



# Red Hat OpenStack Platform 11

## ネットワーク機能仮想化 (NFV) の設定ガイド

ネットワーク機能仮想化 (NFV) の OpenStack デプロイメント設定



# Red Hat OpenStack Platform 11 ネットワーク機能仮想化 (NFV) の設定ガイド

---

ネットワーク機能仮想化 (NFV) の OpenStack デプロイメント設定

OpenStack Team  
rhos-docs@redhat.com

## 法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 概要

本ガイドでは、Red Hat OpenStack Platform 11 の NFV デプロイメントで SR-IOV と OVS-DPDK を設定する手順を説明します。

## 目次

第1章 前書き .....	4
第2章 概要 .....	5
第3章 仮想ネットワークの SR-IOV サポートの設定 .....	7
3.1. VLAN トンネリングを使用する単一 NIC の SR-IOV コンポーザブルロールの設定	7
3.1.1. SR-IOV 用のコンポーザブルロールを作成するための roles_data.yaml の編集	7
3.1.2. first-boot.yaml の編集	8
3.1.3. post-install.yaml の編集	9
3.1.4. network-environment.yaml の編集	10
3.1.5. controller.yaml の編集	12
3.1.6. compute.yaml の編集	13
3.1.7. overcloud_deploy.sh スクリプトを実行します。	14
3.2. SR-IOV 用のフレーバーの作成とインスタンスのデプロイ	14
第4章 仮想ネットワーク向けの OVS-DPDK サポートの設定 .....	16
4.1. OVS-DPDK の設定パラメーターについての理解	17
4.2. VLAN トンネリングを使用した単一ポートの OVS-DPDK の設定	17
4.2.1. first-boot.yaml の編集	17
4.2.2. post-install.yaml の編集	22
4.2.3. network-environment.yaml の編集	23
4.2.4. controller.yaml の編集	25
4.2.5. compute.yaml の編集	27
4.2.6. overcloud_deploy.sh スクリプトの実行	28
4.3. VLAN トンネリングを使用した 2 ポートの OVS-DPDK の設定	28
4.3.1. first-boot.yaml の編集	28
4.3.2. post-install.yaml の編集	33
4.3.3. network-environment.yaml の編集	34
4.3.4. controller.yaml の編集	36
4.3.5. compute.yaml の編集	38
4.3.6. overcloud_deploy.sh スクリプトを実行します。	39
4.4. VLAN トンネリングを使用した 2 ポートの OVS-DPDK データプレーンボンディングの設定	40
4.4.1. first-boot.yaml の編集	40
4.4.2. post-install.yaml の編集	44
4.4.3. network-environment.yaml の編集	45
4.4.4. controller.yaml の編集	48
4.4.5. compute.yaml の編集	49
4.4.6. overcloud_deploy.sh スクリプトを実行します。	50
4.5. VXLAN トンネリングを使用した単一ポートの OVS-DPDK の設定	51
4.5.1. first-boot.yaml の編集	51
4.5.2. post-install.yaml の編集	54
4.5.3. network-environment.yaml の編集	55
4.5.4. controller.yaml の編集	57
4.5.5. compute.yaml の編集	59
4.5.6. overcloud_deploy.sh スクリプトを実行します。	60
4.6. OVS-DPDK コンポーザブルロールの設定	60
4.6.1. コンポーザブルロールを作成するための roles_data.yaml の編集	61
4.6.2. 新規コンポーザブルロール向けの network-environment.yaml の編集	62
4.6.3. overcloud_deploy.sh スクリプトを実行します。	62
4.7. 既知の制限事項	63
4.8. フレーバーの作成と OVS-DPDK 用インスタンスのデプロイ	63
4.9. 設定のトラブルシューティング	64

<b>第5章 詳細情報</b>	<b>67</b>
<b>付録A SR-IOV YAML ファイルのサンプル</b>	<b>68</b>
A.1. SR-IOV コンポーザブルロールの YAML ファイルのサンプル	68
A.1.1. roles_data.yaml	68
A.1.2. first-boot.yaml	70
A.1.3. post-install.yaml	72
A.1.4. network.environment.yaml	74
A.1.5. controller.yaml	76
A.1.6. compute.yaml	79
A.1.7. overcloud_deploy.sh	82
<b>付録B OVS-DPDK YAML ファイルのサンプル</b>	<b>83</b>
B.1. VLAN OVS-DPDK YAML ファイルのサンプル	83
B.1.1. first-boot.yaml	83
B.1.2. post-install.yaml	88
B.1.3. network.environment.yaml	89
B.1.4. controller.yaml	91
B.1.5. compute.yaml	95
B.1.6. overcloud_deploy.sh	97
B.2. 2 ポートの VLAN OVS-DPDK YAML ファイルのサンプル	97
B.2.1. first-boot.yaml	97
B.2.2. post-install.yaml	103
B.2.3. network.environment.yaml	104
B.2.4. controller.yaml	106
B.2.5. compute.yaml	109
B.2.6. overcloud_deploy.sh	112
B.3. VLAN OVS-DPDK データプレーンボンディング YAML ファイルのサンプル	113
B.3.1. first-boot.yaml	113
B.3.2. post-install.yaml	118
B.3.3. network.environment.yaml	120
B.3.4. controller.yaml	122
B.3.5. compute.yaml	124
B.3.6. overcloud_deploy.sh	127
B.4. VXLAN OVS-DPDK データプレーンボンディング YAML ファイルのサンプル	127
B.4.1. first-boot.yaml	127
B.4.2. post-install.yaml	133
B.4.3. network.environment.yaml	134
B.4.4. controller.yaml	136
B.4.5. compute.yaml	139
B.4.6. overcloud_deploy.sh	141
B.5. OVS-DPDK および SR-IOV コンポーザブルロールの YAML ファイルのサンプル	142
B.5.1. roles_data.yaml	142
B.5.2. first-boot.yaml	144
B.5.3. post-install.yaml	150
B.5.4. network.environment.yaml	151
B.5.5. controller.yaml	154
B.5.6. compute.yaml	157
B.5.7. overcloud_deploy.sh	160



## 第1章 前書き

Red Hat OpenStack Platform は、Red Hat Enterprise Linux 上にプライベートまたはパブリックの Infrastructure-as-a-Service (IaaS) クラウドを構築するための基盤を提供します。これにより、スケーラビリティが極めて高く、耐障害性に優れたプラットフォームをクラウド対応のワークロード開発にご利用いただくことができます。

本ガイドでは、NFV デプロイメント向けに Red Hat OpenStack Platform 11 director を使用して SR-IOV および DPDK-accelerated Open vSwitch (OVS) を設定する手順を説明します。



## 第2章 概要

ネットワーク機能仮想化 (NFV) とは、汎用のクラウドベースインフラストラクチャー上でネットワーク機能を仮想化するソフトウェアベースのソリューションです。コストの削減とイノベーションの拡大を図りつつ、敏捷性、柔軟性、容易性、効率性、スケーラビリティを提供します。NFV を活用することで、通信事業者 (CSP) は従来のハードウェアから離れることができます。



### 注記

本ガイドでは、単一ポートのみの SR-IOV および OVS-DPDK を設定する手順について説明します。

Red Hat OpenStack Platform 11 director を使用すると、オーバークラウドのネットワークを分離することができます。この機能では、特定のネットワーク種別 (例: 外部、テナント、内部 API など) を分離ネットワークに分けることができます。ネットワークは、単一ネットワークインターフェース上または複数のネットワークインターフェースに分散してデプロイすることが可能です。Open vSwitch では、複数のインターフェースを単一のブリッジに割り当ててボンディングを作成することができます。Red Hat OpenStack Platform 11 のインストールでは、ネットワークの分離はテンプレートファイルを使用して設定されます。テンプレートファイルを指定しない場合には、サービスネットワークはすべてプロビジョニングネットワーク上にデプロイされます。テンプレートの設定ファイルは 2 種類あります。

- **network-environment.yaml**: このファイルには、オーバークラウドノードのネットワーク設定で使用するサブネット、IP アドレス範囲などのネットワークの情報が含まれます。さらに、このファイルには、さまざまなシナリオで使用できるように、デフォルトのパラメーターの値を上書きする異なる設定も含まれます。
- ホストのテンプレート (例: **compute.yaml**、**controller.yaml** など): これらのファイルはオーバークラウドノードのネットワークインターフェース設定を定義します。ネットワーク情報の値は、**network-environment.yaml** ファイルから抽出されます。
- インストール後の設定用のファイル (**first-boot.yaml**、**post-install.yaml**): これらのファイルは、インストール後に実行するさまざまな設定のステップを提供します。以下に例を示します。
  - Grub 引数の設定
  - DPDK の設定
  - Tuned のインストールと設定。**tuned** パッケージには、システムコンポーネントを監視して、そのモニタリング情報に基づいてシステムの設定を動的にチューニングする **tuned** デーモンが含まれています。OVS-DPDK および SR-IOV のデプロイメントで、適切な CPU アフィニティの設定を指定するには、**tuned-cpu-partitioning** プロファイルを使用すべきです。



### 注記

**tuned** のプロファイルをインストールしてアクティブ化するには、**first-boot.yaml** ファイルで **yum** リポジトリを指定する必要があります。**tuna** パッケージは、**tuned-cpu-partitioning** プロファイルのインストール時に依存関係としてインストールします。

**first-boot.yaml** ファイル内にベース URL を定義する必要があります。リポジトリに必要なベース URL を取得するには、以下のコマンドを実行します。

```
"TUNA_REPO=`yum info tuna | grep Repo | awk '{print $3}' | sed
's#/.*##g'` && yum repolist -v $TUNA_REPO | grep Repo-baseurl &&
TUNED_PROFILE_REPO=`yum info tuned-profiles-cpu-partitioning |
grep Repo | awk '{print $3}' | sed 's#/.*##g'` && yum repolist -v
$TUNED_PROFILE_REPO | grep Repo-baseurl"
```

**first-boot.yaml** ファイルでベース URL を定義する必要があります。

**network-environment.yaml** とホストのテンプレートファイルはいずれもアンダークラウドノードの **/usr/share/openstack-tripleo-heat-templates/** にあります。

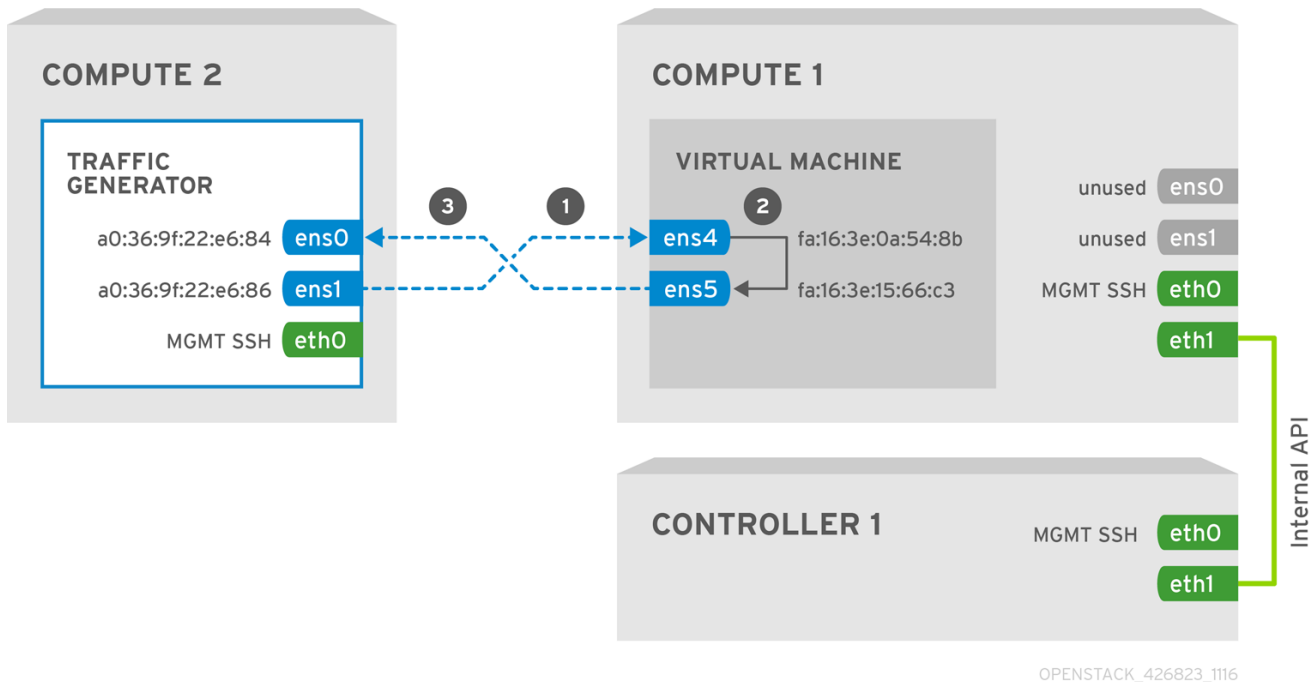
前述した参照ファイルのサンプルは [「SR-IOV YAML ファイルのサンプル」](#) を参照してください。

