the guest, the interface appears but does not function.
- 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 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.
- Do not configure OVS-DPDK and SR-IOV on the same Compute node.

4.11. 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.

```
# openstack aggregate create --zone=dpdk dpdk
# openstack aggregate add host dpdk compute-ovs-dpdk-0.localdomain
```

2. Create a flavor.

```
# openstack flavor create m1.medium_huge_4cpu --ram 4096 --disk 150
--vcpus 4
```

Here, **m1.medium_huge_4cpu** is the flavor name, **4096** is the memory size in MB, **150** is the disk size in GB (default 0G), and **4** is the number of vCPUs.

3. Set additional flavor properties.

```
# openstack flavor set --property hw:cpu_policy=dedicated --property
hw:mem_page_size=large m1.medium_huge_4cpu
```

Here, **m1.medium_huge_4cpu** is the flavor name and the remaining parameters set the other properties for the flavor.

4. Create the network.

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

5. Deploy an instance.

```
# openstack server create --flavor m1.medium_huge_4cpu --
availability-zone dpdk --image rhel_7.3 --nic net-id=net1
```

Where:

- **m1.medium_huge_4cpu** is the flavor name or ID.
- **dpdk** is the availability zone for the server.
- **rhel_7.3** is the image (name or ID) used to create an instance.
- **net1** is the NIC on the server.

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

For using **multi-queue** with OVS-DPDK, there are a couple of additional steps that you need to include in the above procedure. Before you create a flavor, perform the following steps:

1. Set the image properties.

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

Here, **hw_vif_multiqueue_enabled=true** is a property on this image to enable multiqueue, **<image-id>** is the name or ID of the image to modify.

2. Set additional flavor properties.

```
# openstack flavor set m1.vm_mq set hw:vif_multiqueue_enabled=true
```

Here, **m1.vm_mq** is the flavor ID or name, and the remaining options enable multiqueue for the flavor.

4.12. OPTIMIZING PERFORMANCE WITH EMULATOR THREAD PINNING

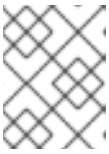
To improve performance, you can pin the Qemu emulator thread to an alternate core.

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`**.

4.13. 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`**.

```
# 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 their status:

```
# tuna -t ovs-vswitchd -CP
thread  ctxt_switches pid SCHED_ rtpri affinity voluntary
nonvoluntary       cmd
3161 OTHER  0     6 765023       614 ovs-vswitchd
3219   OTHER  0     6      1         0    handler24
3220   OTHER  0     6      1         0    handler21
3221   OTHER  0     6      1         0    handler22
3222   OTHER  0     6      1         0    handler23
3223   OTHER  0     6      1         0    handler25
3224   OTHER  0     6      1         0    handler26
3225   OTHER  0     6      1         0    handler27
3226   OTHER  0     6      1         0    handler28
3227   OTHER  0     6      2         0    handler31
3228   OTHER  0     6      2         4    handler30
3229   OTHER  0     6      2         5    handler32
3230   OTHER  0     6 953538       431   revalidator29
3231   OTHER  0     6   1424258       976   revalidator33
3232   OTHER  0     6   1424693       836   revalidator34
3233   OTHER  0     6 951678       503   revalidator36
3234   OTHER  0     6   1425128       498   revalidator35
*3235   OTHER  0     4 151123        51        pmd37*
*3236   OTHER  0    20 298967        48        pmd38*
3164   OTHER  0     6  47575         0  dpdk_watchdog3
3165   OTHER  0     6 237634         0   vhost_thread1
3166   OTHER  0     6   3665         0        urcu2
```

CHAPTER 5. 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 11 Documentation Suite

Table 5.1. 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 Functions Virtualization Planning and Prerequisite Guide.

APPENDIX A. SAMPLE SR-IOV YAML FILES

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

A.1. SAMPLE SR-IOV COMPOSABLE ROLES YAML FILES

A.1.1. `roles_data.yaml`

```
- name: Controller
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackup
    - OS::TripleO::Services::CinderScheduler
    - OS::TripleO::Services::CinderVolume
    - OS::TripleO::Services::Congress
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Keystone
    - OS::TripleO::Services::GlanceApi
    - OS::TripleO::Services::HeatApi
    - OS::TripleO::Services::HeatApiCfn
    - OS::TripleO::Services::HeatApiCloudwatch
    - OS::TripleO::Services::HeatEngine
    - OS::TripleO::Services::MySQL
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronDhcpAgent
    - OS::TripleO::Services::NeutronL3Agent
    - OS::TripleO::Services::NeutronMetadataAgent
    - OS::TripleO::Services::NeutronApi
    - OS::TripleO::Services::NeutronCorePlugin
    - OS::TripleO::Services::NeutronOvsAgent
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::HAproxy
    - OS::TripleO::Services::Keepalived
    - OS::TripleO::Services::Memcached
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::NovaConductor
    - OS::TripleO::Services::MongoDb
    - OS::TripleO::Services::NovaApi
    - OS::TripleO::Services::NovaPlacement
    - OS::TripleO::Services::NovaMetadata
    - OS::TripleO::Services::NovaScheduler
    - OS::TripleO::Services::NovaConsoleauth
    - OS::TripleO::Services::NovaVncProxy
    - OS::TripleO::Services::Ec2Api
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::SwiftProxy
```

