



# Red Hat OpenStack Platform 10

## 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 10 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](mailto: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 10 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

前書き .....	3
第1章 概要 .....	4
第2章 仮想ネットワークの SR-IOV サポートの設定 .....	5
第3章 ネットワーク向けの DPDK-ACCELERATED OPEN VSWITCH (OVS) の設定 .....	9
3.1. RED HAT OPENSTACK PLATFORM での OVS-DPDK の設定 .....	9
3.2. 設定のトラブルシューティング .....	13
第4章 詳細情報 .....	15
付録A YAML ファイルのサンプル .....	16
A.1. SR-IOV YAML ファイルのサンプル .....	16
A.1.1. network.environment.yaml .....	16
A.1.2. first-boot.yaml .....	19
A.1.3. controller.yaml .....	20
A.1.4. compute.yaml .....	23
A.1.5. overcloud_deploy.sh .....	25
A.2. OVS-DPDK YAML ファイルのサンプル .....	25
A.2.1. network-environment.yaml .....	26
A.2.2. first-boot.yaml .....	28
A.2.3. post-install.yaml .....	30
A.2.4. controller.yaml .....	32
A.2.5. compute.yaml .....	35
A.2.6. overcloud_deploy.sh .....	37



## 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。

本ガイドは、NFV デプロイメント向けに Red Hat OpenStack Platform 10 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ています。

## 第1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。

Red Hat OpenStack Platform 10 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 10 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。

**network-environment.yaml** およびホストテンプレートファイルは、アンダークラウドノードの `/usr/share/openstack-tripleo-heat-templates/` に配置されています。サンプルの参照ファイルについては、「[YAML ファイルのサンプル](#)」を参照してください。

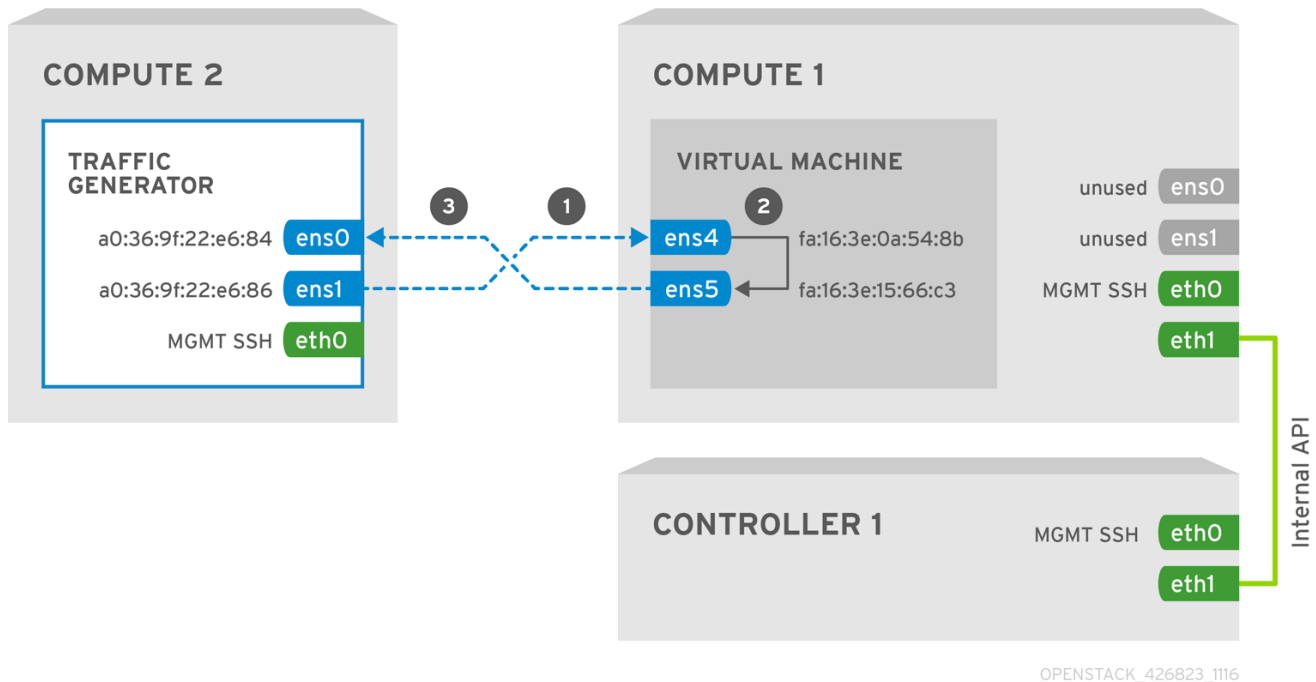
以下のセクションでは、Red Hat OpenStack Platform director を使用して、NFV 向けの SR-IOV および OVS-DPDK 機能を使用できるように **network-environment.yaml** とホストのテンプレートファイルを設定する方法をさらに詳しく説明しています。