以下の項では、Red Hat OpenStack Platform director を使用して、NFV 向けの SR-IOV および OVS-DPDK 機能を使用できるように **network-environment.yaml**、ホストのテンプレートファイル、インストール後の設定用ファイルを設定する方法をさらに詳しく説明しています。

## 第3章 仮想ネットワークの SR-IOV サポートの設定

本章では、director を使用した Red Hat OpenStack Platform 11 環境への Single Root Input/Output Virtualization (SR-IOV) の設定について説明します。

以下の手順では、**network-environment.yaml** ファイルを更新して、カーネルの引数、SR-IOV ドライバー、PCI パススルーなどのパラメーターを追加します。また、**compute.yaml** ファイルも更新して SR-IOV インターフェースのパラメーターを追加してから、**overcloud\_deploy.sh** スクリプトを実行して、その SR-IOV パラメーターを使用してオーバークラウドをデプロイする必要があります。



### 3.1. VLAN トンネリングを使用する単一 NIC の SR-IOV コンポーザブルロールの設定

本項では、OpenStack 環境向けに VLAN トンネリングを使用する SR-IOV 用のコンポーザブルロールを設定する方法について説明します。コンポーザブルロールの作成およびデプロイのプロセスには、以下の作業が含まれます。

- **role\_data.yaml** ファイルのローカルコピーで新規ロールを定義します。
- **network\_environment.yaml** ファイルを編集して、この新規ロールを追加します。
- 更新したロールセットでオーバークラウドをデプロイします。

以下の例では、**ComputeSriov** は、コンピュートノード用のコンポーザブルロールで、サポートされている NIC が搭載されたノードでのみ SR-IOV を有効にします。Red Hat OpenStack Platform によって提供される既存のデフォルトロールセットは、**/home/stack/roles\_data.yaml** ファイルに保管されます。

#### 3.1.1. SR-IOV 用のコンポーザブルロールを作成するための **roles\_data.yaml** の編集

**roles\_data.yaml** ファイルを **/home/stack/templates** ディレクトリーにコピーして、新しい **ComputeSriov** ロールを追加します。

```
- 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

```

### 3.1.2. **first-boot.yaml** の編集

1. **tuned** の設定で CPU アフィニティを指定するように設定します。

```

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
            # Install the tuned package
            yum install -y tuned-profiles-cpu-partitioning

            tuned_conf_path="/etc/tuned/cpu-partitioning-
variables.conf"

```

```

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

        sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/g'
-i /etc/default/grub ;
grub2-mkconfig -o /etc/grub2.cfg

reboot
fi

```

2. 初期ブートパラメーターを設定します。

```

params:
  $KERNEL_ARGS: {get_param: ComputeKernelArgs}
  $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
  $TUNED_CORES: {get_param: HostIsolatedCoreList}

```

### 3.1.3. `post-install.yaml` の編集

1. `tuned` の設定で CPU アフィニティを指定するように設定します。

```

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
          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}

```

### 3.1.4. network-environment.yaml の編集

1. **ComputeSriov** ロールを含む、SR-IOV 用のカスタムリソースを **resource\_registry** 下に追加します。

```

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::Compute::Net::SoftwareConfig: nic-
configs/compute.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. **parameter\_defaults** の下でトンネルの種別を無効にして (値を "" に設定)、ネットワーク種別を **vlan** に設定します。

```

NeutronTunnelTypes: ""
NeutronNetworkType: 'vlan'

```

3. **parameter\_defaults** の下で、Open vSwitch の物理ネットワークをブリッジマッピングに設定します。

```

NeutronBridgeMappings: 'tenant:br-link'

```

4. **parameter\_defaults** の下で、OpenStack Networking ML2 および Open vSwitch VLAN のマッピング範囲を指定します。

```

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

```

この例では、物理ネットワーク (**tenant**) 上の VLAN 範囲を設定しています

5. **parameter\_defaults** の下で、SR-IOV の設定パラメーターを指定します。

- a. SR-IOV メカニズムドライバー (**sriovnicswitch**) を有効化します。

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

- b. Compute の **pci\_passthrough\_whitelist** パラメーターを設定し、SR-IOV インターフェース用の **devname** を指定します。ホワイトリストにより、仮想マシンが利用可能な PCI デバイスが設定されます。

```
NovaPCIPassthrough:
  - devname: "ens2f1"
    physical_network: "tenant"
```

- c. 物理ネットワークと SR-IOV インターフェースを **PHYSICAL\_NETWORK:PHYSICAL DEVICE** の形式で指定します。

サーバー上にある **network\_vlan\_ranges** に表示されている物理ネットワークにはすべて、各エージェントの適切なインターフェースへのマッピングが必要です。

```
NeutronPhysicalDevMappings: "tenant:ens2f1"
```

この例では、**physical\_network** 名に **tenant** を使用しています。

- d. 各 SR-IOV インターフェースに確保する Virtual Function (VF) の数を指定します。

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

この例では、SR-IOV インターフェース用に 5 つの VF を確保しています。



#### 注記

NFV を実装した Red Hat OpenStack Platform で現在サポートされている VF 数は 30 以下となっています。

6. **parameter\_defaults** の下で、ホストのプロセス用のメモリーを確保します。

```
NovaReservedHostMemory: "2048"
```

7. **parameter\_defaults** の下で、仮想マシンのプロセス用に確保する物理 CPU コアのコンマ区切りリストまたは範囲を指定します。

```
NovaVcpuPinSet: "4,20,5,21,6,22,7,23"
```

8. **parameter\_defaults** の下で、最も制約の多いフィルターからリストして、ノードのフィルタリングプロセスをさらに効率化します。

コンピュートは、フィルターの配列を使用してノードをフィルタリングします。これらのフィルターは、リスト順に適用されます。

```
NovaSchedulerDefaultFilters:
['AvailabilityZoneFilter', 'RamFilter', 'ComputeFilter', 'ComputeCapabi
```

```
litiesFilter', 'ImagePropertiesFilter', 'ServerGroupAntiAffinityFilter',
'ServerGroupAffinityFilter', 'PciPassthroughFilter', 'NUMATopologyFilter']
```

9. **parameter\_defaults** の下で、**ComputeKernelArgs** パラメーターを追加して、初期ブート時にそれらのパラメーターがデフォルトの **grub** ファイルに追加されるようにします。

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



#### 注記

DPDK インスタンスに関連付けられたフレーバーには **hw:mem\_page\_size=1GB** を追加する必要があります。このように設定しなかった場合には、そのインスタンスには DHCP による割り当ては行われません。

10. **parameter\_defaults** の下に、チューニングする物理 CPU コアの一覧または範囲を指定します。  
指定の引数は、tuned **cpu-partitioning** プロファイルに追加されます。

```
HostIsolatedCoreList: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"
```

### 3.1.5. **controller.yaml** の編集

1. 分離されたネットワーク用のインターフェースを作成します。

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

2. このインターフェースに VLAN を割り当てます。

```
-
  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. Compute ノードに接続する OVS ブリッジを作成します。

```

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

```

### 3.1.6. compute.yaml の編集

デフォルトの compute.yaml ファイルから **compute-sriov.yaml1** を作成します。これは、**ComputeSriov** コンポーザブルロールを使用するコンピュータノードのパラメーターを制御するファイルです。

1. 分離されたネットワーク用のインターフェースを作成します。

```

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

```

2. このインターフェースに VLAN を割り当てます。

```

-
  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}
-

```

### 3.1.7. `overcloud_deploy.sh` スクリプトを実行します。

以下の例では、コンポーザブルロールを使用する **openstack overcloud deploy** の Bash スクリプトを定義しています。

```

# #!/bin/bash

openstack overcloud deploy --templates \
-r /home/stack/templates/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` は、更新された **roles\_data.yaml** ファイルの場所です。このファイルが SR-IOV のコンポーザブルロールを定義します。

## 3.2. SR-IOV 用のフレーバーの作成とインスタンスのデプロイ

NFV を実装する Red Hat OpenStack Platform デプロイメントの SR-IOV の設定を完了した後は、以下の手順に従ってフレーバーを作成してインスタンスをデプロイします。

1. アグリゲートグループを作成して、SR-IOV 用にホストを追加します。

```

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

```

2. フレーバーを作成します。

```

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

```

このコマンドでは、**m1.medium\_huge\_4cpu** はフレーバー名、**4096** は MB 単位のメモリー容量、**150** は GB 単位のディスク容量 (デフォルトでは 0 G)、**4** は仮想 CPU 数を設定しています。

### 3. フレーバーの追加のプロパティを設定します。

```
# openstack flavor set --property hw:cpu_policy=dedicated --  
property hw:mem_page_size=large --property hw:numa_nodes=1 --  
property hw:numa_mempolicy=preferred --property  
hw:numa_cpus.0=0,1,2,3 --property hw:numa_mem.0=4096  
m1.medium_huge_4cpu
```

このコマンドでは、**m1.medium\_huge\_4cpu** はフレーバー名を指定しており、残りはそのフレーバーのその他のプロパティを設定しています。

### 4. インスタンスをデプロイします。