```
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::CeilometerApi
    - OS::TripleO::Services::CeilometerCollector
    - OS::TripleO::Services::CeilometerExpirer
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::Horizon
    - OS::TripleO::Services::GnocchiApi
    - OS::TripleO::Services::GnocchiMetricd
    - OS::TripleO::Services::GnocchiStatsd
    - OS::TripleO::Services::ManilaApi
    - OS::TripleO::Services::ManilaScheduler
    - OS::TripleO::Services::ManilaBackendGeneric
    - OS::TripleO::Services::ManilaBackendNetapp
    - OS::TripleO::Services::ManilaBackendCephFs
    - OS::TripleO::Services::ManilaShare
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::IronicApi
    - OS::TripleO::Services::IronicConductor
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::OpenDaylightApi
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::OVNDBs
    - OS::TripleO::Services::NeutronML2FujitsuCfab
    - OS::TripleO::Services::NeutronML2FujitsuFossw
    - OS::TripleO::Services::CinderHPELeftHandISCSI
    - OS::TripleO::Services::Etcd
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::OctaviaApi
    - OS::TripleO::Services::OctaviaHealthManager
    - OS::TripleO::Services::OctaviaHousekeeping
    - OS::TripleO::Services::OctaviaWorker


- name: ComputeSriov
  CountDefault: 1
  HostnameFormatDefault: 'compute-sriov-%index%'
  disable_upgrade_deployment: True
```

```
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

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: boot_config}
      - config: {get_resource: compute_kernel_args}

  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile and activate it, we
should create relevant repos.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  # 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
              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

              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#m
aking-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. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  ComputeHostnameFormat:
    type: string
    default: ""

resources:
  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      # Do this on CREATE/UPDATE (which is actually the default)
      actions: ['CREATE', 'UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      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
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                grep -q "Before=.*" $tuned_service
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                else
                  sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                fi
              fi
              systemctl daemon-reload
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
```

A.1.4. `network.environment.yaml`

```
resource_registry:
  # Specify the relative/absolute path to the config files you want to use
for override the default.
  OS::TripleO::ComputeSriov::Net::SoftwareConfig: nic-configs/compute-
sriov.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::Services::NeutronSriovAgent: /usr/share/openstack-tripleo-
heat-templates/puppet/services/neutron-sriov-agent.yaml

  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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: 10.35.141.64/28
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  InternalApiAllocationPools: [{'start': '10.10.10.100', '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': '10.35.141.66', 'end':
'10.35.141.69'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 10.35.141.78
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.0.24.1
  # Generally the IP of the Undercloud
  EC2MetadataIp: 192.0.24.1
  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: 'datacentre:br-ex,tenant:br-link'
  NeutronBridgeMappings: 'tenant:br-link'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  ###NeutronNetworkVLANRanges:
'datacentre:419:419,tenant:420:420,tenant:421:421'
  NeutronNetworkVLANRanges: 'tenant:420:420,tenant:421:421'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  OvercloudComputeSriovFlavor: sriov

  # Number of nodes to deploy.
  ControllerCount: 1
  ComputeSriovCount: 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%'
  ComputeSriovHostnameFormat: 'compute-sriov-%index%'
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'

  ############################
  # SR-IOV 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: "ens2f1"
      physical_network: "tenant"
  # List of supported pci vendor devices in the format VendorID:ProductID.
  NeutronSupportedPCIVendorDevs: ['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:ens2f1"
  # 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: "ens2f1:5"

  ############################
  # Additional config      #
  ############################
  # List of scheduler available filters
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthro
ugh_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','ComputeCapabilities
Filter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGro
upAffinityFilter','PciPassthroughFilter','NUMATopologyFilter']
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
```

A.1.5. `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
  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
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations)
that will be added to resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: eno1
              use_dhcp: false
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
```

```
                next_hop: {get_param: EC2MetadataIp}
            -
              type: interface
              name: ens1f1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
            -
              type: vlan
              vlan_id: {get_param: InternalApiNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: InternalApiIpSubnet}
            -
              type: vlan
              vlan_id: {get_param: TenantNetworkVlanID}
              device: ens1f1
              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: ens1f1
              addresses:
              -
                ip_netmask: {get_param: ExternalIpSubnet}
              routes:
                -
                  default: true
                  next_hop: {get_param: ExternalInterfaceDefaultRoute}
            -
              type: ovs_bridge
              name: br-link
              use_dhcp: false
              members:
                -
                  type: interface
                  name: ens2f1


outputs:
```

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

A.1.6. compute-sriov.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
  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
  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: eno1
              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: interface
              name: ens1f1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
            -
              type: vlan
```

```
              vlan_id: {get_param: InternalApiNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: InternalApiIpSubnet}
            -
              type: vlan
              vlan_id: {get_param: TenantNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: TenantIpSubnet}
            -
              type: vlan
              vlan_id: {get_param: StorageNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: StorageIpSubnet}
            -
              type: interface
              name: ens2f1
              use_dhcp: false
              defroute: false

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

A.1.7. `overcloud_deploy.sh`

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-r /home/stack/ospd-11-vlan-sriov-single-port-composable-roles/roles-
data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
-e /home/stack/ospd-11-vlan-sriov-single-port-composable-roles/network-
environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