## 第2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 10 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、**overcloud\_deploy.sh** スクリプトを実行し、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。



このセクションでは、SR-IOV を設定するためにオーバークラウドのデプロイ前に変更や更新を加える必要のある YAML ファイルについて説明しています。

1. **network-environment.yaml** ファイルを変更します。

- a. **first-boot.yaml** を追加して、カーネルの引数を設定します。

```
OS::TripleO::NodeUserData: /home/stack/templates/first-boot.yaml
```

- b. **ComputeKernelArgs** パラメーターをデフォルトの **grub** ファイルに追加します。

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



### 注記

「hw:mem\_page\_size=1GB」をそのフレーバーの仮想マシンフレーバーに追加します。

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

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

- d. コンピュートの **pci\_passthrough\_whitelist** パラメーターを設定して、**devname** を SR-IOV インターフェースとして設定します。

```
NovaPCIPassthrough:
  - devname: "p6p1"
    physical_network: "tenant"
```

- e. Red Hat OpenStack Platform 10 リリースの場合は、**devname** を使用するのではなく、Physical Function (PFs) の **Product** と **Vendor ID** を設定します。

```
- vendor_id: "8086"
  product_id: "154d"
  physical_network: "tenant"
```

以下の例では **tenant** を **physical\_network** の名前として使用します。

- f. 利用可能なスケジューラーフィルターを一覧表示します。

```
NovaSchedulerAvailableFilters:
  ["nova.scheduler.filters.all_filters", "nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter"]
```

- g. 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', 'ComputeCapabilitiesFilter', 'ImagePropertiesFilter', 'ServerGroupAntiAffinityFilter', 'ServerGroupAffinityFilter', 'PciPassthroughFilter']
```

- h. 仮想マシンプロセス用に確保する物理 CPU コアの一覧または範囲を表示します。コマンド区切りの CPU コア一覧を追加します。

```
NovaVcpuPinSet: ['4-12', '^8']
```

以下の例では、4-12 (8 以外) からコアを確保します。

- i. **VendorID:ProductID** の形式でサポートのある PCI ベンダーのデバイスを一覧表示します。デフォルトでは、Intel および Mellanox SR-IOV が有効な NIC がサポートされています。

```
NeutronSupportedPCIVendorDevs: ['15b3:1004', '8086:10ca']
```

- j. 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:p6p1"
```

- k. 各 SR-IOV インターフェース用に確保する Virtual Function (VF) の一覧を提供します。

```
NeutronSriovNumVFs: "p6p1:5"
```

- l. テナントネットワークのトンネルタイプを設定します (**vxlan** または **gre**)。トンネルタイプのパラメーターを無効にするには、`""` に値を設定します。

```
NeutronTunnelTypes: ""
```

- m. OpenStack Networking のテナントネットワークタイプを設定します。利用可能なオプションは **vlan** または **vxlan** です。デフォルトでは、この値は **vxlan** に設定されています。

```
NeutronNetworkType: 'vlan'
```

- n. Open vSwitch の論理ブリッジマッピングを物理ブリッジマッピングに設定します。

```
NeutronBridgeMappings: 'datacentre:br-ex,tenant:br-isolated'
```

ここでは **datacentre:br-ex,tenant:br-isolated** はブリッジに関連付けられた物理デバイスのことです。

- o. OpenStack Networking ML2 と Open vSwitch VLAN のマッピング範囲を設定します。

```
NeutronNetworkVLANRanges: 'datacentre:398:399,tenant:405:406'
```

ここでは **datacentre:398:399,tenant:405:406** は各物理デバイスの VLAN ID を表します。

Physnet (物理ネットワーク) は、**NeutronBridgeMapping** からのブリッジと **NeutronNetworkVLANRanges** からの VLAN ID を関連付ける値で、オーバークラウドにネットワークを作成する際に使用します。

- p. 調整を行う物理 CPU コアの範囲または一覧を設定します。  
指定の引数は、**tuned cpu-partitioning** プロファイルに追加されます。

```
HostCpusList: '4-8'
```

以下の例では、コア 4、5、6、7、8 を調整します。

2. **compute.yaml** ファイルを変更します。

以下を **compute.yaml** ファイルに追加して、SR-IOV インターフェースを設定します。

```
-
    type: interface
    name: p6p1
    use_dhcp: false
    defroute: false
```