```
# openstack server create --flavor m1.medium_huge_4cpu --  
availability-zone sriov --image rhel_7.3 --nic port-id=<direct-  
port-id> sriov_vm
```

このコマンドでは、**m1.medium\_huge\_4cpu** はフレーバーの名前または ID、**sriov** はサーバーのアベイラビリティゾーン、**rhel\_7.3** はインスタンスの作成に使用するイメージ (名前または ID)、**<direct-port-id>** はサーバー上の NIC、**sriov\_vm** はインスタンス名を設定します。

これで、NFV ユースケースの SR-IOV 向けインスタンスのデプロイが完了しました。

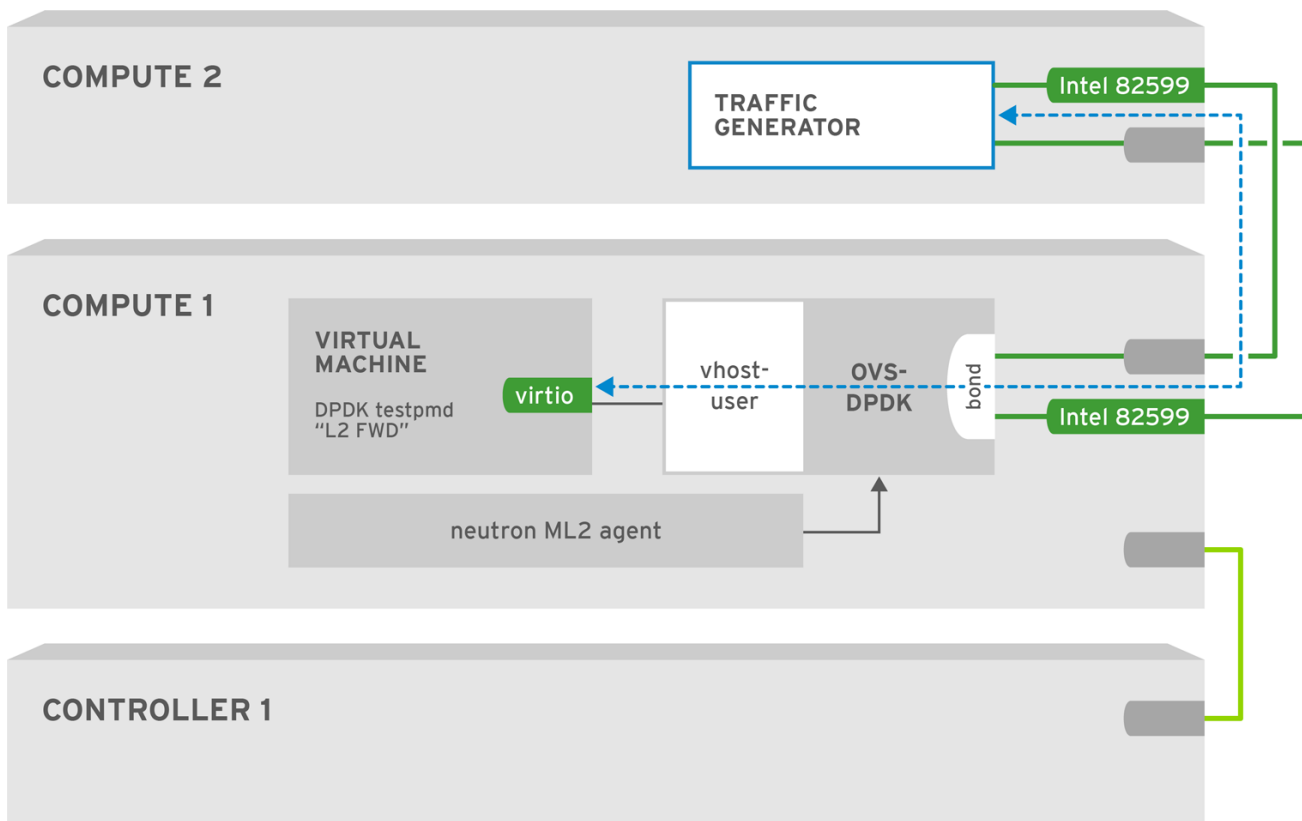
## 第4章 仮想ネットワーク向けの OVS-DPDK サポートの設定

本章では、コンポーザブルロールを使用して、DPDK (OVS-DPDK) を Open vSwitch とともに Red Hat OpenStack Platform 環境内にデプロイします。オーバークラウドは、通常コントローラーノードやコンピュートノードなどの事前定義済みロールのノードと、異なる種別のストレージノードで構成されます。これらのデフォルトロールにはそれぞれ、director ノード上のコア Heat テンプレートコレクションで定義されている一式のサービスが含まれます。

Red Hat OpenStack Platform 11 では、コンポーザブルロール機能を使用し、各ロールからサービスを追加/削除してカスタムのデプロイメントロールを作成できます。コンポーザブルロールの詳細情報は「[コンポーザブルサービスとカスタムロール](#)」を参照してください。

OVS-DPDK の設定は、以下の作業で構成されます。

- コンポーザブルロールを使用する場合には、**roles-data.yaml** ファイルをコピーして編集し、OVS-DPDK 用のコンポーザブルロールを追加します。
- 適切な **network-environment.yaml** ファイルを更新して、カーネル引数と DPDK 引数のパラメーターを追加します。
- **compute.yaml** ファイルを更新して、DPDK インターフェース用のブリッジを追加します。
- **controller.yaml** ファイルを更新して、DPDK インターフェースパラメーター用の同じブリッジ情報を追加します。
- **overcloud\_deploy.sh** スクリプトを実行して、DPDK パラメーターを使用してオーバークラウドをデプロイします。

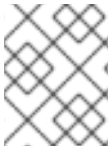


OPENSTACK\_426823\_1116

手順を開始する前に、以下の項目が揃っていることを確認します。

- Red Hat Enterprise Linux 7.3 をベースとした Red Hat OpenStack Platform 11

- OVS-DPDK 2.6
- NIC: Dual port Intel x520



### 注記

OVS-DPDK は Red Hat Enterprise Linux に依存しており、Red Hat Enterprise Linux 7.3 では OVS バージョン 2.6.\* が必要です。

## 4.1. OVS-DPDK の設定パラメーターについての理解

本項では、OVS-DPDK 向けの OpenStack ネットワークを最適化するために network-environment.yaml ファイルで設定する一般的な OVS-DPDK パラメーターについて説明します。

- NeutronDpdkMemoryChannels: メモリーチャンネルを CPU にマッピングします。
- NeutronDpdkCoreList: DPDK PMD に使用される CPU コアを指定します。
- NeutronDpdkSocketMemory: NUMA ノードごとに、ヒュージページプールから事前に割り当てられるメモリー容量を指定します。たとえば、DPDK Poll Mode Driver (PMD) (NeutronDpdkCoreList) が 2 つの NUMA ノードを使用する場合には、NeutronDpdkSocketMemory の設定は **1024,1024** となります。
- NovaReservedHostMemory: ホスト上のタスク用のメモリーを確保します (例: **4096**)。
- NovaVcpuPinSet: CPU ピニングのコアを設定します。このコアの範囲は、NeutronDpdkCoreList コアとは重複しないようにする必要があります。
- hugepagesz: CPU 上のヒュージページのサイズ。この値は、CPU ハードウェアによって異なる場合があります。
- hugepages count: 利用可能なヒュージページの数。この値は、ホストのメモリーの空き容量によって異なります。コンピュートノードに関連付けられている OpenStack フレーバー内のヒュージページ数の値を設定する必要もあります。
- Tuned CPU partitioning プロファイル: DPDK のパフォーマンスを設定するために適切な tuned プロファイルを設定します。

## 4.2. VLAN トンネリングを使用した単一ポートの OVS-DPDK の設定

本項では、OpenStack 環境で、単一のデータプレーンポートと 2 つのコントロールプレーンポート (1 つの Linux ボンディング) を使用する OVS-DPDK を設定およびデプロイする手順について説明します。

### 4.2.1. first-boot.yaml の編集

first-boot.yaml ファイルを編集して、OVS と DPDK のパラメーターを設定し、tuned を CPU アフィニティー向けに構成します。

1. 追加のリソースを加えます。

```
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. NeutronVhostUserSocketDir の設定を決定します。

```

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. OVS の構成を設定します。

```

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. DPDK のパラメーターを設定します。

```

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=$((1<<core))
    bm[$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 $TUNED_CORES )
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dppk-init=true
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dppk-socket-mem=$SOCKET_MEMORY
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:pmd-cpu-mask=$pmd_cpu_mask
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dppk-lcore-mask=$host_cpu_mask
fi
params:
    $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}

```



```

$TUNED_CORES: {get_param: HostCpusList}
$PMD_CORES: {get_param: NeutronDpdkCoreList}
$SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}

```

5. **tuned** の設定で CPU アフィニティを指定するように設定します。

```

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

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

```

6. カーネル引数を設定します。

```

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"/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}

```

#### 4.2.2. `post-install.yaml` の編集

1. **tuned** の設定で CPU アフィニティを指定するように設定します。

```

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}

```

### 4.2.3. network-environment.yaml の編集

1. **resource\_registry** の下に OVS-DPDK 用のカスタムリソースを追加します。

```

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. **parameter\_defaults** の下でトンネルの種別を無効にして (値を "" に設定)、ネットワーク種別を **vlan** に設定します。

```

NeutronTunnelTypes: ""
NeutronNetworkType: 'vlan'

```

3. **parameter\_defaults** の下で、物理ネットワークを仮想ブリッジにマッピングします。

```

NeutronBridgeMappings: 'dpdk_data:br-link0'

```

4. **parameter\_defaults** の下で、OpenStack Networking ML2 および Open vSwitch VLAN のマッピング範囲を指定します。

```

NeutronNetworkVLANRanges: 'dpdk_data:22:22'

```

この例では、物理ネットワーク (**dpdk\_data**) 上の VLAN 範囲を設定しています。

5. **parameter\_defaults** の下で、OVS-DPDK の設定パラメーターを指定します。



#### 注記

**NeutronDPDKCoreList** および **NeutronDPDKMemoryChannels** はこの手順では **必須** の設定です。適切な値を指定せずに DPDK のデプロイを試みると、デプロイメントが失敗するか、不安定なデプロイメントとなってしまいます。

- a. **[allowed\_pattern: "'[0-9, -]+'"]** の形式で DPDK Poll Mode Driver (PMD) として使用可能なコアの一覧を指定します。

```

NeutronDpdkCoreList: "'4,6,20,22'"

```

■

OVS-DPDK のパフォーマンスを最適化するには、以下のオプションを検討してください。

- DPDK インターフェースの NUMA ノードに関連付けられた CPU を選択します。**cat /sys/class/net/<interface>/device/numa\_node** を使用してインターフェースに関連付けられた NUMA ノードをリストし、**lscpu** を使用してその NUMA ノードに関連付けられた CPU をリストします。
- CPU のシブリングをグループにまとめます (ハイパースレッディングの場合)。CPU のシブリングを特定するには、**cat /sys/devices/system/cpu/<cpu>/topology/thread\_siblings\_list** コマンドを実行します。
- CPU 0 はホストのプロセス用に確保します。
- PMD に割り当てられた CPU を分離して、ホストのプロセスがそれらの CPU を使用しないようにします。
- **NovaVcpuPinset** を使用して、PDM に割り当てられている CPU を Compute のスケジューリングから除外します。

b. **[allowed\_pattern: "[0-9]+"]** の形式でメモリーチャンネル数を指定します。

```
NeutronDpdkMemoryChannels: "4"
```

c. 各ソケットに対してヒュージページプールから事前割り当てされるメモリーを設定します。

```
NeutronDpdkSocketMemory: "'2048,2048'"
```

これは、CPU ソケットを昇順で指定するコンマ区切りの文字列です。この例では、2 NUMA ノード構成を前提としており、ヒュージページからソケット 0 に 2048 MB、ソケット 1 に 2048 MB を事前に割り当てるように設定しています。単一 NUMA ノード構成のシステムの場合には、この値を **1024,0** に設定してください。

d. OVS ブリッジの DPDK ドライバーの種別とデータパスの種別を設定します。

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

6. **parameter\_defaults** の下で、OVS 用の vhost-user ソケットディレクトリーを指定します。

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

7. **parameter\_defaults** の下で、ホストのプロセス用のメモリーを確保します。

```
NovaReservedHostMemory: "2048"
```

8. **parameter\_defaults** の下で、仮想マシンのプロセス用に確保する物理 CPU コアのコンマ区切りリストまたは範囲を指定します。

```
NovaVcpuPinSet: "8,10,12,14,18,24,26,28,30"
```

9. **parameter\_defaults** の下で、最も制約の多いフィルターからリストして、ノードのフィルタリングプロセスをさらに効率化します。  
 コンピュートは、フィルターの配列を使用してノードをフィルタリングします。これらのフィルターは、リスト順に適用されます。

```
NovaSchedulerDefaultFilters:
  "RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

10. **parameter\_defaults** の下で、**ComputeKernelArgs** パラメーターを追加して、初期ブート時にそれらのパラメーターがデフォルトの **grub** ファイルに追加されるようにします。

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



### 注記

これらのヒュージページは、仮想マシンにより消費されます。また、本手順に記載の **NeutronDpdkSocketMemory** パラメーターを使用すると OVS-DPDK により消費されます。仮想マシンが利用可能なヒュージページは、**boot** パラメーターから **NeutronDpdkSocketMemory** を減算した値です。

DPDK に関連付けられたフレーバーに **hw:mem\_page\_size=1GB** を追加する必要があります。追加しなかった場合には、そのインスタンスは DHCP から割り当てを受けなくなります。

11. **parameter\_defaults** の下に、チューニングする物理 CPU コアの一覧または範囲を指定します。  
 指定の引数は、tuned **cpu-partitioning** プロファイルに追加されます。

```
HostIsolatedCpusList: "'2,4,6,8,10,12,14,18,20,22,24,26,28,30'"
```

12. **parameter\_defaults** の下で、OVS-DPDK プロセスが **dpdk-lcore-mask** に使用する論理コアの一覧を設定します。これらのコアは、**NeutronDpdkCoreList** および **NovaVcpuPinSet** のコアリストとは相互に排他的である必要があります。

```
HostCpusList: "'0,1,2,3,5,17,18,19,21'"
```

#### 4.2.4. controller.yaml の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```
-
  type: linux_bond
  name: bond_api
  members:
    -
      type: interface
      name: nic3
    -
      type: interface
      name: nic4
```

## 2. この Linux ボンディングに VLAN を割り当てます。

```

-
  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. Compute ノードに接続する OVS ブリッジを作成します。

```

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

```

### 4.2.5. compute.yaml の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```
-
  type: linux_bond
  name: bond_api
  members:
    -
      type: interface
      name: nic3
    -
      type: interface
      name: nic4
```

2. この Linux ボンディングに VLAN を割り当てます。

```
-
  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. コントローラーにリンクするための DPDK ポートを使用したブリッジを設定します。

```
-
  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: nic5
```

**注記**

複数の DPDK デバイスが含まれるようにするには、追加する各 DPDK デバイスごとに **type** コードスニペットを繰り返してください。

**注記**

OVS-DPDK を使用する場合には、**すべての** コンピュートノードのブリッジが **ovs\_user\_bridge** の種別である必要があります。director は設定を受け入れることができますが、Red Hat OpenStack Platform は **ovs\_bridge** と **ovs\_user\_bridge** の混在する構成はサポートしていません。

#### 4.2.6. **overcloud\_deploy.sh** スクリプトの実行

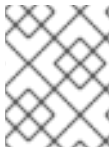
以下の例では、Bash スクリプト内で、OVS-DPDK 環境向けの **openstack overcloud deploy** コマンドを定義しています。

```
#!/bin/bash

openstack overcloud deploy --templates \
-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
```

ここで

- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml** は、Compute ロールの OVS-DPDK パラメーターを有効にするデフォルトの **neutron-ovs-dpdk.yaml** ファイルの場所です。
- **/home/stack/<relative-directory>/network-environment.yaml** は、**network-environment.yaml** ファイルのパスです。このファイルを使用して、**neutron-ovs-dpdk.yaml** ファイルからのデフォルト値を上書きします。