A.2. SAMPLE SR-IOV HCI COMPOSABLE ROLES YAML FILES

A.2.1. `custom-roles.yaml`

```
# Specifies which roles (groups of nodes) will be deployed
# Note this is used as an input to the various *.j2.yaml
# jinja2 templates, so that they are converted into *.yaml
# during the plan creation (via a mistral action/workflow).
#
```

```
# The format is a list, with the following format:
#
# * name: (string) mandatory, name of the role, must be unique
#
# CountDefault: (number) optional, default number of nodes, defaults to 0
# sets the default for the {{role.name}}Count parameter in overcloud.yaml
#
# HostnameFormatDefault: (string) optional default format string for
hostname
# defaults to '%stackname%-{{role.name.lower()}}-%index%'
# sets the default for {{role.name}}HostnameFormat parameter in
overcloud.yaml
#
# disable_constraints: (boolean) optional, whether to disable Nova and
Glance
# constraints for each role specified in the templates.
#
# disable_upgrade_deployment: (boolean) optional, whether to run the
# ansible upgrade steps for all services that are deployed on the role. If
set
# to True, the operator will drive the upgrade for this role's nodes.
#
# upgrade_batch_size: (number): batch size for upgrades where tasks are
# specified by services to run in batches vs all nodes at once.
# This defaults to 1, but larger batches may be specified here.
#
# ServicesDefault: (list) optional default list of services to be deployed
# on the role, defaults to an empty list. Sets the default for the
# {{role.name}}Services parameter in overcloud.yaml

- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackup
    - OS::TripleO::Services::CinderScheduler
    - OS::TripleO::Services::CinderVolume
    - OS::TripleO::Services::Congress
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Keystone
    - OS::TripleO::Services::GlanceApi
    - OS::TripleO::Services::HeatApi
    - OS::TripleO::Services::HeatApiCfn
    - OS::TripleO::Services::HeatApiCloudwatch
    - OS::TripleO::Services::HeatEngine
    - OS::TripleO::Services::MySQL
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronDhcpAgent
    - OS::TripleO::Services::NeutronL3Agent
    - OS::TripleO::Services::NeutronMetadataAgent
```

```
    - OS::TripleO::Services::NeutronApi
    - OS::TripleO::Services::NeutronCorePlugin
    - OS::TripleO::Services::NeutronOvsAgent
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::HAproxy
    - OS::TripleO::Services::Keepalived
    - OS::TripleO::Services::Memcached
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::NovaConductor
    - OS::TripleO::Services::MongoDb
    - OS::TripleO::Services::NovaApi
    - OS::TripleO::Services::NovaPlacement
    - OS::TripleO::Services::NovaMetadata
    - OS::TripleO::Services::NovaScheduler
    - OS::TripleO::Services::NovaConsoleauth
    - OS::TripleO::Services::NovaVncProxy
    - OS::TripleO::Services::Ec2Api
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::SwiftProxy
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::CeilometerApi
    - OS::TripleO::Services::CeilometerCollector
    - OS::TripleO::Services::CeilometerExpirer
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::Horizon
    - OS::TripleO::Services::GnocchiApi
    - OS::TripleO::Services::GnocchiMetricd
    - OS::TripleO::Services::GnocchiStatsd
    - OS::TripleO::Services::ManilaApi
    - OS::TripleO::Services::ManilaScheduler
    - OS::TripleO::Services::ManilaBackendGeneric
    - OS::TripleO::Services::ManilaBackendNetapp
    - OS::TripleO::Services::ManilaBackendCephFs
    - OS::TripleO::Services::ManilaShare
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::IronicApi
    - OS::TripleO::Services::IronicConductor
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::OpenDaylightApi
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Collectd
```

```
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::OVNDBs
    - OS::TripleO::Services::NeutronML2FujitsuCfab
    - OS::TripleO::Services::NeutronML2FujitsuFossw
    - OS::TripleO::Services::CinderHPELeftHandISCSI
    - OS::TripleO::Services::Etcd
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::OctaviaApi
    - OS::TripleO::Services::OctaviaHealthManager
    - OS::TripleO::Services::OctaviaHousekeeping
    - OS::TripleO::Services::OctaviaWorker

- name: Compute
  CountDefault: 1
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

A.2.2. `first-boot.yaml`

```
heat_template_version: 2014-10-16
######################################################################
########################################
# This YAML file is tuned for the NFV use cases, HCI additional
configuration is completion to the following file #
######################################################################
########################################
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.
      The given args will be appended to the tuned cpu-partitioning
profile.
      Ex. HostCpusList: '4-12' will tune cores from 4-12
    type: string
    default: ""

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

  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile and activate it, we
should create relevant repos.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  # Verify the logs on /var/log/cloud-init.log on the overcloud node
  tuned_config:
    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
              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 $KERNEL_ARGS
isolcpus=$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#m
aking-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.2.3. `post-install.yaml`

```
heat_template_version: 2014-10-16
######################################################################
########################################
# This YAML file is tuned for the NFV use cases, HCI additional
configuration is completion to the following file #
######################################################################
########################################
description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  ComputeHostnameFormat:
    type: string
    default: ""

resources:
  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      # Do this on CREATE/UPDATE (which is actually the default)
      actions: ['CREATE', 'UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      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
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                grep -q "Before=.*" $tuned_service
                if [ "$?" -eq 0 ]; then
                  sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                else
                  sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                fi
              fi

              systemctl daemon-reload
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
```