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

以下の例は、VLAN 環境の **openstack overcloud deploy** コマンドを定義します。

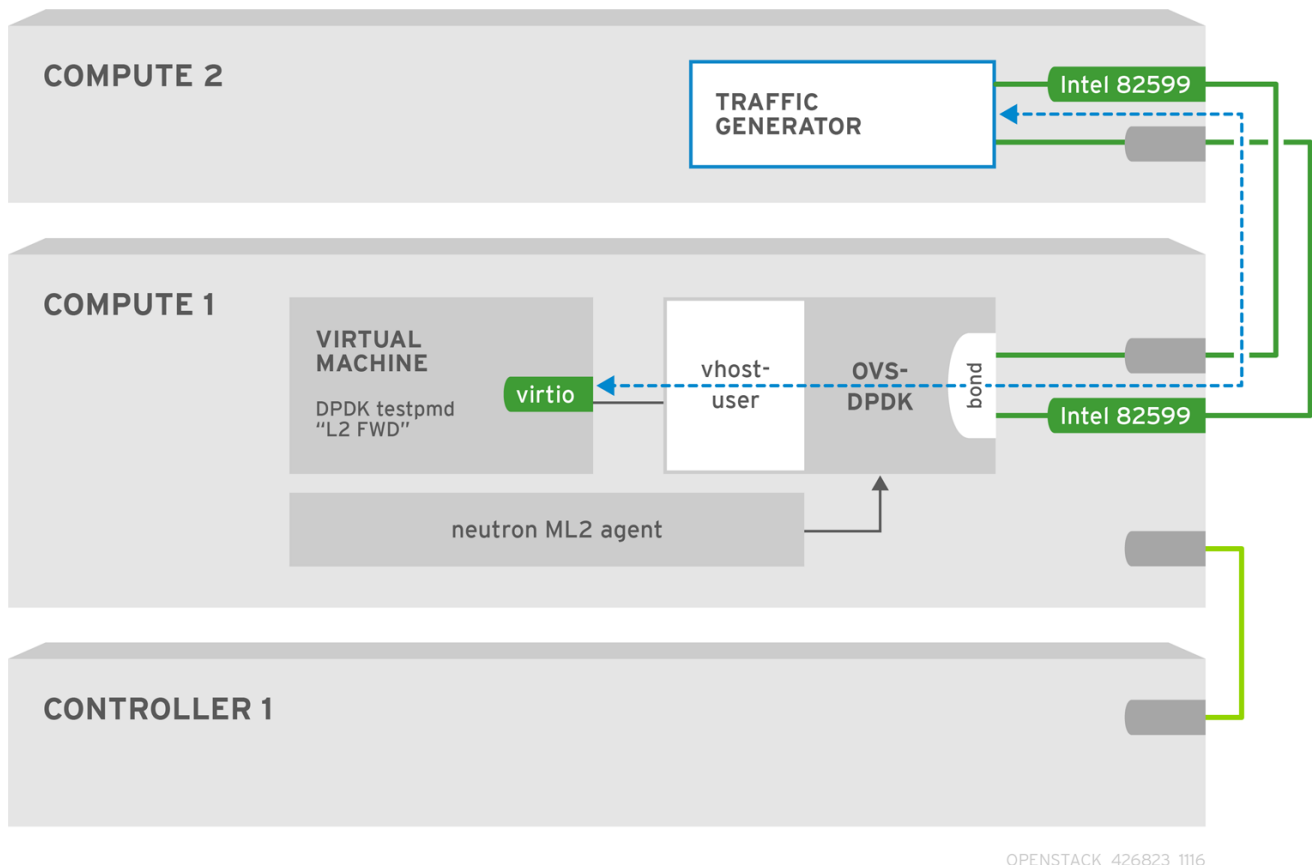
```
# openstack overcloud deploy --debug \  
--templates \  
--environment-file "$HOME/extra_env.yaml" \  
--libvirt-type kvm \  
--ntp-server clock.redhat.com \  
--control-scale 1 \  
--control-flavor baremetal \  
--compute-scale 1 \  
--compute-flavor baremetal \  
-e /home/stack/<relative-directory>/network/network-environment.yaml \  
\  
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-  
sriov.yaml \  
--log-file overcloud_install.log &> overcloud_install.log
```

- **/home/stack/<relative-directory>/network/network-environment.yaml** は **network-environment.yaml** ファイルのパスです。
- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml** は、デフォルトの **neutron-sriov.yaml** ファイル場所です。
- デフォルトの **neutron-sriov.yaml** 値は、**network-environment.yaml** ファイルで上書きできます。

## 第3章 ネットワーク向けの DPDK-ACCELERATED OPEN VSWITCH (OVS) の設定

本章では、Red Hat OpenStack Platform 環境内で DPDK を Open vSwitch とともにインストールしてチューニングする方法について説明します。

以下の手順では、**network-environment.yaml** ファイルにはカーネル引数と DPDK 引数のパラメーター、**compute.yaml** ファイルには DPDK インターフェースパラメーターのブリッジ、**controller.yaml** には同じブリッジ情報が含まれるように更新してから、**overcloud\_deploy.sh** スクリプトを実行して、DPDK パラメーターでオーバークラウドをデプロイする必要があります。



この手順を開始する前に、以下が用意されていることを確認してください。

- Red Hat Enterprise Linux 7.3 をベースとした Red Hat OpenStack Platform 10
- OVS-DPDK 2.5.0-14
- NIC: Dual port Intel x520