**注記**

OVS-DPDK はセキュリティーグループおよびライブマイグレーションはサポートしていません。

### 4.3. VLAN トンネリングを使用した 2 ポートの OVS-DPDK の設定

本項では、OpenStack 環境にコントロールプレーンの Linux ボンディングを使用する 2 データプレーンポートの OVS-DPDK を設定およびデプロイする手順について説明します。

#### 4.3.1. **first-boot.yaml** の編集

**first-boot.yaml** ファイルを編集して、OVS と DPDK のパラメーターを設定し、**tuned** を CPU アフィニティー向けに構成します。

1. 追加のリソースを加えます。

```
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. NeutronVhostUserSocketDir の設定を決定します。

```

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. OVS の構成を設定します。

```

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. DPDKのパラメーターを設定します。

```

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))
        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 $TUNED_CORES )
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-init=true
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-socket-mem=$SOCKET_MEMORY
    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}
    $TUNED_CORES: {get_param: HostCpusList}
    $PMD_CORES: {get_param: NeutronDpdkCoreList}
    $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}

```

5. **tuned** の設定で CPU アフィニティーを指定するように設定します。

```

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

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

```

6. カーネル引数を設定します。

```

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"/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}

```

### 4.3.2. `post-install.yaml` の編集

1. `tuned` の設定で CPU アフィニティを指定するように設定します。

```

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}

```

### 4.3.3. network-environment.yaml の編集

1. **resource\_registry** の下に OVS-DPDK 用のカスタムリソースを追加します。

```

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. **parameter\_defaults** の下でトンネルの種別を無効にして (値を "" に設定)、ネットワーク種別を **vlan** に設定します。

```

NeutronTunnelTypes: ""
NeutronNetworkType: 'vlan'

```

3. **parameter\_defaults** の下で、物理ネットワークを仮想ブリッジにマッピングします。

```

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

```

4. **parameter\_defaults** の下で、OpenStack Networking ML2 および Open vSwitch VLAN のマッピング範囲を指定します。

```

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

```

5. **parameter\_defaults** の下で、OVS-DPDK の設定パラメーターを指定します。



#### 注記

**NeutronDPDKCoreList** および **NeutronDPDKMemoryChannels** はこの手順では **必須** の設定です。適切な値を指定せずに DPDK のデプロイを試みると、デプロイメントが失敗するか、不安定なデプロイメントとなってしまいます。

- a. **[allowed\_pattern: "'[0-9, -]+'"]** の形式で DPDK Poll Mode Driver (PMD) として使用可能なコアの一覧を指定します。

```

NeutronDpdkCoreList: "'4,6,20,22'"

```

OVS-DPDK のパフォーマンスを最適化するには、以下のオプションを検討してください。

- DPDK インターフェースの NUMA ノードに関連付けられた CPU を選択します。**cat /sys/class/net/<interface>/device/numa\_node** を使用してインターフェースに関連付けられた NUMA ノードをリストし、**lscpu** を使用してその NUMA ノードに関連付けられた CPU をリストします。
- CPU のシブリングをグループにまとめます (ハイパースレッディングの場合)。CPU のシブリングを特定するには、**cat /sys/devices/system/cpu/<cpu>/topology/thread\_siblings\_list** コマンドを実行します。
- CPU 0 はホストのプロセス用に確保します。
- PMD に割り当てられた CPU を分離して、ホストのプロセスがそれらの CPU を使用しないようにします。
- **NovaVcpuPinset** を使用して、PDM に割り当てられている CPU を Compute のスケジューリングから除外します。

b. **[allowed\_pattern: "[0-9]+"]** の形式でメモリーチャンネル数を指定します。

```
NeutronDpdkMemoryChannels: "4"
```

c. 各ソケットに対してヒュージページプールから事前割り当てされるメモリーを設定します。

```
NeutronDpdkSocketMemory: "2048,2048"
```

これは、CPU ソケットを昇順で指定するコンマ区切りの文字列です。この例では、2 NUMA ノード構成を前提としており、ヒュージページからソケット 0 に 2048 MB、ソケット 1 に 2048 MB を事前に割り当てるように設定しています。単一 NUMA ノード構成のシステムの場合には、この値を **1024,0** に設定してください。

d. OVS ブリッジの DPDK ドライバーの種別とデータパスの種別を設定します。

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

6. **parameter\_defaults** の下で、OVS 用の vhost-user ソケットディレクトリーを指定します。

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

7. **parameter\_defaults** の下で、ホストのプロセス用のメモリーを確保します。

```
NovaReservedHostMemory: "2048"
```

8. **parameter\_defaults** の下で、仮想マシンのプロセス用に確保する物理 CPU コアのコンマ区切りリストまたは範囲を指定します。

```
NovaVcpuPinSet: "8,10,12,14,18,24,26,28,30"
```

9. **parameter\_defaults** の下で、最も制約の多いフィルターからリストして、ノードのフィルタリングプロセスをさらに効率化します。  
 コンピュートは、フィルターの配列を使用してノードをフィルタリングします。これらのフィルターは、リスト順に適用されます。

```
NovaSchedulerDefaultFilters:
  "RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

10. **parameter\_defaults** の下で、**ComputeKernelArgs** パラメーターを追加して、初期ブート時にそれらのパラメーターがデフォルトの **grub** ファイルに追加されるようにします。

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



### 注記

これらのヒュージページは、仮想マシンにより消費されます。また、本手順に記載の **NeutronDpdkSocketMemory** パラメーターを使用すると OVS-DPDK により消費されます。仮想マシンが利用可能なヒュージページは、**boot** パラメーターから **NeutronDpdkSocketMemory** を減算した値です。

DPDK に関連付けられたフレーバーに **hw:mem\_page\_size=1GB** を追加する必要があります。追加しなかった場合には、そのインスタンスは DHCP から割り当てを受けなくなります。

11. **parameter\_defaults** の下に、チューニングする物理 CPU コアの一覧または範囲を指定します。  
 指定の引数は、tuned **cpu-partitioning** プロファイルに追加されます。

```
HostIsolatedCpusList: "'2,4,6,8,10,12,14,18,20,22,24,26,28,30'"
```

12. **parameter\_defaults** の下で、OVS-DPDK プロセスが **dpdk-lcore-mask** に使用する論理コアの一覧を設定します。これらのコアは、**NeutronDpdkCoreList** および **NovaVcpuPinSet** のコアリストとは相互に排他的である必要があります。

```
HostCpusList: "'0,1,2,3,5,17,18,19,21'"
```

#### 4.3.4. **controller.yaml** の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```
-
  type: linux_bond
  name: bond_api
  members:
    -
      type: interface
      name: nic3
    -
```



```

    type: interface
    name: nic4
  -

```

2. この Linux ボンディングに VLAN を割り当てます。

```

  -
    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. コンピュートノードへの OVS ブリッジを 2 つ作成します。

```

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

```

```

        type: interface
        name: nic5
-
    type: ovs_bridge
    name: br-link1
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    members:
      -
        type: interface
        name: nic6

```

#### 4.3.5. `compute.yaml` の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```

-
  type: linux_bond
  name: bond_api
  members:
    -
      type: interface
      name: nic3
    -
      type: interface
      name: nic4

```

2. この Linux ボンディングに VLAN を割り当てます。

```

-
  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. コントローラーにリンクするための DPDK ポートを 1 つずつ使用した ブリッジを 2 つ設定します。

■

```

-
  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: nic5
    -
      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: nic6

```



#### 注記

複数の DPDK デバイスが含まれるようにするには、追加する各 DPDK デバイスごとに **type** コードスニペットを繰り返してください。



#### 注記

OVS-DPDK を使用する場合には、**すべての** コンピュートノードのブリッジが **ovs\_user\_bridge** の種別である必要があります。director は設定を受け入れることができますが、Red Hat OpenStack Platform は **ovs\_bridge** と **ovs\_user\_bridge** の混在する構成はサポートしていません。

#### 4.3.6. **overcloud\_deploy.sh** スクリプトを実行します。

以下の例では、Bash スクリプト内で、OVS-DPDK 環境向けの **openstack overcloud deploy** コマンドを定義しています。

```

#!/bin/bash

openstack overcloud deploy --templates \
-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

```

ここで

- `/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml` は、Compute ロールの OVS-DPDK パラメーターを有効にするデフォルトの `neutron-ovs-dpdk.yaml` ファイルの場所です。
- `/home/stack/<relative-directory>/network-environment.yaml` は、`network-environment.yaml` ファイルのパスです。このファイルを使用して、`neutron-ovs-dpdk.yaml` ファイルからのデフォルト値を上書きします。



#### 注記

OVS-DPDK はセキュリティーグループおよびライブマイグレーションはサポートしていません。

## 4.4. VLAN トンネリングを使用した 2 ポートの OVS-DPDK データプレーンボンディングの設定

本項では、OpenStack 環境で、1 つの OVS-DPDK ボンディングが 2 つのデータポートで構成される OVS-DPDK を設定およびデプロイする手順について説明します。このユースケースでは、コントロールプレーンに Linux ボンディングも使用します。

### 4.4.1. `first-boot.yaml` の編集

`first-boot.yaml` ファイルを編集して、OVS と DPDK のパラメーターを設定し、`tuned` を CPU アフィニティー向けに構成します。

1. 追加のリソースを加えます。

```
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. `NeutronVhostUserSocketDir` の設定を決定します。

```
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. OVS の構成を設定します。

```

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. DPDKのパラメーターを設定します。

```

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=$((1<<core))
              bm[$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 $TUNED_CORES )
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpgk-init=true
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpgk-socket-mem=$SOCKET_MEMORY
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:pmd-cpu-mask=$pmd_cpu_mask
    ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpgk-lcore-mask=$host_cpu_mask
fi
params:
    $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
    $TUNED_CORES: {get_param: HostCpusList}
    $PMD_CORES: {get_param: NeutronDpgkCoreList}
    $SOCKET_MEMORY: {get_param: NeutronDpgkSocketMemory}

```

5. **tuned** の設定で CPU アフィニティを指定するように設定します。

```

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

```

```

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

```

#### 6. カーネル引数を設定します。

```

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"/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}

```

### 4.4.2. `post-install.yaml` の編集

#### 1. `tuned` の設定で CPU アフィニティーを指定するように設定します。

```

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}

```

#### 4.4.3. network-environment.yaml の編集

1. **resource\_registry** の下に OVS-DPDK 用のカスタムリソースを追加します。

```

resource_registry:
  # Specify the relative/absolute path to the config files you want
to use for override the default.
  OS::Triple0::Compute::Net::SoftwareConfig: nic-configs/compute-
ovs-dpdk.yaml
  OS::Triple0::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::Triple0::NodeUserData: first-boot.yaml
  OS::Triple0::NodeExtraConfigPost: post-install.yaml

```

2. **parameter\_defaults** の下でトンネルの種別を無効にして (値を "" に設定)、ネットワーク種別を **vlan** に設定します。

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

3. **parameter\_defaults** の下で、物理ネットワークを仮想ブリッジにマッピングします。

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

4. **parameter\_defaults** の下で、OpenStack Networking ML2 および Open vSwitch VLAN のマッピング範囲を指定します。

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

この例では、物理ネットワーク (**dpdk\_data**) 上の VLAN 範囲を設定しています。

5. **parameter\_defaults** の下で、OVS-DPDK の設定パラメーターを指定します。



#### 注記

**NeutronDPDKCoreList** および **NeutronDPDKMemoryChannels** はこの手順では **必須** の設定です。適切な値を指定せずに DPDK のデプロイを試みると、デプロイメントが失敗するか、不安定なデプロイメントとなってしまいます。

- a. **[allowed\_pattern: "'[0-9, -]+'"]** の形式で DPDK Poll Mode Driver (PMD) として使用可能なコアの一覧を指定します。

```
NeutronDpdkCoreList: "'4,6,20,22'"
```

OVS-DPDK のパフォーマンスを最適化するには、以下のオプションを検討してください。

- DPDK インターフェースの NUMA ノードに関連付けられた CPU を選択します。**cat /sys/class/net/<interface>/device/numa\_node** を使用してインターフェースに関連付けられた NUMA ノードをリストし、**lscpu** を使用してその NUMA ノードに関連付けられた CPU をリストします。
- CPU のシブリングをグループにまとめます (ハイパースレッディングの場合)。CPU のシブリングを特定するには、**cat /sys/devices/system/cpu/<cpu>/topology/thread\_siblings\_list** コマンドを実行します。
- CPU 0 はホストのプロセス用に確保します。
- PMD に割り当てられた CPU を分離して、ホストのプロセスがそれらの CPU を使用しないようにします。
- **NovaVcpuPinset** を使用して、PDM に割り当てられている CPU を Compute のスケジューリングから除外します。