A.2.4. `network.environment.yaml`

```
##############################################################################
#########################################
# This YAML file is tuned for the NFV use cases, HCI additional
configuration is completion to the following file #
##############################################################################
#########################################
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
  OS::TripleO::NodeExtraConfigPost: post-install.yaml

parameter_defaults:
  # Customize all these values to match the local environment
  InternalApiNetCidr: 10.10.10.0/24
  TenantNetCidr: 10.10.2.0/24
  StorageMgmtNetCidr: 10.10.3.0/24
  StorageNetCidr: 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'}]
  # Ip allocation pool range for the storage mgmt network.
  StorageMgmtAllocationPools: [{'start': '10.10.3.100', 'end':
'10.10.3.200'}]
  # Ip allocation pool range for the storage network.
  StorageAllocationPools: [{'start': '10.10.4.100', 'end': '10.10.4.200'}]
  # Use an External allocation pool which will leave room for floating IPs
  ExternalAllocationPools: [{'start': '10.35.185.2', 'end':
'10.35.185.13'}]
  # 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
  StorageMgmtNetworkVlanID: 13
  StorageNetworkVlanID: 14
  ExternalNetworkVlanID: 12
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["10.35.28.28","8.8.8.8"]
  # 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: 'datacentre:br-isolated,tenant:br-
sriov,tenant2:br-sriov2'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges:
'datacentre:419:419,tenant:420:420,tenant2:421:421'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  OvercloudComputeFlavor: compute
  # Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 2
  # NTP server configuration.
  NtpServer: clock.redhat.com
  # Set 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: "ens2f0"
      physical_network: "tenant"
    - devname: "ens2f1"
      physical_network: "tenant2"
  # List of supported pci vendor devices in the format VendorID:ProductID.
  NeutronSupportedPCIVendorDevs: ['8086:10fb','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:ens2f0,tenant2:ens2f1"
  # 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: "ens2f0:7,ens2f1:7"

  # List of scheduler available filters
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthro
ugh_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','ComputeCapabilities
Filter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGro
upAffinityFilter','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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
```

A.2.5. `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
  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
              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: ovs_bridge
              name: br-isolated
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
                  name: nic2
                  # force the MAC address of the bridge to this interface
                  primary: true
                -
```

```
                  type: vlan
                  vlan_id: {get_param: ExternalNetworkVlanID}
                  addresses:
                  -
                    ip_netmask: {get_param: ExternalIpSubnet}
                  routes:
                    -
                      default: true
                      next_hop: {get_param:
ExternalInterfaceDefaultRoute}
                -
                  type: vlan
                  vlan_id: {get_param: InternalApiNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: InternalApiIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: TenantIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: StorageMgmtNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageMgmtIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: StorageNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageIpSubnet}
            -
               type: ovs_bridge
               name: br-sriov
               use_dhcp: false
               members:
                -
                   type: interface
                   name: nic3
            -
               type: ovs_bridge
               name: br-sriov2
               use_dhcp: false
               members:
                -
                   type: interface
                   name: nic4

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

A.2.6. `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
  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: eno1
              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: ovs_bridge
              name: br-isolated
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
```

```
                  name: ens1f1
                  # force the MAC address of the bridge to this interface
                  primary: true
                -
                  type: vlan
                  vlan_id: {get_param: InternalApiNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: InternalApiIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: TenantIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: StorageMgmtNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageMgmtIpSubnet}
                -
                  type: vlan
                  vlan_id: {get_param: StorageNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageIpSubnet}
              -
                type: interface
                name: ens2f0
                use_dhcp: false
                defroute: false

              -
                type: interface
                name: ens2f1
                use_dhcp: false
                defroute: false

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

A.2.7. `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/storage-
environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-
```

```
sriov.yaml \
-r /home/stack/ospd-11-vlan-ovs-sriov-two-ports-and-hci/custom-roles.yaml
\
-e /home/stack/ospd-11-vlan-ovs-sriov-two-ports-and-hci/network-
environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

APPENDIX B. SAMPLE OVS-DPDK YAML FILES

This section provides sample OVS-DPDK YAML files as a reference.

B.1. SAMPLE VLAN OVS-DPDK 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
  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: boot_config}
      - config: {get_resource: set_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile an activate it, we
should create FDP repo.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}

  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then
                ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service ];
then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

  # 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
              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
              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#m
aking-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. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
    type: string
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. (pmd-cpu-mask)
    type: string
resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE','UPDATE']
```

```
  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-
cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-
lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

B.1.3. `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-ovs-
dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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': '172.10.2.100', 'end':
'172.10.2.200'}]
  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: 'dpdk_data:br-link0'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'dpdk_data:22:22'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  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: "1,17,9,25"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "2048,2048"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # Datapath type for ovs bridges
  NeutronDatapathType: "netdev"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

```
  ########################
  # Additional settings
  ########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
  # 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,
ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
  HostCpusList: "0,16,8,24"
```

B.1.4. `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
  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
            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: ExternalInterfaceDefaultRoute}
          -
            type: linux_bond
            name: bond_api
            bonding_options: "mode=active-backup"
            use_dhcp: false
            dns_servers: {get_param: DnsServers}
            members:
              -
                type: interface
                name: nic2
                primary: true
              -
                type: interface
                name: nic3
          -
            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
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
                  name: nic4

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

B.1.5. `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
  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
  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
              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: nic2
                  primary: true
                -
                  type: interface
                  name: nic3
            -
              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
                dns_servers: {get_param: DnsServers}
                members:
                  -
                    type: ovs_dpdk_port
                    name: dpdk0
                    members:
                      -
                        type: interface
                        name: nic4