### 注記

OVS-DPDK は Red Hat Enterprise Linux に依存しています。Red Hat Enterprise Linux 7.3 では、OVS 2.5.\* のバージョンが必要です。

### 3.1. RED HAT OPENSTACK PLATFORM での OVS-DPDK の設定

本セクションでは、OpenStack 環境用に OVS-DPDK を設定してデプロイする手順を説明します。



## 注記

**NeutronDPDKCoreList** と **NeutronDPDKMemoryChannels** は、この手順の **必須** の設定です。適切な値なしに DPDK をデプロイしようとする、デプロイが失敗するか、デプロイが不安定になる可能性があります。

1. **network-environment.yaml** ファイルを変更します。

- a. **first-boot.yaml** を追加して、カーネルのパラメーターを設定します。

```
OS::TripleO::NodeUserData: /home/stack/templates/first-boot.yaml
```

- b. **ComputeKernelArgs** パラメーターをデフォルトの **grub** ファイルに追加します。

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



## 注記

以下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、本手順の後半に記載の **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** を減算した値であることを理解する必要があります。

DPDK インスタンスのフレーバーに **hw:mem\_page\_size=1GB** を追加する必要があります。これを行わない場合には、インスタンスは **DHCP** の割り当てを受けることができません。

- c. **post-install.yaml** ファイルを追加して、DPDK の引数を設定します。

```
OS::TripleO::NodeUserData: <relative directory>/post-install.yaml
```

- d. **[allowed\_pattern: '[0-9,-]+']** の形式で DPDK PMD として利用可能なコア一覧を指定します。

```
NeutronDpdkCoreList: "'2'"
```

- e. **[allowed\_pattern: "[0-9]+"]** の形式でメモリーチャンネルの番号を指定します。

```
NeutronDpdkMemoryChannels: "'2'"
```

- f. 各ソケットに割り当てるメモリーを設定します。

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

- g. DPDK ドライバー種別を設定します。デフォルト値は **vfio-pci** モジュールです。

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

- h. ホストのプロセス用にメモリーを確保します。

■

```
NovaReservedHostMemory: "4096"
```

- i. 仮想マシンプロセス用に確保する物理 CPU コアの一覧または範囲を追加します。

```
NovaVcpuPinSet: ['4-12','^8']
```

は、8 を除く 4-12 からのコアを確保します。

- j. 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。

コンピュータは、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"
```

- k. テナントネットワークのトンネルタイプを設定します (例: **vxlan** または **gre**)。トンネルタイプのパラメーターを無効にするには、**"** に値を設定します。

```
NeutronTunnelTypes: ""
```

- l. OpenStack Networking のテナントネットワークタイプを設定します。利用可能なオプションは **vlan** または **vxlan** です。

```
NeutronNetworkType: 'vlan'
```

- m. Open vSwitch の論理ブリッジマッピングを物理ブリッジマッピングに設定します。

```
NeutronBridgeMappings: 'datacentre: br-ex,dpdk:br-link '
```

ここでは **datacentre:br-ex,dpdk:br-link** はブリッジに関連付けられた物理デバイスです。

- n. OpenStack Networking ML2 と Open vSwitch VLAN のマッピング範囲を設定します。

```
NeutronNetworkVLANRanges: ' datacentre:22:22,dpdk:25:25 '
```

ここでは **datacentre:22:22,dpdk:25:25** の値は、各物理デバイスの VLAN ID を表します。

Physnet (物理ネットワーク) は、**NeutronBridgeMapping** からのブリッジと **NeutronNetworkVLANRanges** からの VLAN ID を関連付ける値で、オーバークラウドにネットワークを作成する際に使用します。

- o. 調整を行う物理 CPU コアの範囲または一覧を設定します。  
指定の引数は、**tuned cpu-partitioning** プロファイルに追加されます。

```
HostCpusList: '4-8'
```

以下の例では、コア 4、5、6、7、8 を調整します。

## 2. `compute.yaml` ファイルを変更します。

**compute.yaml** ファイルに以下の行を追加して、希望のインターフェースを使用した **dpdk** ポートでのブリッジを設定します。

```
-
  type: ovs_user_bridge
  name: br-link                                #<BRIDGE_NAME>
  use_dhcp: false
  members:
    -
      type: ovs_dpdk_port
      name: dpdk0
      members:
        -
          type: interface
          name: nic3                          # <SUPPORTED_INTERFACE_NAME>
```

### 3. **controller.yaml** ファイルを変更します。

以下の行を **controller.yaml** ファイルに追加して、同じ VLAN 設定のインターフェースを含む同じ名前のブリッジを設定します。

```
-
  type: ovs_bridge
  name: br-link                                #<BRIDGE_NAME>
  use_dhcp: false
  members:
    -
      type: interface
      name: nic4                              # <SUPPORTED_INTERFACE_NAME>
```

### 4. **overcloud\_deploy.sh** スクリプトを実行して OVS-DPDK でオーバークラウドをデプロイします。