- b. **[allowed\_pattern: "[0-9]+'"]** の形式でメモリーチャンネル数を指定します。

```
NeutronDpdkMemoryChannels: "4"
```

- c. 各ソケットに対してヒュージページプールから事前割り当てされるメモリーを設定します。

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

これは、CPU ソケットの昇順で指定するコンマ区切りの文字列です。この例では、2 NUMA ノード構成を前提としており、ヒュージページからソケット 0 に 1024 MB、ソケット 1 に 1024 MB を事前に割り当てるように設定しています。単一 NUMA ノード構成のシステムの場合には、この値を **1024,0** に設定してください。

- d. OVS ブリッジの DPDK ドライバーの種別とデータパスの種別を設定します。

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

6. **parameter\_defaults** の下で、OVS 用の vhost-user ソケットディレクトリーを指定します。

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

7. **parameter\_defaults** の下で、ホストのプロセス用のメモリーを確保します。

```
NovaReservedHostMemory: "2048"
```

8. **parameter\_defaults** の下で、仮想マシンのプロセス用に確保する物理 CPU コアのコンマ区切りリストまたは範囲を指定します。

```
NovaVcpuPinSet: "8,10,12,14,18,24,26,28,30"
```

9. **parameter\_defaults** の下で、最も制約の多いフィルターからリストして、ノードのフィルタリングプロセスをさらに効率化します。  
コンピュートは、フィルターの配列を使用してノードをフィルタリングします。これらのフィルターは、リスト順に適用されます。

```
NovaSchedulerDefaultFilters:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

10. **parameter\_defaults** の下で、**ComputeKernelArgs** パラメーターを追加して、初期ブート時にそれらのパラメーターがデフォルトの **grub** ファイルに追加されるようにします。

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

## 注記

これらのヒュージページは、仮想マシンにより消費されます。また、本手順に記載の **NeutronDpdkSocketMemory** パラメーターを使用すると OVS-DPDK により消費されます。仮想マシンが利用可能なヒュージページは、**boot** パラメーターから **NeutronDpdkSocketMemory** を減算した値です。

DPDK に関連付けられたフレーバーに **hw:mem\_page\_size=1GB** を追加する必要があります。追加しなかった場合には、そのインスタンスは DHCP から割り当てを受けなくなります。

11. **parameter\_defaults** の下に、チューニングする物理 CPU コアの一覧または範囲を指定します。

指定の引数は、tuned **cpu-partitioning** プロファイルに追加されます。

```
HostIsolatedCpusList: "'2,4,6,8,10,12,14,18,20,22,24,26,28,30'"
```

12. **parameter\_defaults** の下で、OVS-DPDK プロセスが dpdk-lcore-mask に使用する論理コアの一覧を設定します。これらのコアは、**NeutronDpdkCoreList** および **NovaVcpuPinSet** のコアリストとは相互に排他的である必要があります。

```
HostCpusList: "'0,1,2,3,5,17,18,19,21'"
```

#### 4.4.4. **controller.yaml** の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```
-
  type: linux_bond
  name: bond1
  bonding_options: "mode=active-backup"
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  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}
  members:
    -
      type: interface
      name: nic2
      primary: true
    -
      type: interface
      name: nic3
-
```

2. この Linux ボンディングに VLAN を割り当てます。

```
-
  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. Compute ノードに接続する OVS ブリッジを作成します。

```

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

```

#### 4.4.5. compute.yaml の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```

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

```

2. この Linux ボンディングに VLAN を割り当てます。

```

-
  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. コントローラーにリンクする OVS-DPDK ボンディングに 2 つの DPDK ポートにブリッジを設定します。

```

-
  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: nic5
        -
          type: ovs_dpdk_port
          name: dpdk1
          members:
            -
              type: interface
              name: nic6

```



#### 注記

複数の DPDK デバイスが含まれるようにするには、追加する各 DPDK デバイスごとに **type** コードスニペットを繰り返してください。



#### 注記

OVS-DPDK を使用する場合には、**すべての** コンピュートノードのブリッジが **ovs\_user\_bridge** の種別である必要があります。director は設定を受け入れることができますが、Red Hat OpenStack Platform は **ovs\_bridge** と **ovs\_user\_bridge** の混在する構成はサポートしていません。

#### 4.4.6. **overcloud\_deploy.sh** スクリプトを実行します。

以下の例では、Bash スクリプト内で、OVS-DPDK 環境向けの **openstack overcloud deploy** コマンドを定義しています。

```
#!/bin/bash

openstack overcloud deploy --templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-
dppdk.yaml \
-e /home/stack/ospd-11-vlan-ovs-dppdk-bonding-dataplane-bonding-
ctlplane/network-environment.yaml
```

ここで

- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dppdk.yaml** は、Compute ロールの OVS-DPDK パラメーターを有効にするデフォルトの **neutron-ovs-dppdk.yaml** ファイルの場所です。
- **/home/stack/<relative-directory>/network-environment.yaml** は、**network-environment.yaml** ファイルのパスです。このファイルを使用して、**neutron-ovs-dppdk.yaml** ファイルからのデフォルト値を上書きします。



#### 注記

OVS-DPDK はセキュリティーグループおよびライブマイグレーションはサポートしていません。

## 4.5. VXLAN トンネリングを使用した単一ポートの OVS-DPDK の設定

本項では、OpenStack 環境にコントロールプレーンの Linux ボンディングと VXLAN トンネリングを使用する単一データプレーンポートの OVS-DPDK を設定およびデプロイする手順について説明します。

### 4.5.1. first-boot.yaml の編集

**first-boot.yaml** ファイルを編集して、OVS と DPDK のパラメーターを設定し、**tuned** を CPU アフィニティー向けに構成します。

1. 追加のリソースを加えます。

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

2. **NeutronVhostUserSocketDir** の設定を決定します。

```
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
fi
params:
    $COMPUTE_HOSTNAME_FORMAT: {get_param:
ComputeHostnameFormat}
    $NEUTRON_VHOSTUSER_SOCKET_DIR: {get_param:
NeutronVhostuserSocketDir}
```

### 3. DPDKのパラメーターを設定します。

```
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=$((1<<core))
              bm[$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 $TUNED_CORES )
        ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-init=true
        ovs-vsctl --no-wait set Open_vSwitch .
other_config:dpdk-socket-mem=$SOCKET_MEMORY
        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}
        $TUNED_CORES: {get_param: HostCpusList}
        $PMD_CORES: {get_param: NeutronDpdkCoreList}
        $SOCKET_MEMORY: {get_param: NeutronDpdkSocketMemory}

```

#### 4. tuned の設定で CPU アフィニティを指定するように設定します。

```

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

```

```

yum install -y tuned-profiles-cpu-partitioning

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

```

#### 5. カーネル引数を設定します。

```

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"/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}

```

### 4.5.2. `post-install.yaml` の編集

#### 1. `tuned` の設定で CPU アフィニティーを指定するように設定します。

```
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}

```

### 4.5.3. network-environment.yaml の編集

1. **resource\_registry** の下に OVS-DPDK 用のカスタムリソースを追加します。

```

resource_registry:
  # Specify the relative/absolute path to the config files you want
to use for override the default.
  OS::Triple0::Compute::Net::SoftwareConfig: nic-
configs/compute.yaml
  OS::Triple0::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
  OS::Triple0::NodeUserData: first-boot.yaml
  OS::Triple0::NodeExtraConfigPost: post-install.yaml
  OS::Triple0::AllNodes::Validation: dummy_all_nodes-validation.yaml

```

2. **parameter\_defaults** の下でトンネルの種別とテナントの種別を **vxlan** に設定します。

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

3. **parameter\_defaults** の下で、OVS-DPDK の設定パラメーターを指定します。



#### 注記

**NeutronDPDKCoreList** および **NeutronDPDKMemoryChannels** はこの手順では **必須** の設定です。適切な値を指定せずに DPDK のデプロイを試みると、デプロイメントが失敗するか、不安定なデプロイメントとなってしまいます。

- a. **[allowed\_pattern: "'[0-9, -]+'"]** の形式で DPDK Poll Mode Driver (PMD) として使用可能なコアの一覧を指定します。

```
NeutronDpdkCoreList: "'4,6,20,22'"
```

OVS-DPDK のパフォーマンスを最適化するには、以下のオプションを検討してください。

- DPDK インターフェースの NUMA ノードに関連付けられた CPU を選択します。**cat /sys/class/net/<interface>/device/numa\_node** を使用してインターフェースに関連付けられた NUMA ノードをリストし、**lscpu** を使用してその NUMA ノードに関連付けられた CPU をリストします。
- CPU のシブリングをグループにまとめます (ハイパースレッディングの場合)。CPU のシブリングを特定するには、**cat /sys/devices/system/cpu/<cpu>/topology/thread\_siblings\_list** コマンドを実行します。
- CPU 0 はホストのプロセス用に確保します。
- PMD に割り当てられた CPU を分離して、ホストのプロセスがそれらの CPU を使用しないようにします。
- **NovaVcpuPinset** を使用して、PDM に割り当てられている CPU を Compute のスケジューリングから除外します。

- b. **[allowed\_pattern: "[0-9]+'"]** の形式でメモリーチャンネル数を指定します。

```
NeutronDpdkMemoryChannels: "4"
```

- c. 各ソケットに対してヒュージページプールから事前割り当てされるメモリーを設定します。

```
NeutronDpdkSocketMemory: "'2048,2048'"
```

これは、CPU ソケットを昇順で指定するコンマ区切りの文字列です。この例では、2 NUMA ノード構成を前提としており、ヒュージページからソケット 0 に 2048 MB、ソケット 1 に 2048 MB を事前に割り当てるように設定しています。単一 NUMA ノード構成のシステムの場合には、この値を **1024,0** に設定してください。

- d. OVS ブリッジの DPDK ドライバーの種別とデータパスの種別を設定します。

■

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

4. **parameter\_defaults** の下で、OVS 用の vhost-user ソケットディレクトリーを指定します。

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

5. **parameter\_defaults** の下で、ホストのプロセス用のメモリーを確保します。

```
NovaReservedHostMemory: "2048"
```

6. **parameter\_defaults** の下で、仮想マシンのプロセス用に確保する物理 CPU コアのコンマ区切りリストまたは範囲を指定します。

```
NovaVcpuPinSet: "8,10,12,14,18,24,26,28,30"
```

7. **parameter\_defaults** の下で、最も制約の多いフィルターからリストして、ノードのフィルタリングプロセスをさらに効率化します。  
コンピュートは、フィルターの配列を使用してノードをフィルタリングします。これらのフィルターは、リスト順に適用されます。

```
NovaSchedulerDefaultFilters:
"RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesF
ilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter"
```

8. **parameter\_defaults** の下で、**ComputeKernelArgs** パラメーターを追加して、初期ブート時にそれらのパラメーターがデフォルトの **grub** ファイルに追加されるようにします。

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



### 注記

これらのヒュージページは、仮想マシンにより消費されます。また、本手順に記載の **NeutronDpdkSocketMemory** パラメーターを使用すると OVS-DPDK により消費されます。仮想マシンが利用可能なヒュージページは、**boot** パラメーターから **NeutronDpdkSocketMemory** を減算した値です。

DPDK に関連付けられたフレーバーに **hw:mem\_page\_size=1GB** を追加する必要があります。追加しなかった場合には、そのインスタンスは DHCP から割り当てを受けなくなります。

9. **parameter\_defaults** の下に、チューニングする物理 CPU コアの一覧または範囲を指定します。  
指定の引数は、tuned **cpu-partitioning** プロファイルに追加されます。

```
HostCpusList: "'2,4,6,8,10,12,14,18,20,22,24,26,28,30'"
```

#### 4.5.4. controller.yaml の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```

-
  type: linux_bond
  name: bond_api
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  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}
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
-

```

2. この Linux ボンディングに VLAN を割り当てます。

```

-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: vlan
  vlan_id: {get_param: ExternalNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: ExternalIpSubnet}
-

```

3. Compute ノードに接続する OVS ブリッジを作成します。

```

-
  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}

```

#### 4.5.5. compute.yaml の編集

1. 分離ネットワーク用にコントロールプレーンの Linux ボンディングを作成します。

```

-
  type: linux_bond
  name: bond_api
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: interface
      name: nic4

```

2. この Linux ボンディングに VLAN を割り当てます。

```

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

```

3. コントローラーにリンクするための DPDK ポートを使用したブリッジを設定します。

```

-
  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: nic5
```