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

B.1.6. `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 /home/stack/ospd-11-vlan-dpdk-single-port-ctlplane-bonding/network-
environment.yaml \
--log-file overcloud_install.log
```

B.2. SAMPLE TWO-PORT VLAN OVS-DPDK 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
  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: boot_config}
      - config: {get_resource: set_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  boot_config:
    type: OS::Heat::CloudConfig
    properties:
```

```
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile an activate it, we
should create FDP repo.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}

  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then
```

```
                ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service ];
then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

  # 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
              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
              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#m
aking-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. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
    type: string
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. (pmd-cpu-mask)
    type: string
resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE','UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
```

```
                grep -q "Before=.*" $tuned_service
                if [ "$?" -eq 0 ]; then
                    sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                else
                    sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                fi
            fi
        }

        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"
        }

        if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
neutron_ovs_dpdk_agent; then
            pmd_cpu_mask=$( get_mask $PMD_CORES )
            host_cpu_mask=$( get_mask $LCORE_LIST )
            ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-
cpu-mask=$pmd_cpu_mask
            ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-
lcore-mask=$host_cpu_mask
            tuned_service_dependency
        fi
```

```
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

B.2.3. `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-ovs-
dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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': '172.10.2.100', 'end':
'172.10.2.200'}]
  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: 'dpdk_mgmt:br-link0,dpdk_data:br-link1'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'dpdk_mgmt:22:22,dpdk_data:25:28'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  OvercloudComputeFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 1
  # NTP server configuration.
  NtpServer: clock.redhat.com
  # Configure the classname of the firewall driver to use for implementing
security groups.
  NeutronOVSFirewallDriver: openvswitch

  # 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: "1,17,9,25"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "2048,2048"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # Datapath type for ovs bridges
  NeutronDatapathType: "netdev"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ##########################
  # Additional settings
  ##########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
```

```
  # 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,
ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
  HostCpusList: "0,16,8,24"
```

B.2.4. `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
  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
              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: ExternalInterfaceDefaultRoute}
        -
          type: linux_bond
          name: bond_api
          bonding_options: "mode=active-backup"
          use_dhcp: false
          dns_servers: {get_param: DnsServers}
          members:
            -
              type: interface
              name: nic2
              primary: true
            -
              type: interface
              name: nic3
        -
          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
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
                  name: nic4
            -
              type: ovs_bridge
              name: br-link1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
                  name: nic5


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

B.2.5. `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
  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
  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
              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: nic2
                  primary: true
                -
                  type: interface
                  name: nic3
            -
              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
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: ovs_dpdk_port
                  name: dpdk0
                  members:
                    -
                      type: interface
                      name: nic4
            -
              type: ovs_user_bridge
              name: br-link1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: ovs_dpdk_port
                  name: dpdk1
                  members:
                    -
                      type: interface
                      name: nic5

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

B.2.6. `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 /home/stack/ospd-11-vlan-dpdk-two-ports-ctlplane-bonding/network-
environment.yaml \
--log-file overcloud_install.log
```

B.3. SAMPLE VLAN OVS-DPDK DATA PLANE BONDING YAML FILES

B.3.1. 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
  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: boot_config}
```

```
      - config: {get_resource: set_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile an activate it, we
should create FDP repo.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}

  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then
                ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service ];
then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

  # 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
              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
              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#m
aking-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.3.2. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
    type: string
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. (pmd-cpu-mask)
    type: string
resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE','UPDATE']

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

```
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
```

```
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-
cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-
lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

B.3.3. `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
  OS::TripleO::NodeExtraConfigPost: post-install.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': '172.10.2.100', 'end':
'172.10.2.200'}]
  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: 'dpdk_mgmt:br-link'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'dpdk_mgmt:22:22'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  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: "1,17,9,25"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "1024,1024"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # Datapath type for ovs bridges
  NeutronDatapathType: "netdev"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ##########################
  # Additional settings
  ##########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
  # 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,
ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
  HostCpusList: "0,16,8,24"
```

B.3.4. `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
  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
              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: ExternalInterfaceDefaultRoute}
          -
            type: linux_bond
            name: bond1
            bonding_options: "mode=active-backup"
            use_dhcp: false
            dns_servers: {get_param: DnsServers}
            members:
              -
                type: interface
                name: nic2
                primary: true
              -
                type: interface
                name: nic3
          -
            type: vlan
            vlan_id: {get_param: InternalApiNetworkVlanID}
            device: bond1
            addresses:
              -
                ip_netmask: {get_param: InternalApiIpSubnet}
          -
            type: vlan
            vlan_id: {get_param: ExternalNetworkVlanID}
            device: bond1
            addresses:
              -
                ip_netmask: {get_param: ExternalIpSubnet}
          -
            type: vlan
            vlan_id: {get_param: TenantNetworkVlanID}
            device: bond1
            addresses:
              -
                ip_netmask: {get_param: TenantIpSubnet}
          -
            type: ovs_bridge
            name: br-link
            use_dhcp: false
            members:
              -
                type: interface
                name: nic4

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

B.3.5. `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
  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
              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: nic2
                  primary: true
                -
                  type: interface
                  name: nic3
            -
              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: ovs_user_bridge
              name: br-link
              use_dhcp: false
              members:
                -
                  type: ovs_dpdk_bond
                  name: bond_dpdk0
                  members:
                    -
                      type: ovs_dpdk_port
                      name: dpdk0
                      members:
                        -
                          type: interface
                          name: nic4
                    -
                      type: ovs_dpdk_port
                      name: dpdk1
                      members:
                        -
                          type: interface
                          name: nic5


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

B.3.6. `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 /home/stack/ospd-11-vlan-ovs-dpdk-bonding-dataplane-bonding-
ctlplane/network-environment.yaml \
--log-file overcloud_install.log
```

B.4. SAMPLE VXLAN OVS-DPDK DATA PLANE BONDING YAML FILES

B.4.1. `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
  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_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}