```
# openstack overcloud deploy --debug \
--templates \
--environment-file "$HOME/extra_env.yaml" \
--libvirt-type kvm \
--ntp-server clock.redhat.com \
--control-scale 1 \
--control-flavor baremetal \
--compute-scale 1 \
--compute-flavor baremetal \
-e /home/stack/multiple-nic-vlans-ovs-dpdk-single-port/network-
environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-
ovs-dpdk.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

- **/home/stack/<relative directory>/network/network-environment.yaml** は **network-environment.yaml** ファイルのパスを設定します。
- **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-dpdk.yaml** はデフォルトの **neutron-ovs-dpdk.yaml** ファイルの場所を設定します。



- デフォルトの **neutron-ovs-dpdk.yaml** 値は、**network-environment.yaml** ファイルで上書きできます。



### 注記

OVS-DPDK の以下の設定は、セキュリティーグループやライブマイグレーションはサポートしません。

## 3.2. 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. ブリッジの設定をレビューして、ブリッジが **datapath\_type=netdev** で作成されたことを確認します。以下に例を示します。

```
# 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 dpdk_watchdog3
3165 OTHER 0      6 237634      0 vhost_thread1
3166 OTHER 0      6 3665      0 urcu2
```

## 第4章 詳細情報

以下の表では、参考資料として他の Red Hat のドキュメントを紹介しています。

Red Hat OpenStack Platform のドキュメントスイートは [Red Hat OpenStack Platform 10 の製品ドキュメント](#) から参照してください。

表 4.1. 利用可能なドキュメント一覧

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 コンポーネントと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 は基本の OpenStack インストールを <a href="#">アンダークラウド</a>として使用して、最終的な <a href="#">オーバークラウド</a>に OpenStack ノードをインストール、設定して、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、アンダークラウドのインストールには、追加のホストマシンが必要な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="#">Advanced Overcloud Customization</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="#">ネットワーク機能仮想化 (NFV) のプランニングガイド</a>』を参照してください。</p>

## 付録A YAML ファイルのサンプル

本項では、参考として YAML ファイルのサンプルを紹介します。

### A.1. SR-IOV YAML ファイルのサンプル

#### A.1.1. network.environment.yaml

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

  # Network isolation configuration
  # Service section
  # If some service should be disable, use the following example
  # OS::TripleO::Network::Management: OS::Heat::None
  OS::TripleO::Network::External: /usr/share/openstack-tripleo-heat-
  templates/network/external.yaml
  OS::TripleO::Network::InternalApi: /usr/share/openstack-tripleo-heat-
  templates/network/internal_api.yaml
  OS::TripleO::Network::Tenant: /usr/share/openstack-tripleo-heat-
  templates/network/tenant.yaml
  OS::TripleO::Network::Management: OS::Heat::None
  OS::TripleO::Network::StorageMgmt: OS::Heat::None
  OS::TripleO::Network::Storage: OS::Heat::None

  # Port assignments for the VIPs
  OS::TripleO::Network::Ports::ExternalVipPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/external.yaml
  OS::TripleO::Network::Ports::InternalApiVipPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/internal_api.yaml
  OS::TripleO::Network::Ports::RedisVipPort: /usr/share/openstack-tripleo-
  heat-templates/network/ports/vip.yaml
  OS::TripleO::Network::Ports::StorageVipPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Network::Ports::StorageMgmtVipPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/noop.yaml

  # Port assignments for the controller role
  OS::TripleO::Controller::Ports::ExternalPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/external.yaml
  OS::TripleO::Controller::Ports::InternalApiPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/internal_api.yaml
  OS::TripleO::Controller::Ports::TenantPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/tenant.yaml
  OS::TripleO::Controller::Ports::ManagementPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Controller::Ports::StoragePort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Controller::Ports::StorageMgmtPort: /usr/share/openstack-
  tripleo-heat-templates/network/ports/noop.yaml
```

```

# Port assignments for the compute role
OS::TripleO::Compute::Ports::ExternalPort: /usr/share/openstack-tripleo-
heat-templates/network/ports/external.yaml
OS::TripleO::Compute::Ports::InternalApiPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/internal_api.yaml
OS::TripleO::Compute::Ports::TenantPort: /usr/share/openstack-tripleo-
heat-templates/network/ports/tenant.yaml
OS::TripleO::Compute::Ports::ManagementPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml
OS::TripleO::Compute::Ports::StoragePort: /usr/share/openstack-tripleo-
heat-templates/network/ports/noop.yaml
OS::TripleO::Compute::Ports::StorageMgmtPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml

# Port assignments for service virtual IPs for the controller role
OS::TripleO::Controller::Ports::RedisVipPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/vip.yaml