#### 注記

複数の DPDK デバイスが含まれるようにするには、追加する各 DPDK デバイスごとに **type** コードスニペットを繰り返してください。



#### 注記

OVS-DPDK を使用する場合には、**すべての** コンピュートノードのブリッジが **ovs\_user\_bridge** の種別である必要があります。director は設定を受け入れることができますが、Red Hat OpenStack Platform は **ovs\_bridge** と **ovs\_user\_bridge** の混在する構成はサポートしていません。

### 4.5.6. `overcloud_deploy.sh` スクリプトを実行します。

以下の例では、Bash スクリプト内で、OVS-DPDK 環境向けの **openstack overcloud deploy** コマンドを定義しています。

```
#!/bin/bash

openstack overcloud deploy --templates \
-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
```

ここで

- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml** は、Compute ロールの OVS-DPDK パラメーターを有効にするデフォルトの **neutron-ovs-dpdk.yaml** ファイルの場所です。
- **/home/stack/<relative-directory>/network-environment.yaml** は、**network-environment.yaml** ファイルのパスです。このファイルを使用して、**neutron-ovs-dpdk.yaml** ファイルからのデフォルト値を上書きします。



#### 注記

OVS-DPDK はセキュリティーグループおよびライブマイグレーションはサポートしていません。

## 4.6. OVS-DPDK コンポーザブルロールの設定

本項では、VLAN トンネリングを使用した OVS-DPDK および SR-IOV 向けのコンポーザブルロールの設定方法について説明します。コンポーザブルロールの作成/デプロイプロセスは、以下の作業が含まれます。

- **role\_data.yaml** ファイルのローカルコピーで新規ロールを定義します。
- **network\_environment.yaml** ファイルを編集して、この新規ロールを追加します。



- 更新したロールセットでオーバークラウドをデプロイします。

以下の例では、**ComputeOvsDpdk** と **ComputeSriov** は、コンピュートノード用のコンポーザブルロールで、サポートされている NIC が搭載されたノードでのみ DPDK または SR-IOV を有効にします。Red Hat OpenStack Platform によって提供される既存のデフォルトロールセットは、`/home/stack/roles_data.yaml` ファイルに保管されます。

#### 4.6.1. コンポーザブルロールを作成するための `roles_data.yaml` の編集

`roles_data.yaml` ファイルを `/home/stack/templates` ディレクトリーにコピーして、新しい **ComputeOvsDpdk** および **ComputeSriov** のロールを追加します。

##### 1. OVS-DPDK のコンポーザブルロールの定義

```
- name: ComputeOvsDpdk
  CountDefault: 1
  HostnameFormatDefault: compute-ovs-dpdk-%index%
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephClient
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::Timezone
    - OS::Triple0::Services::Ntp
    - OS::Triple0::Services::Snmp
    - OS::Triple0::Services::Sshd
    - OS::Triple0::Services::NovaCompute
    - OS::Triple0::Services::NovaLibvirt
    - OS::Triple0::Services::Kernel
    - OS::Triple0::Services::ComputeNeutronCorePlugin
    - OS::Triple0::Services::ComputeNeutronOvsDpdkAgent
    - OS::Triple0::Services::ComputeCeilometerAgent
    - OS::Triple0::Services::ComputeNeutronL3Agent
    - OS::Triple0::Services::ComputeNeutronMetadataAgent
    - OS::Triple0::Services::TripleoPackages
    - OS::Triple0::Services::TripleoFirewall
    - OS::Triple0::Services::OpenDaylightOvs
    - OS::Triple0::Services::SensuClient
    - OS::Triple0::Services::FluentdClient
    - OS::Triple0::Services::AuditD
    - OS::Triple0::Services::Collectd
```

この例では、新しい **ComputeOvsDpdk** ロールのサービスはすべて通常の Compute ロールのサービスと同じですが、**ComputeNeutronOvsAgent** は例外で、OVS-DPDK サービスにマッピングする **ComputeNeutronOvsDpdkAgent** に置き換える必要があります。

##### 2. SR-IOV のコンポーザブルロールを定義します。

```
- name: ComputeSriov
  CountDefault: 1
  HostnameFormatDefault: compute-sriov-%index%
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephClient
```

```

- OS::Triple0::Services::CephExternal
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::NovaCompute
- OS::Triple0::Services::NovaLibvirt
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::ComputeNeutronCorePlugin
- OS::Triple0::Services::ComputeNeutronOvsAgent
- OS::Triple0::Services::ComputeCeilometerAgent
- OS::Triple0::Services::ComputeNeutronL3Agent
- OS::Triple0::Services::ComputeNeutronMetadataAgent
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::NeutronSriovAgent
- OS::Triple0::Services::OpenDaylightOvs
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::FluentdClient
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::Collectd

```

#### 4.6.2. 新規コンポーザブルロール向けの **network-environment.yaml** の編集

**network-environment.yaml** ファイルに OVS-DPDK および SR-IOV サービス向けのリソースマッピングと、このノードのネットワーク設定を以下のように追加します。

```

resource_registry:
  # Specify the relative/absolute path to the config files you want to use
  # for override the default.
  OS::Triple0::ComputeSriov::Net::SoftwareConfig: nic-configs/compute-
  sriov.yaml
  OS::Triple0::Controller::Net::SoftwareConfig: nic-
  configs/controller.yaml
  OS::Triple0::ComputeOvsDpdk::Net::SoftwareConfig: nic-configs/compute-
  ovs-dpdk.yaml

  OS::Triple0::Services::ComputeNeutronOvsDpdkAgent: /usr/share/openstack-
  tripleo-heat-templates/puppet/services/neutron-ovs-dpdk-agent.yaml
  OS::Triple0::Services::NeutronSriovAgent: /usr/share/openstack-tripleo-
  heat-templates/puppet/services/neutron-sriov-agent.yaml

```

**network-environment.yaml** ファイルの残りを設定して、OpenStack デプロイメントの必要に応じて **neutron-ovs-dpdk-agent.yaml** および **neutron-sriov-agent.yaml** のファイルからのデフォルトのパラメーターを上書きします。

#### 4.6.3. **overcloud\_deploy.sh** スクリプトを実行します。

以下の例では、コンポーザブルロールを使用する **openstack overcloud deploy** の Bash スクリプトを定義しています。

```

# #!/bin/bash

openstack overcloud deploy --templates \

```

```
-r /home/stack/templates/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` は、更新された `roles_data.yaml` ファイルの場所です。このファイルが OVS-DPDK と SR-IOV のコンポーザブルロールを定義します。

## 4.7. 既知の制限事項

NFV のユースケース向けに Red Hat OpenStack Platform 11 で OVS-DPDK を設定する場合には特定の制限事項があります。

- ヒュージページは OVS-DPDK を使用するホスト上で実行される全インスタンスに必要です。ゲストのヒュージページがない場合には、インターフェースは表示されても機能しません。
- TAP デバイスは DPDK をサポートしていないため、それらのデバイスを使用するサービスのパフォーマンスが低下します。たとえば、DVR、FWaaS、LBaaS などのサービスは TAP デバイスを使用します。
  - OVS-DPDK では、**netdev datapath** で DVR を有効化することができますが、パフォーマンスが低いので、実稼働環境には適していません。DVR はカーネルの名前空間と TAP デバイスを使用してルーティングを実行します。
  - OVS-DPDK で DVR ルーティングのパフォーマンスを良好な状態にするには、OpenFlow ルールとしてルーティングを実装する ODL などのコントローラーを使用する必要があります。OVS-DPDK では、OpenFlow ルーティングは、Linux カーネルインターフェースによって生じるボトルネックをなくすので、データパスの完全なパフォーマンスが維持されます。
- OVS-DPDK を使用する場合には、**すべて** のブリッジがコンピュータード上で **ovs\_user\_bridge** の種別である必要があります。director は設定を受け入れることができますが、Red Hat OpenStack Platform はコンピュータード上で **ovs\_bridge** と **ovs\_user\_bridge** の混在する構成はサポートしていません。

## 4.8. フレーバーの作成と OVS-DPDK 用インスタンスのデプロイ

NFV を実装する Red Hat OpenStack Platform デプロイメントの OVS-DPDK の設定を完了した後は、以下の手順に従ってフレーバーを作成してインスタンスをデプロイすることができます。

1. アグリゲートグループを作成して、OVS-DPDK 用にホストを追加します。

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

2. フレーバーを作成します。

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

このコマンドでは、**m1.medium\_huge\_4cpu** はフレーバー名、**4096** は MB 単位のメモリー容量、**150** は GB 単位のディスク容量 (デフォルトでは 0 G)、**4** は仮想 CPU 数を設定しています。

3. フレーバーの追加のプロパティを設定します。

```
# openstack flavor set --property hw:cpu_policy=dedicated --
property hw:cpu_thread_policy=require --property
hw:mem_page_size=large --property hw:numa_nodes=1 --property
hw:numa_mempolicy=preferred --property hw:numa_cpus.0=0,1,2,3 --
property hw:numa_mem.0=4096 m1.medium_huge_4cpu
```

このコマンドでは、**m1.medium\_huge\_4cpu** はフレーバー名を指定しており、残りはそのフレーバーのその他のプロパティを設定しています。

4. インスタンスをデプロイします。

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

このコマンドでは、**m1.medium\_huge\_4cpu** はフレーバーの名前または ID、**dpdk** はサーバーのアベイラビリティゾーン、**rhel\_7.3** はインスタンスの作成に使用するイメージ (名前または ID)、**<net-id>** はサーバー上の NIC を設定します。

これで、NFV ユースケースの OVS-DPDK 向けインスタンスのデプロイが完了しました。

**multi-queue** を OVS-DPDK で使用するには、上記の手順に数ステップを追加する必要があります。フレーバーを作成する前に、以下のステップを実行してください。

1. イメージのプロパティを設定します。

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

ここで、**hw\_vif\_multiqueue\_enabled=true** はこのイメージ上でマルチキューを有効にするためのプロパティで、**<image-id>** は変更するイメージの名前または ID です。

2. フレーバーの追加のプロパティを設定します。

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

ここで、**m1.vm\_mq** はフレーバーの ID または名前、残りのオプションはそのフレーバーのマルチキューを有効化します。

## 4.9. 設定のトラブルシューティング

本項では、DPDK-OVS 設定のトラブルシューティングの手順を説明します。

1. ブリッジの設定を見直して、ブリッジが **datapath\_type=netdev** で作成されたことを確認します。

```
# 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. **neutron-ovs-agent** が自動的に起動するように設定されていることを確認して、OVS サービスをレビューします。

```

# 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

```

サービスの起動に問題がある場合には、以下のコマンドを実行して関連のメッセージを表示することができます。

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

3. **ovs-dpdk** の PMD CPU マスクが CPU にピンングされていることを確認します。HT の場合には、シブリング CPU を使用します。  
たとえば **CPU4** を例に取ります。

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

CPU 4 と 20 を使用します。

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

ステータスを表示します。

```

# 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	dppk_watchdog3
3165	OTHER	0	6	237634	0	vhost_thread1
3166	OTHER	0	6	3665	0	urcu2

## 第5章 詳細情報

以下の表には、その他の参考となる Red Hat のドキュメントの一覧を記載しています。

Red Hat OpenStack Platform のドキュメントスイートは [Red Hat OpenStack Platform 11 の製品ドキュメントスイート](#) から参照してください。

表5.1 参考資料一覧

コンポーネント	参考情報
Red Hat Enterprise Linux	Red Hat OpenStack Platform は Red Hat Enterprise Linux 7.3 でサポートされています。Red Hat Enterprise Linux のインストールに関する情報は、 <a href="#">Red Hat Enterprise Linux のドキュメント</a> から対応するインストールガイドを参照してください。
Red Hat OpenStack Platform	<p>OpenStack のコンポーネントとそれらの依存関係をインストールするには、Red Hat OpenStack Platform director を使用します。director は、基本的な OpenStack 環境をアンダークラウドとして使用して、最終的な <b>オーバークラウド</b> 内の OpenStack ノードをインストール、設定、管理します。デプロイしたオーバークラウドに必要な環境に加えて、アンダークラウドをインストールするための追加のホストマシンが 1 台必要となる点に注意してください。詳しい手順は、<a href="#">『Red Hat OpenStack Platform director のインストールと使用方法』</a> を参照してください。</p> <p>Red Hat OpenStack Platform director を使用して、ネットワークの分離、ストレージ設定、SSL 通信、一般的な設定方法など Red Hat OpenStack Platform のエンタープライズ環境の高度な機能設定に関する情報は <a href="#">『オーバークラウドの高度なカスタマイズ』</a> を参照してください。</p> <p>Red Hat OpenStack Platform コンポーネントを手動でインストールすることもできます。方法については、<a href="#">『手動インストール手順』</a> を参照してください。</p>
NFV のドキュメント	<p>NFV の概念に関する俯瞰的な情報は、<a href="#">『ネットワーク機能仮想化 (NFV) の製品ガイド』</a> を参照してください。</p> <p>NFV での Red Hat OpenStack Platform のデプロイメント計画に関する詳細は、<a href="#">『Network Functions Virtualization Planning and Prerequisite Guide』</a> を参照してください。</p>

## 付録A SR-IOV YAML ファイルのサンプル

本項では、参考として SR-IOV YAML ファイルのサンプルを紹介します。

### A.1. SR-IOV コンポーザブルロールの YAML ファイルのサンプル

#### A.1.1. roles\_data.yaml