  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then
                ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service ];
```

```
then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

  # 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
              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
              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#m
```

```
aking-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.4.2. post-install.yaml

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  ComputeHostnameFormat:
    type: string
    default: ""

resources:
  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      # Do this on CREATE/UPDATE (which is actually the default)
      actions: ['CREATE', 'UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      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
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
```

```
            grep -q "Before=.*network.target" $tuned_service
            if [ ! "$?" -eq 0 ]; then
              grep -q "Before=.*" $tuned_service
              if [ "$?" -eq 0 ]; then
                sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
              else
                sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
              fi
            fi
            systemctl daemon-reload
          fi
        params:
          $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
```

B.4.3. `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-ovs-
dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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': '172.10.2.100', 'end':
'172.10.2.200'}]
  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'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  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: "1,17,9,25"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "2048,2048"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # Datapath type for ovs bridges
  NeutronDatapathType: "netdev"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ##########################
  # Additional settings
  ##########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
  # 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,
ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
  HostCpusList: "0,16,8,24"
```

B.4.4. `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
              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: ExternalInterfaceDefaultRoute}
            -
              type: linux_bond
              name: bond_api
              bonding_options: "mode=active-backup"
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
                  name: nic2
                  primary: true
                -
                  type: interface
                  name: nic3
            -
              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}
            -
              type: ovs_bridge
              name: br-link
              use_dhcp: false
              members:
                -
                  type: interface
                  name: nic4
                -
                  type: vlan
                  vlan_id: {get_param: TenantNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: TenantIpSubnet}


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

B.4.5. `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
  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
  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
  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: nic2
      primary: true
    -
      type: interface
      name: nic3
-
  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-link
  use_dhcp: false
  ovs_extra:
    -
      str_replace:
        template: set port br-link tag=_VLAN_TAG_
        params:
          _VLAN_TAG_: {get_param: TenantNetworkVlanID}
```

```
              addresses:
                -
                  ip_netmask: {get_param: TenantIpSubnet}
              members:
                -
                  type: ovs_dpdk_port
                  name: dpdk0
                  members:
                    -
                      type: interface
                      name: nic4

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

B.4.6. `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 /home/stack/ospd-11-vxlan-dpdk-single-port-ctlplane-bonding/network-
environment.yaml \
--log-file overcloud_install.log
```

B.5. SAMPLE OVS-DPDK AND SR-IOV COMPOSABLE ROLES YAML FILES

B.5.1. `roles_data.yaml`

```
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackup
    - OS::TripleO::Services::CinderScheduler
    - OS::TripleO::Services::CinderVolume
    - OS::TripleO::Services::Congress
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Keystone
    - OS::TripleO::Services::GlanceApi
```

```
    - OS::TripleO::Services::HeatApi
    - OS::TripleO::Services::HeatApiCfn
    - OS::TripleO::Services::HeatApiCloudwatch
    - OS::TripleO::Services::HeatEngine
    - OS::TripleO::Services::MySQL
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronDhcpAgent
    - OS::TripleO::Services::NeutronL3Agent
    - OS::TripleO::Services::NeutronMetadataAgent
    - OS::TripleO::Services::NeutronApi
    - OS::TripleO::Services::NeutronCorePlugin
    - OS::TripleO::Services::NeutronOvsAgent
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::HAproxy
    - OS::TripleO::Services::Keepalived
    - OS::TripleO::Services::Memcached
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::NovaConductor
    - OS::TripleO::Services::MongoDb
    - OS::TripleO::Services::NovaApi
    - OS::TripleO::Services::NovaPlacement
    - OS::TripleO::Services::NovaMetadata
    - OS::TripleO::Services::NovaScheduler
    - OS::TripleO::Services::NovaConsoleauth
    - OS::TripleO::Services::NovaVncProxy
    - OS::TripleO::Services::Ec2Api
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::SwiftProxy
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::CeilometerApi
    - OS::TripleO::Services::CeilometerCollector
    - OS::TripleO::Services::CeilometerExpirer
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::Horizon
    - OS::TripleO::Services::GnocchiApi
    - OS::TripleO::Services::GnocchiMetricd
    - OS::TripleO::Services::GnocchiStatsd
    - OS::TripleO::Services::ManilaApi
    - OS::TripleO::Services::ManilaScheduler
    - OS::TripleO::Services::ManilaBackendGeneric
    - OS::TripleO::Services::ManilaBackendNetapp
    - OS::TripleO::Services::ManilaBackendCephFs
    - OS::TripleO::Services::ManilaShare
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::IronicApi
```

```
    - OS::TripleO::Services::IronicConductor
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::OpenDaylightApi
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::OVNDBs
    - OS::TripleO::Services::NeutronML2FujitsuCfab
    - OS::TripleO::Services::NeutronML2FujitsuFossw
    - OS::TripleO::Services::CinderHPELeftHandISCSI
    - OS::TripleO::Services::Etcd
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::OctaviaApi
    - OS::TripleO::Services::OctaviaHealthManager
    - OS::TripleO::Services::OctaviaHousekeeping
    - OS::TripleO::Services::OctaviaWorker

- name: ComputeSriov
  CountDefault: 1
  HostnameFormatDefault: 'compute-sriov-%index%'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd

- name: ComputeOvsDpdk
  CountDefault: 1
  HostnameFormatDefault: 'compute-ovs-dpdk-%index%'
```