# First boot and Kernel Args
OS::TripleO::NodeUserData: /home/stack/<relative-directory>/first-
boot.yaml

parameter_defaults:
# Customize all these values to match the local environment
InternalApiNetCidr: 10.10.10.0/24
TenantNetCidr: 10.10.2.0/24
ExternalNetCidr: 172.20.12.112/28
# CIDR subnet mask length for provisioning network
ControlPlaneSubnetCidr: '24'
InternalApiAllocationPools: [{'start': '10.10.10.10', 'end':
'10.10.10.200'}]
TenantAllocationPools: [{'start': '172.10.2.100', 'end':
'172.10.2.200'}]
# Use an External allocation pool which will leave room for floating IPs
ExternalAllocationPools: [{'start': '172.20.12.114', 'end':
'172.20.12.125'}]
# Set to the router gateway on the external network
ExternalInterfaceDefaultRoute: 172.20.12.126
# Gateway router for the provisioning network (or Undercloud IP)
ControlPlaneDefaultRoute: 192.0.2.254
# Generally the IP of the Undercloud
EC2MetadataIp: 192.0.2.1
InternalApiNetworkVlanID: 10
TenantNetworkVlanID: 11
ExternalNetworkVlanID: 12
# Define the DNS servers (maximum 2) for the overcloud nodes
DnsServers: ["8.8.4.4","8.8.8.8"]
# May set to br-ex if using floating IPs only on native VLAN on bridge
br-ex
NeutronExternalNetworkBridge: ''
# The tunnel type for the tenant network (vxlan or gre). Set to '' to
disable tunneling.
NeutronTunnelTypes: ''
# The tenant network type for Neutron (vlan or vxlan).
NeutronNetworkType: 'vlan'

```

```

# The OVS logical->physical bridge mappings to use.
NeutronBridgeMappings: 'datacentre:br-ex,sriov:br-sriov'
# The Neutron ML2 and OpenVSwitch vlan mapping range to support.
NeutronNetworkVLANRanges: 'datacentre:22:22,sriov:25:25'
# Nova flavor to use.
OvercloudControlFlavor: baremetal
OvercloudComputeFlavor: baremetal
# Number of nodes to deploy.
ControllerCount: 1
ComputeCount: 1

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

#####
# SRIOV configuration #
#####
# The mechanism drivers for the Neutron tenant network.
NeutronMechanismDrivers: "openvswitch,sriovnicswitch"
# List of PCI Passthrough whitelist parameters.
# Use ONE of the following examples.
# Example 1:
# NovaPCIPassthrough:
#   - vendor_id: "8086"
#     product_id: "154c"
#     address: "0000:05:00.0" - (optional)
#     physical_network: "datacentre"
#
# Example 2:
# NovaPCIPassthrough:
#   - devname: "p6p1"
#     physical_network: "tenant"
NovaPCIPassthrough:
  - devname: "p6p1"
    physical_network: "sriov"
  - devname: "p6p2"
    physical_network: "sriov"
# 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: "sriov:p6p1,sriov:p6p2"
# 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: "p6p1:5,p6p2:5"

# List of scheduler available filters
NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthro
ugh_filter.PciPassthroughFilter"]

```

```

# An array of filters used by Nova to filter a node. These filters will
be applied in the order they are listed,
# so place your most restrictive filters first to make the filtering
process more efficient.
NovaSchedulerDefaultFilters:
['AvailabilityZoneFilter', 'RamFilter', 'ComputeFilter', 'ComputeCapabilities
Filter', 'ImagePropertiesFilter', 'ServerGroupAntiAffinityFilter', 'ServerGro
upAffinityFilter', 'PciPassthroughFilter']
# Kernel arguments for Compute node
ComputeKernelArgs: "intel_iommu=on default_hugepagesz=1GB hugepagesz=1G
hugepages=12"

```

### 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: ""

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

  # Verify the logs on /var/log/cloud-init.log on the overcloud node
  compute_kernel_args:
    type: OS::Heat::SoftwareConfig
    properties:
      config:
        str_replace:
          template: |
            #!/bin/bash
            set -x
            FORMAT=$COMPUTE_HOSTNAME_FORMAT
            if [[ -z $FORMAT ]] ; then
              FORMAT="compute" ;
            else

```

```

        # Assumption: only %index% and %stackname% are the variables
in Host name format
        FORMAT=$(echo $FORMAT | sed 's/\%index%\%/g' | sed
's/\%stackname%\%/g') ;
        fi
        if [[ $(hostname) == *$FORMAT* ]] ; then
            sed 's/^\(GRUB_CMDLINE_LINUX=".*\)"/\1 $KERNEL_ARGS"/g' -i
/etc/default/grub ;
            grub2-mkconfig -o /etc/grub2.cfg

            # IP not acquired, give some time for dhcp_all_interface
            sleep 5

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

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