```
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephMds
    - OS::Triple0::Services::CephMon
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::CephRbdMirror
    - OS::Triple0::Services::CephRgw
    - OS::Triple0::Services::CinderApi
    - OS::Triple0::Services::CinderBackup
    - OS::Triple0::Services::CinderScheduler
    - OS::Triple0::Services::CinderVolume
    - OS::Triple0::Services::Congress
    - OS::Triple0::Services::Kernel
    - OS::Triple0::Services::Keystone
    - OS::Triple0::Services::GlanceApi
    - OS::Triple0::Services::HeatApi
    - OS::Triple0::Services::HeatApiCfn
    - OS::Triple0::Services::HeatApiCloudwatch
    - OS::Triple0::Services::HeatEngine
    - OS::Triple0::Services::MySQL
    - OS::Triple0::Services::MySQLClient
    - OS::Triple0::Services::NeutronDhcpAgent
    - OS::Triple0::Services::NeutronL3Agent
    - OS::Triple0::Services::NeutronMetadataAgent
    - OS::Triple0::Services::NeutronApi
    - OS::Triple0::Services::NeutronCorePlugin
    - OS::Triple0::Services::NeutronOvsAgent
    - OS::Triple0::Services::RabbitMQ
    - OS::Triple0::Services::HAproxy
    - OS::Triple0::Services::Keepalived
    - OS::Triple0::Services::Memcached
    - OS::Triple0::Services::Pacemaker
    - OS::Triple0::Services::Redis
    - OS::Triple0::Services::NovaConductor
    - OS::Triple0::Services::MongoDb
    - OS::Triple0::Services::NovaApi
    - OS::Triple0::Services::NovaPlacement
    - OS::Triple0::Services::NovaMetadata
    - OS::Triple0::Services::NovaScheduler
    - OS::Triple0::Services::NovaConsoleauth
    - OS::Triple0::Services::NovaVncProxy
    - OS::Triple0::Services::Ec2Api
    - OS::Triple0::Services::Ntp
    - OS::Triple0::Services::SwiftProxy
```



- OS::Triple0::Services::SwiftStorage
- OS::Triple0::Services::SwiftRingBuilder
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::CeilometerApi
- OS::Triple0::Services::CeilometerCollector
- OS::Triple0::Services::CeilometerExpirer
- OS::Triple0::Services::CeilometerAgentCentral
- OS::Triple0::Services::CeilometerAgentNotification
- OS::Triple0::Services::Horizon
- OS::Triple0::Services::GnocchiApi
- OS::Triple0::Services::GnocchiMetricd
- OS::Triple0::Services::GnocchiStatsd
- OS::Triple0::Services::ManilaApi
- OS::Triple0::Services::ManilaScheduler
- OS::Triple0::Services::ManilaBackendGeneric
- OS::Triple0::Services::ManilaBackendNetapp
- OS::Triple0::Services::ManilaBackendCephFs
- OS::Triple0::Services::ManilaShare
- OS::Triple0::Services::AodhApi
- OS::Triple0::Services::AodhEvaluator
- OS::Triple0::Services::AodhNotifier
- OS::Triple0::Services::AodhListener
- OS::Triple0::Services::SaharaApi
- OS::Triple0::Services::SaharaEngine
- OS::Triple0::Services::IronicApi
- OS::Triple0::Services::IronicConductor
- OS::Triple0::Services::NovaIronic
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::OpenDaylightApi
- OS::Triple0::Services::OpenDaylightOvs
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::FluentdClient
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::BarbicanApi
- OS::Triple0::Services::PankoApi
- OS::Triple0::Services::Tacker
- OS::Triple0::Services::Zaqar
- OS::Triple0::Services::OVNDBs
- OS::Triple0::Services::NeutronML2FujitsuCfab
- OS::Triple0::Services::NeutronML2FujitsuFossw
- OS::Triple0::Services::CinderHPELeftHandISCSI
- OS::Triple0::Services::Etc
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::OctaviaApi
- OS::Triple0::Services::OctaviaHealthManager
- OS::Triple0::Services::OctaviaHousekeeping
- OS::Triple0::Services::OctaviaWorker

###- name: Compute

- name: ComputeSriov
  - CountDefault: 1
  - HostnameFormatDefault: 'compute-sriov-%index%'
  - disable\_upgrade\_deployment: True

```

ServicesDefault:
  - OS::Triple0::Services::CACerts
  - OS::Triple0::Services::CephClient
  - OS::Triple0::Services::CephExternal
  - OS::Triple0::Services::Timezone
  - OS::Triple0::Services::Ntp
  - OS::Triple0::Services::Snmp
  - OS::Triple0::Services::Sshd
  - OS::Triple0::Services::NovaCompute
  - OS::Triple0::Services::NovaLibvirt
  - OS::Triple0::Services::Kernel
  - OS::Triple0::Services::ComputeNeutronCorePlugin
  - OS::Triple0::Services::ComputeNeutronOvsAgent
  - OS::Triple0::Services::ComputeCeilometerAgent
  - OS::Triple0::Services::ComputeNeutronL3Agent
  - OS::Triple0::Services::ComputeNeutronMetadataAgent
  - OS::Triple0::Services::TripleoPackages
  - OS::Triple0::Services::TripleoFirewall
  - OS::Triple0::Services::NeutronSriovAgent
  - OS::Triple0::Services::OpenDaylightOvs
  - OS::Triple0::Services::SensuClient
  - OS::Triple0::Services::FluentdClient
  - OS::Triple0::Services::AuditD
  - OS::Triple0::Services::Collectd

- name: Compute
  CountDefault: 1
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephClient
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::Timezone
    - OS::Triple0::Services::Ntp
    - OS::Triple0::Services::Snmp
    - OS::Triple0::Services::Sshd
    - OS::Triple0::Services::NovaCompute
    - OS::Triple0::Services::NovaLibvirt
    - OS::Triple0::Services::Kernel
    - OS::Triple0::Services::ComputeNeutronCorePlugin
    - OS::Triple0::Services::ComputeNeutronOvsAgent
    - OS::Triple0::Services::ComputeCeilometerAgent
    - OS::Triple0::Services::ComputeNeutronL3Agent
    - OS::Triple0::Services::ComputeNeutronMetadataAgent
    - OS::Triple0::Services::TripleoPackages
    - OS::Triple0::Services::TripleoFirewall
    - OS::Triple0::Services::NeutronSriovAgent
    - OS::Triple0::Services::OpenDaylightOvs
    - OS::Triple0::Services::SensuClient
    - OS::Triple0::Services::FluentdClient
    - OS::Triple0::Services::AuditD
    - OS::Triple0::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
          # Install the tuned package
          yum install -y tuned-profiles-cpu-partitioning

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

          sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/g' -i
/etc/default/grub ;
          grub2-mkconfig -o /etc/grub2.cfg

          reboot
        fi
      params:
        $KERNEL_ARGS: {get_param: ComputeKernelArgs}
        $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
        $TUNED_CORES: {get_param: HostIsolatedCoreList}

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

```

### 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::Compute::Net::SoftwareConfig: nic-configs/compute.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
  OvercloudComputeFlavor: compute
  # Number of nodes to deploy.
  ControllerCount: 1
  ComputeSriovCount: 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%'
  ComputeSriovHostnameFormat: 'compute-sriov-%index%'
  ComputeHostnameFormat: 'compute-%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: "4,20,5,21,6,22,7,23"
# Kernel arguments for Compute node
ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32
iommu=pt intel_iommu=on"
# A list or range of physical CPU cores to be tuned.
# The given args will be appended to the tuned cpu-partitioning profile.
HostIsolatedCoreList: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"

```

### 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
              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}

```

### A.1.6. 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
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)
    type: comma_delimited_list
that will be added to resolv.conf.

```

```

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}

```

### A.1.7. overcloud\_deploy.sh

```

#!/bin/bash

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

```

## 付録B OVS-DPDK YAML ファイルのサンプル

本項では、参考として OVS-DPDK YAML ファイルのサンプルを紹介します。

### B.1. VLAN OVS-DPDK YAML ファイルのサンプル

#### 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 separated 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: ""
  HostCpusList:
    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: ""
  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 processes (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 and 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
                        yum install -y tuned-profiles-cpu-partitioning

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

compute_kernel_args:

```

```

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

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

```

### B.1.2. 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.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::Triple0::Compute::Net::SoftwareConfig: nic-configs/compute-ovs-
  dpdk.yaml
  OS::Triple0::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: 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: '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: "'4,6,20,22'"
# 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"

# 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: "8,10,12,14,18,24,26,28,30"
# 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: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"
# List of logical cores to be used by ovs-dpdk processes (dpdk-lcore-
mask)
HostCpusList: "'3,5,7,19,21,23'"

```

#### 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
              defroute: false
            -
              type: interface
              name: nic2
              use_dhcp: false
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
            -
              type: linux_bond
              name: bond_api
              members:
                -
                  type: interface
                  name: nic3
                -
                  type: interface

```

```

        name: nic4
    -
      type: vlan
      vlan_id: {get_param: InternalApiNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: InternalApiIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: TenantNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: StorageNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: StorageIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: StorageMgmtNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: StorageMgmtIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: ExternalNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: ExternalIpSubnet}
      routes:
        -
          default: true
          next_hop: {get_param: ExternalInterfaceDefaultRoute}
    -
      type: ovs_bridge
      name: br-link0
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      members:
        -
          type: interface
          name: nic5

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

```

**B.1.5. 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
              defroute: false
            -
              type: linux_bridge
              name: br-isolated
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              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}
            members:
              -
                type: interface
                name: nic2
                # force the MAC address of the bridge to this interface
                primary: true
              -
                type: vlan
                vlan_id: {get_param: InternalApiNetworkVlanID}
                device: br-isolated

```

```

        addresses:
          -
            ip_netmask: {get_param: InternalApiIpSubnet}
          -
            type: vlan
            vlan_id: {get_param: TenantNetworkVlanID}
            device: br-isolated
            addresses:
              -
                ip_netmask: {get_param: TenantIpSubnet}
          -
            type: ovs_user_bridge
            name: br-link
            use_dhcp: false
            members:
              -
                type: ovs_dpdk_port
                name: dpdk0
                members:
                  -
                    type: interface
                    name: nic3

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. 2 ポートの VLAN OVS-DPDK YAML ファイルのサンプル

### 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: ""
  HostCpusList:
    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: ""
  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 and 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
            yum install -y tuned-profiles-cpu-partitioning

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

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

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

```

### B.2.2. 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.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::Triple0::Compute::Net::SoftwareConfig: nic-configs/compute-ovs-
dppk.yaml
    OS::Triple0::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
    OS::Triple0::NodeUserData: first-boot.yaml
    OS::Triple0::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: '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

  # 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: "'4,6,20,22'"
# 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"

# 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: "8,10,12,14,18,24,26,28,30"
# 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: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"
# List of logical cores to be used by ovs-dpdk processes (dpdk-lcore-
mask)
HostCpusList: "'3,5,7,19,21,23'"

```

#### 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
              defroute: false
            -
              type: interface
              name: nic2
              use_dhcp: false
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
            -
              type: linux_bond
              name: bond_api
              members:
                -
                  type: interface
                  name: nic3
                -
                  type: interface
                  name: nic4
            -
              type: vlan
              vlan_id: {get_param: InternalApiNetworkVlanID}
              device: bond_api
              addresses:
                -
                  ip_netmask: {get_param: InternalApiIpSubnet}
            -
              type: vlan
              vlan_id: {get_param: TenantNetworkVlanID}
              device: bond_api
              addresses:
                -
                  ip_netmask: {get_param: TenantIpSubnet}
            -
              type: vlan

```



```

    vlan_id: {get_param: StorageNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: StorageIpSubnet}
  -
    type: vlan
    vlan_id: {get_param: StorageMgmtNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: StorageMgmtIpSubnet}
  -
    type: vlan
    vlan_id: {get_param: ExternalNetworkVlanID}
    device: bond_api
    addresses:
      -
        ip_netmask: {get_param: ExternalIpSubnet}
    routes:
      -
        default: true
        next_hop: {get_param: ExternalInterfaceDefaultRoute}
  -
    type: ovs_bridge
    name: br-link0
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    members:
      -
        type: interface
        name: nic5
  -
    type: ovs_bridge
    name: br-link1
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    members:
      -
        type: interface
        name: nic6