```
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsDpdkAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::Collectd
```

B.5.2. `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: ""
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. Its mandatory parameter.
    type: string
  NeutronDpdkSocketMemory:
    description: Memory allocated for each socket
    default: ""
    type: string
```

```
  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: boot_config}
      - config: {get_resource: set_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  boot_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile an activate it, we
should create FDP repo.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}

  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
nonetwork.service ]; then
                ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service ];
then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

  # 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
              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
              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#m
aking-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.5.3. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  HostCpusList:
```

```
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
    type: string
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. (pmd-cpu-mask)
    type: string
resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE','UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-
cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-
lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

B.5.4. `network.environment.yaml`

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

  OS::TripleO::Services::ComputeNeutronOvsDpdkAgent: /usr/share/openstack-
tripleo-heat-templates/puppet/services/neutron-ovs-dpdk-agent.yaml
  OS::TripleO::Services::NeutronSriovAgent: /usr/share/openstack-tripleo-
heat-templates/puppet/services/neutron-sriov-agent.yaml

  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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': '172.10.2.100', 'end':
'172.10.2.200'}]
  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-link'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'tenant:420:420,tenant:421:421'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  OvercloudComputeSriovFlavor: sriov
  OvercloudComputeOvsDpdkFlavor: dpdk
  # Number of nodes to deploy.
  ControllerCount: 1
  ComputeSriovCount: 1
  ComputeOvsDpdkCount: 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%'
  ComputeSriovHostnameFormat: 'compute-sriov-%index%'
  ComputeOvsDpdkHostnameFormat: 'compute-ovs-dpdk-%index%'
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'

  #########################
  # SR-IOV 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: "ens2f1"
      physical_network: "tenant"
  # List of supported pci vendor devices in the format VendorID:ProductID.
  NeutronSupportedPCIVendorDevs: ['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:ens2f1"
  # 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: "ens2f1:5"

  ############################
  # 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: "1,17,9,25"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "1024,1024"
  # NeutronDpdkDriverType
```

```
  NeutronDpdkDriverType: "vfio-pci"
  # Datapath type for ovs bridges
  NeutronDatapathType: "netdev"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"

  ######################
  # Additional config #
  ######################
  # List of scheduler available filters
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthro
ugh_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','ComputeCapabilities
Filter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGro
upAffinityFilter','PciPassthroughFilter','NUMATopologyFilter']
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
  HostCpusList: "0,16,8,24"
```

B.5.5. `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
  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
```

```
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations)
that will be added to resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: eno1
              use_dhcp: false
              defroute: false
            -
              type: interface
              name: ens1f0
              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: interface
              name: ens1f1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
            -
              type: vlan
              vlan_id: {get_param: InternalApiNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: InternalApiIpSubnet}
            -
```

```
              type: vlan
              vlan_id: {get_param: TenantNetworkVlanID}
              device: ens1f1
              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: ens1f1
              addresses:
              -
                ip_netmask: {get_param: ExternalIpSubnet}
              routes:
                -
                  default: true
                  next_hop: {get_param: ExternalInterfaceDefaultRoute}
            -
              type: ovs_bridge
              name: br-link
              use_dhcp: false
              members:
                -
                  type: interface
                  name: ens2f1


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

B.5.6. `compute-sriov.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
  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
  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: eno1
              use_dhcp: false
              defroute: false
            -
              type: interface
              name: ens1f0
              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: interface
              name: ens1f1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
            -
              type: vlan
              vlan_id: {get_param: InternalApiNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: InternalApiIpSubnet}
```

```
            -
              type: vlan
              vlan_id: {get_param: TenantNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: TenantIpSubnet}
            -
              type: vlan
              vlan_id: {get_param: StorageNetworkVlanID}
              device: ens1f1
              addresses:
                -
                  ip_netmask: {get_param: StorageIpSubnet}
            -
              type: interface
              name: ens2f1
              use_dhcp: false
              defroute: false

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

B.5.7. `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
  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
  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: eno1
  use_dhcp: false
  defroute: false
-
  type: interface
  name: ens1f0
  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: interface
  name: ens1f1
  use_dhcp: false
-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: vlan
  vlan_id: {get_param: TenantNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: ens1f1
  addresses:
    -
      ip_netmask: {get_param: StorageIpSubnet}
-
  type: ovs_user_bridge
  name: br-link
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: ovs_dpdk_port
```

```
                  name: dpdk0
                  members:
                    -
                      type: interface
                      name: ens2f0

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

B.5.8. `overcloud_deploy.sh`

```
#!/bin/bash

openstack overcloud deploy \
--templates \
-r /home/stack/ospd-11-vlan-dpdk-sriov-single-port-composable-roles/roles-
data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
-e /home/stack/ospd-11-vlan-dpdk-sriov-single-port-composable-
roles/network-environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

B.6. SAMPLE VLAN OVS-DPDK YAML FILES WITH MULTIQUEUE AND MTU SETTINGS

B.6.1. `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-ovs-
dpdk.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::TripleO::NodeUserData: first-boot.yaml
  OS::TripleO::NodeExtraConfigPost: post-install.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': '172.10.2.100', 'end':
'172.10.2.200'}]
```

```
  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: 'dpdk_mgmt:br-link0,dpdk_data:br-link1'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'dpdk_mgmt:22:22,dpdk_data:25:28'
  # Nova flavor to use.
  OvercloudControlFlavor: control
  OvercloudComputeFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 1
  ComputeCount: 1
  # NTP server configuration.
  NtpServer: clock.redhat.com
  # Configure the classname of the firewall driver to use for implementing
security groups.
  NeutronOVSFirewallDriver: openvswitch

  # 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: "1,17,9,25"
  # Number of memory channels to be used for DPDK
  NeutronDpdkMemoryChannels: "4"
  # NeutronDpdkSocketMemory
  NeutronDpdkSocketMemory: "2048,2048"
  # NeutronDpdkDriverType
  NeutronDpdkDriverType: "vfio-pci"
  # Datapath type for ovs bridges
  NeutronDatapathType: "netdev"
  # The vhost-user socket directory for OVS
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
  # Set the global MTU
  NeutronGlobalPhysnetMtu: 2000

  ##########################
  # Additional settings
  ##########################
  # Reserved RAM for host processes
  NovaReservedHostMemory: 2048
  # 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:
"2,3,4,5,6,7,18,19,20,21,22,23,10,11,12,13,14,15,26,27,28,29,30,31"
  # 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:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,
ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
  # 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:
"1,2,3,4,5,6,7,9,10,17,18,19,20,21,22,23,11,12,13,14,15,25,26,27,28,29,30,
31"
  # List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
  HostCpusList: "0,16,8,24"
```