```

### A.1.3. controller.yaml

```

heat_template_version: 2015-04-30

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

parameters:
    ControlPlaneIp:
        default: ''
        description: IP address/subnet on the ctlplane network
        type: string
    ExternalIpSubnet:
        default: ''
        description: IP address/subnet on the external network
        type: string
    InternalApiIpSubnet:
        default: ''
        description: IP address/subnet on the internal API network
        type: string
    StorageIpSubnet:
        default: ''
        description: IP address/subnet on the storage network

```



```

    type: string
StorageMgmtIpSubnet:
  default: ''
  description: IP address/subnet on the storage mgmt network
  type: string
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: ovs_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}
    members:
      -
        type: interface
        name: nic2
        # force the MAC address of the bridge to this interface
        primary: true
      -
        type: vlan
        vlan_id: {get_param: InternalApiNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: InternalApiIpSubnet}
      -
        type: vlan
        vlan_id: {get_param: TenantNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: TenantIpSubnet}
  -
    type: ovs_bridge
    name: br-ex
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    members:
      -
        type: interface
        name: nic3
        # force the MAC address of the bridge to this interface
      -
        type: vlan
        vlan_id: {get_param: ExternalNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: ExternalIpSubnet}
        routes:
          -
            default: true
            next_hop: {get_param:
ExternalInterfaceDefaultRoute}
  -
    type: ovs_bridge

```

```

        name: br-sriov
        use_dhcp: false
        members:
          -
            type: interface
            name: nic4

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

```

#### A.1.4. compute.yaml

```

heat_template_version: 2015-04-30

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

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including
environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  InternalApiNetworkVlanID:
    default: ''
    description: Vlan ID for the internal_api network traffic.
    type: number
  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: ovs_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}
                -
                  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}
        addresses:
          -
            ip_netmask: {get_param: InternalApiIpSubnet}
      -
        type: vlan
        vlan_id: {get_param: TenantNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: TenantIpSubnet}
    -
      type: interface
      name: nic3
      use_dhcp: false
      defroute: false
    -
      type: interface
      name: nic4
      use_dhcp: false
      defroute: false

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

```

### A.1.5. overcloud\_deploy.sh

```

#!/bin/bash

cat >"$HOME/extra_env.yaml"<<EOF
---
parameter_defaults:
  Debug: true
EOF

openstack overcloud deploy --debug \
--templates \
--environment-file "$HOME/extra_env.yaml" \
--libvirt-type kvm \
--ntp-server clock.redhat.com \
-e /home/stack/<relative-directory>/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-
sriov.yaml \
--log-file overcloud_install.log &> overcloud_install.log

```

## A.2. OVS-DPDK YAML ファイルのサンプル

## A.2.1. network-environment.yaml

```

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

  # Network isolation configuration
  # Service section
  # If some service should be disable, use the following example
  # OS::TripleO::Network::Management: OS::Heat::None
  OS::TripleO::Network::External: /usr/share/openstack-tripleo-heat-
templates/network/external.yaml
  OS::TripleO::Network::InternalApi: /usr/share/openstack-tripleo-heat-
templates/network/internal_api.yaml
  OS::TripleO::Network::Tenant: /usr/share/openstack-tripleo-heat-
templates/network/tenant.yaml
  OS::TripleO::Network::Management: OS::Heat::None
  OS::TripleO::Network::StorageMgmt: OS::Heat::None
  OS::TripleO::Network::Storage: OS::Heat::None

  # Port assignments for the VIPs
  OS::TripleO::Network::Ports::ExternalVipPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/external.yaml
  OS::TripleO::Network::Ports::InternalApiVipPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/internal_api.yaml
  OS::TripleO::Network::Ports::RedisVipPort: /usr/share/openstack-tripleo-
heat-templates/network/ports/vip.yaml
  OS::TripleO::Network::Ports::StorageVipPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Network::Ports::StorageMgmtVipPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml

  # Port assignments for the controller role
  OS::TripleO::Controller::Ports::ExternalPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/external.yaml
  OS::TripleO::Controller::Ports::InternalApiPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/internal_api.yaml
  OS::TripleO::Controller::Ports::TenantPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/tenant.yaml
  OS::TripleO::Controller::Ports::ManagementPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Controller::Ports::StoragePort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Controller::Ports::StorageMgmtPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml

  # Port assignments for the compute role
  OS::TripleO::Compute::Ports::ExternalPort: /usr/share/openstack-tripleo-
heat-templates/network/ports/external.yaml
  OS::TripleO::Compute::Ports::InternalApiPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/internal_api.yaml