```

```

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

```

### B.2.5. 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
  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
              defroute: false
            -
              type: interface
              name: nic2
              use_dhcp: false
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                  - {get_param: ControlPlaneIp}
                  - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
            -
              type: linux_bond
              name: bond_api
              members:
                -
                  type: interface
                  name: nic3
                -
                  type: interface
                  name: nic4

```

```

-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: vlan
  vlan_id: {get_param: TenantNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: TenantIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: bond_api
  addresses:
    -
      ip_netmask: {get_param: StorageIpSubnet}
-
  type: ovs_user_bridge
  name: br-link0
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  members:
    -
      type: ovs_dpdk_port
      name: dpdk0
      members:
        -
          type: interface
          name: nic5
-
  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: nic6

```

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. VLAN OVS-DPDK データプレーンボンディング YAML ファイルのサ ンプル

### 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: ""
  HostCpusList:
    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: ""
  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 processes (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 and 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
                        yum install -y tuned-profiles-cpu-partitioning

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

```

```

        fi
        tuned-adm profile cpu-partitioning
    fi
fi
params:
    $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}
    $TUNED_CORES: {get_param: HostIsolatedCoreList}

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

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

```

### B.3.2. 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.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: 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: '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

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: "'4,6,20,22'"
# 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"

# 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: "8,10,12,14,18,24,26,28,30"
# 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: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"
# List of logical cores to be used by ovs-dpdk processes (dpdk-lcore-
mask)
HostCpusList: "'3,5,7,19,21,23'"

```

### 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
              defroute: false
            -
              type: linux_bond
              name: bond1
              bonding_options: "mode=active-backup"
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              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}
          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
      defroute: false
    -
      type: interface
      name: nic2
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      addresses:
        -
          ip_netmask:
            list_join:
              - '/'
              - - {get_param: ControlPlaneIp}
                - {get_param: ControlPlaneSubnetCidr}
      routes:
        -
          ip_netmask: 169.254.169.254/32
          next_hop: {get_param: EC2MetadataIp}
        -
          default: true
          next_hop: {get_param: ControlPlaneDefaultRoute}
    -
      type: linux_bond
      name: bond_api
      bonding_options: "mode=active-backup"
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      members:
        -
          type: interface
          name: nic3
          primary: true
        -
          type: interface
          name: nic4
    -
      type: vlan
      vlan_id: {get_param: InternalApiNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: InternalApiIpSubnet}
    -
      type: vlan
      vlan_id: {get_param: TenantNetworkVlanID}
      device: bond_api
      addresses:
        -
          ip_netmask: {get_param: TenantIpSubnet}
    -
      type: 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: nic5
          -
            type: ovs_dpdk_port
            name: dpdk1
            members:
              -
                type: interface
                name: nic6

  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. VXLAN OVS-DPDK データプレーンボンディング YAML ファイルのサ ンプル

### 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 separated 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: ""
  HostCpusList:
    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: ""
  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
            yum install -y tuned-profiles-cpu-partitioning

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

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

```



```

        sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/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}

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

```

#### B.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
    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::Triple0::Compute::Net::SoftwareConfig: nic-configs/compute.yaml
    OS::Triple0::Controller::Net::SoftwareConfig: nic-
configs/controller.yaml
    OS::Triple0::NodeUserData: first-boot.yaml
    OS::Triple0::NodeExtraConfigPost: post-install.yaml
    OS::Triple0::AllNodes::Validation: dummy_all_nodes-validation.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: '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

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: "'4,6,20,22'"
# 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"

# 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: "8,10,12,14,18,24,26,28,30"
# 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
intel_iommu=on"
# A list or range of physical CPU cores to be tuned.
# The given args will be appended to the tuned cpu-partitioning profile.
HostIsolatedCoreList: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"
# List of logical cores to be used by ovs-dpdk processes (dpdk-lcore-
mask)
HostCpusList: "'3,5,7,19,21,23'"

```

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

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              defroute: false

```

```

-
  type: linux_bond
  name: bond_api
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  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}
  members:
    -
      type: interface
      name: nic2
    -
      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: 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.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
              defroute: false
            -
              type: linux_bond
              name: bond_provision
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              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}
          members:
            -
              type: interface
              name: nic2
            -

```



```

        type: interface
        name: nic3
    -
        type: linux_bond
        name: bond_api
        use_dhcp: false
        dns_servers: {get_param: DnsServers}
        members:
            -
                type: interface
                name: nic4
    -
        type: vlan
        vlan_id: {get_param: InternalApiNetworkVlanID}
        device: bond_api
        addresses:
            -
                ip_netmask: {get_param: InternalApiIpSubnet}
    -
        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: nic5

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. OVS-DPDK および SR-IOV コンポーザブルロールの YAML ファイルのサンプル

### B.5.1. roles\_data.yaml

```
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephMds
    - OS::Triple0::Services::CephMon
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::CephRbdMirror
    - OS::Triple0::Services::CephRgw
    - OS::Triple0::Services::CinderApi
    - OS::Triple0::Services::CinderBackup
    - OS::Triple0::Services::CinderScheduler
    - OS::Triple0::Services::CinderVolume
    - OS::Triple0::Services::Congress
    - OS::Triple0::Services::Kernel
    - OS::Triple0::Services::Keystone
    - OS::Triple0::Services::GlanceApi
    - OS::Triple0::Services::HeatApi
    - OS::Triple0::Services::HeatApiCfn
    - OS::Triple0::Services::HeatApiCloudwatch
    - OS::Triple0::Services::HeatEngine
    - OS::Triple0::Services::MySQL
    - OS::Triple0::Services::MySQLClient
    - OS::Triple0::Services::NeutronDhcpAgent
    - OS::Triple0::Services::NeutronL3Agent
    - OS::Triple0::Services::NeutronMetadataAgent
    - OS::Triple0::Services::NeutronApi
    - OS::Triple0::Services::NeutronCorePlugin
    - OS::Triple0::Services::NeutronOvsAgent
    - OS::Triple0::Services::RabbitMQ
    - OS::Triple0::Services::HAproxy
    - OS::Triple0::Services::Keepalived
    - OS::Triple0::Services::Memcached
    - OS::Triple0::Services::Pacemaker
    - OS::Triple0::Services::Redis
    - OS::Triple0::Services::NovaConductor
    - OS::Triple0::Services::MongoDb
    - OS::Triple0::Services::NovaApi
    - OS::Triple0::Services::NovaPlacement
    - OS::Triple0::Services::NovaMetadata
    - OS::Triple0::Services::NovaScheduler
    - OS::Triple0::Services::NovaConsoleauth
    - OS::Triple0::Services::NovaVncProxy
    - OS::Triple0::Services::Ec2Api
    - OS::Triple0::Services::Ntp
```

```

- OS::Triple0::Services::SwiftProxy
- OS::Triple0::Services::SwiftStorage
- OS::Triple0::Services::SwiftRingBuilder
- OS::Triple0::Services::Snmpp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::CeilometerApi
- OS::Triple0::Services::CeilometerCollector
- OS::Triple0::Services::CeilometerExpirer
- OS::Triple0::Services::CeilometerAgentCentral
- OS::Triple0::Services::CeilometerAgentNotification
- OS::Triple0::Services::Horizon
- OS::Triple0::Services::GnocchiApi
- OS::Triple0::Services::GnocchiMetricd
- OS::Triple0::Services::GnocchiStatsd
- OS::Triple0::Services::ManilaApi
- OS::Triple0::Services::ManilaScheduler
- OS::Triple0::Services::ManilaBackendGeneric
- OS::Triple0::Services::ManilaBackendNetapp
- OS::Triple0::Services::ManilaBackendCephFs
- OS::Triple0::Services::ManilaShare
- OS::Triple0::Services::AodhApi
- OS::Triple0::Services::AodhEvaluator
- OS::Triple0::Services::AodhNotifier
- OS::Triple0::Services::AodhListener
- OS::Triple0::Services::SaharaApi
- OS::Triple0::Services::SaharaEngine
- OS::Triple0::Services::IronicApi
- OS::Triple0::Services::IronicConductor
- OS::Triple0::Services::NovaIronic
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::OpenDaylightApi
- OS::Triple0::Services::OpenDaylightOvs
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::FluentdClient
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::BarbicanApi
- OS::Triple0::Services::PankoApi
- OS::Triple0::Services::Tacker
- OS::Triple0::Services::Zaqar
- OS::Triple0::Services::OVNDBs
- OS::Triple0::Services::NeutronML2FujitsuCfab
- OS::Triple0::Services::NeutronML2FujitsuFossw
- OS::Triple0::Services::CinderHPELeftHandISCSI
- OS::Triple0::Services::Etcd
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::OctaviaApi
- OS::Triple0::Services::OctaviaHealthManager
- OS::Triple0::Services::OctaviaHousekeeping
- OS::Triple0::Services::OctaviaWorker

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

```

```

ServicesDefault:
  - OS::Triple0::Services::CACerts
  - OS::Triple0::Services::CephClient
  - OS::Triple0::Services::CephExternal
  - OS::Triple0::Services::Timezone
  - OS::Triple0::Services::Ntp
  - OS::Triple0::Services::Snmp
  - OS::Triple0::Services::Sshd
  - OS::Triple0::Services::NovaCompute
  - OS::Triple0::Services::NovaLibvirt
  - OS::Triple0::Services::Kernel
  - OS::Triple0::Services::ComputeNeutronCorePlugin
  - OS::Triple0::Services::ComputeNeutronOvsAgent
  - OS::Triple0::Services::ComputeCeilometerAgent
  - OS::Triple0::Services::ComputeNeutronL3Agent
  - OS::Triple0::Services::ComputeNeutronMetadataAgent
  - OS::Triple0::Services::TripleoPackages
  - OS::Triple0::Services::TripleoFirewall
  - OS::Triple0::Services::NeutronSriovAgent
  - OS::Triple0::Services::OpenDaylightOvs
  - OS::Triple0::Services::SensuClient
  - OS::Triple0::Services::FluentdClient
  - OS::Triple0::Services::AuditD
  - OS::Triple0::Services::Collectd

- name: ComputeOvsDpdk
  CountDefault: 1
  HostnameFormatDefault: 'compute-ovs-dpdk-%index%'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephClient
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::Timezone
    - OS::Triple0::Services::Ntp
    - OS::Triple0::Services::Snmp
    - OS::Triple0::Services::Sshd
    - OS::Triple0::Services::NovaCompute
    - OS::Triple0::Services::NovaLibvirt
    - OS::Triple0::Services::Kernel
    - OS::Triple0::Services::ComputeNeutronCorePlugin
    - OS::Triple0::Services::ComputeNeutronOvsDpdkAgent
    - OS::Triple0::Services::ComputeCeilometerAgent
    - OS::Triple0::Services::ComputeNeutronL3Agent
    - OS::Triple0::Services::ComputeNeutronMetadataAgent
    - OS::Triple0::Services::TripleoPackages
    - OS::Triple0::Services::TripleoFirewall
    - OS::Triple0::Services::OpenDaylightOvs
    - OS::Triple0::Services::SensuClient
    - OS::Triple0::Services::FluentdClient
    - OS::Triple0::Services::AuditD
    - OS::Triple0::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: ""
```

```
  HostCpusList:
```

```
    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: ""
```

```
  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
            yum install -y tuned-profiles-cpu-partitioning

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

compute_kernel_args:
  type: OS::Heat::SoftwareConfig
  properties:
    config:
      str_replace:
        template: |
          #!/bin/bash
          FORMAT=$COMPUTE_HOSTNAME_FORMAT
          if [[ -z $FORMAT ]] ; then
            FORMAT="compute" ;
          else

```

```

        # Assumption: only %index% and %stackname% are the variables
in Host name format
        FORMAT=$(echo $FORMAT | sed 's/\%index%\%/g' | sed
's/\%stackname%\%/g') ;
        fi
        if [[ $(hostname) == *$FORMAT* ]] ; then
            sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/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}

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

```

### B.5.3. 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}

```

#### 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: 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: ["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: '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: "'4,6,20,22'"
# 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: "8,10,12,14,18,24,26,28,30"
# Kernel arguments for Compute node
ComputeKernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32
iommu=pt intel_iommu=on"
# A list or range of physical CPU cores to be tuned.
# The given args will be appended to the tuned cpu-partitioning profile.
HostIsolatedCoreList: "2,4,6,8,10,12,14,18,20,22,24,26,28,30"
# List of logical cores to be used by ovs-dpdk processess (dpdk-lcore-
mask)
HostCpusList: "'3,5,7,19,21,23'"

```

### 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.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. 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

```