B.6.2. `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: ""
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. Its mandatory parameter.
    type: string
  NeutronDpdkSocketMemory:
    description: Memory allocated for each socket
    default: ""
    type: string
  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: boot_config}
      - config: {get_resource: set_ovs_socket_config}
      - config: {get_resource: set_ovs_config}
      - config: {get_resource: set_dpdk_params}
      - config: {get_resource: install_tuned}
      - config: {get_resource: compute_kernel_args}

  boot_config:
    type: OS::Heat::CloudConfig
```

```
    properties:
      cloud_config:
        yum_repos:
          # Overcloud images deployed without any repos.
          # In order to install required tuned profile an activate it, we
should create FDP repo.
          <repo-file-name>:
            name: <repo-name>
            baseurl: <repo-baseurl>
            enabled: 1
            gpgcheck: 0

  set_ovs_socket_config:
    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
              mkdir -p $NEUTRON_VHOSTUSER_SOCKET_DIR
              chown -R qemu:qemu $NEUTRON_VHOSTUSER_SOCKET_DIR
              restorecon $NEUTRON_VHOSTUSER_SOCKET_DIR
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
            $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}

  set_ovs_config:
    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
              if [ -f /usr/lib/systemd/system/openvswitch-
```

```
nonetwork.service ]; then
                ovs_service_path="/usr/lib/systemd/system/openvswitch-
nonetwork.service"
              elif [ -f /usr/lib/systemd/system/ovs-vswitchd.service ];
then
                ovs_service_path="/usr/lib/systemd/system/ovs-
vswitchd.service"
              fi
              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
            fi
          params:
            $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

  # 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
              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
              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#m
aking-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.6.3. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json
  HostCpusList:
    description: >
      List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
    type: string
  NeutronDpdkCoreList:
    description: >
      List of logical cores for PMD threads. (pmd-cpu-mask)
    type: string
resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE','UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            function tuned_service_dependency() {
              tuned_service=/usr/lib/systemd/system/tuned.service
              grep -q "network.target" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i '/After=.*/s/network.target//g' $tuned_service
              fi
              grep -q "Before=.*network.target" $tuned_service
```

```
              if [ ! "$?" -eq 0 ]; then
                  grep -q "Before=.*" $tuned_service
                  if [ "$?" -eq 0 ]; then
                      sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
                  else
                      sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
                  fi
              fi
            }

            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"
            }

            if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
neutron_ovs_dpdk_agent; then
                pmd_cpu_mask=$( get_mask $PMD_CORES )
                host_cpu_mask=$( get_mask $LCORE_LIST )
                ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-
cpu-mask=$pmd_cpu_mask
                ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-
lcore-mask=$host_cpu_mask
                tuned_service_dependency
            fi
```

```
          params:
            $LCORE_LIST: {get_param: HostCpusList}
            $PMD_CORES: {get_param: NeutronDpdkCoreList}
```

B.6.4. `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
  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
              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: ExternalInterfaceDefaultRoute}
            -
              type: linux_bond
```

```
      name: bond_api
      bonding_options: "mode=active-backup"
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      members:
        -
          type: interface
          name: nic2
          primary: true
        -
          type: interface
          name: nic3
    -
      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
      dns_servers: {get_param: DnsServers}
```

```
              members:
                -
                  type: interface
                  name: nic4
                  mtu: 2000
            -
              type: ovs_bridge
              name: br-link1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: interface
                  name: nic5
                  mtu: 2000


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

B.6.5. `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
  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
  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
      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: nic2
          primary: true
        -
          type: interface
          name: nic3
    -
      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
      dns_servers: {get_param: DnsServers}
      members:
```

```
                -
                  type: ovs_dpdk_port
                  name: dpdk0
                  mtu: 2000
                  ovs_extra:
                  - set interface $DEVICE options:n_rxq=2
                  - set interface $DEVICE mtu_request=$MTU
                  members:
                    -
                      type: interface
                      name: nic4
            -
              type: ovs_user_bridge
              name: br-link1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              members:
                -
                  type: ovs_dpdk_port
                  name: dpdk1
                  mtu: 2000
                  ovs_extra:
                  - set interface $DEVICE options:n_rxq=2
                  - set interface $DEVICE mtu_request=$MTU
                  members:
                    -
                      type: interface
                      name: nic5

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

B.6.6. `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 /home/stack/ospd-11-vlan-dpdk-two-ports-ctlplane-bonding-
multiqueue/network-environment.yaml \
--log-file overcloud_install.log
```