```

```

    OS::Triple0::Compute::Ports::TenantPort: /usr/share/openstack-tripleo-
heat-templates/network/ports/tenant.yaml
    OS::Triple0::Compute::Ports::ManagementPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml
    OS::Triple0::Compute::Ports::StoragePort: /usr/share/openstack-tripleo-
heat-templates/network/ports/noop.yaml
    OS::Triple0::Compute::Ports::StorageMgmtPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/noop.yaml

    # Port assignments for service virtual IPs for the controller role
    OS::Triple0::Controller::Ports::RedisVipPort: /usr/share/openstack-
tripleo-heat-templates/network/ports/vip.yaml

parameter_defaults:
    # Customize all these values to match the local environment
    InternalApiNetCidr: 10.10.10.0/24
    TenantNetCidr: 10.10.2.0/24
    ExternalNetCidr: 172.20.12.112/28
    # CIDR subnet mask length for provisioning network
    ControlPlaneSubnetCidr: '24'
    InternalApiAllocationPools: [{'start': '10.10.10.10', 'end':
'10.10.10.200'}]
    TenantAllocationPools: [{'start': '172.10.2.100', 'end':
'172.10.2.200'}]
    # Use an External allocation pool which will leave room for floating IPs
    ExternalAllocationPools: [{'start': '172.20.12.114', 'end':
'172.20.12.125'}]
    # Set to the router gateway on the external network
    ExternalInterfaceDefaultRoute: 172.20.12.126
    # Gateway router for the provisioning network (or Undercloud IP)
    ControlPlaneDefaultRoute: 192.0.2.254
    # Generally the IP of the Undercloud
    EC2MetadataIp: 192.0.2.1
    InternalApiNetworkVlanID: 10
    TenantNetworkVlanID: 11
    ExternalNetworkVlanID: 12
    # Define the DNS servers (maximum 2) for the overcloud nodes
    DnsServers: ["8.8.4.4", "8.8.8.8"]
    # May set to br-ex if using floating IPs only on native VLAN on bridge
br-ex
    NeutronExternalNetworkBridge: ""
    # The tunnel type for the tenant network (vxlan or gre). Set to '' to
disable tunneling.
    NeutronTunnelTypes: ''
    # The tenant network type for Neutron (vlan or vxlan).
    NeutronNetworkType: 'vlan'
    # The OVS logical->physical bridge mappings to use.
    NeutronBridgeMappings: 'datacentre:br-ex,dpdk0:br-link0,dpdk1:br-link1'
    # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
    NeutronNetworkVLANRanges: 'datacentre:22:22,dpdk0:25:25,dpdk1:26:26'
    # Nova flavor to use.
    OvercloudControlFlavor: baremetal
    OvercloudComputeFlavor: baremetal
    # Number of nodes to deploy.
    ControllerCount: 1
    ComputeCount: 1

```

```

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

#####
# OVS DPDK configuration #
#####
## NeutronDpdkCoreList and NeutronDpdkMemoryChannels are REQUIRED
settings.
## Attempting to deploy DPDK without appropriate values will cause
deployment to fail or lead to unstable deployments.
# List of cores to be used for DPDK Poll Mode Driver
NeutronDpdkCoreList: "'2'"
# Number of memory channels to be used for DPDK
NeutronDpdkMemoryChannels: "4"
# NeutronDpdkSocketMemory
NeutronDpdkSocketMemory: "1024"
# NeutronDpdkDriverType
NeutronDpdkDriverType: "vfio-pci"

# 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: "3-15"
# 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.2.2. 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: ""

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

# 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 [ -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

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

        sleep 5
        reboot

    fi
    params:
        $KERNEL_ARGS: {get_param: ComputeKernelArgs}
        $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

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

```

### A.2.3. post-install.yaml

```

heat_template_version: 2014-10-16

description: >
    Example extra config for post-deployment

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

```

```

resources:
  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers: {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE'] # Only do this on CREATE

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: |
            #!/bin/bash
            core_mask=''
            get_core_mask()
            {
              list=$1
              declare -a bm
              bm=(0 0 0 0 0 0 0 0 0 0)
              max_idx=0
              for core in $(echo $list | sed 's/,/ /g')
              do
                index=$((core/32))
                temp=$((1<<core))
                bm[$index]=$(( ${bm[$index]} | $temp ))
                if [ $max_idx -lt $index ]; then
                  max_idx=$index
                fi
              done

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

            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
            get_core_mask $DPDK_PMD_CORES
            ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-
mask=$core_mask
            sed -ri '/^DPDK_OPTIONS/s/-l [0-9\,]+ /-l 0 /'
/etc/sysconfig/openvswitch
            systemctl daemon-reload
            systemctl restart openvswitch
        fi
    params:
        $DPDK_PMD_CORES: {get_param: NeutronDpdkCoreList}
        $COMPUTE_HOSTNAME_FORMAT: {get_param: ComputeHostnameFormat}

```

#### A.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
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: ovs_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}
            members:
              -
                type: interface

```

```

        name: nic2
        # force the MAC address of the bridge to this interface
        primary: true
      -
        type: vlan
        vlan_id: {get_param: InternalApiNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: InternalApiIpSubnet}
      -
        type: vlan
        vlan_id: {get_param: TenantNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: TenantIpSubnet}
    -
      type: ovs_bridge
      name: br-ex
      use_dhcp: false
      dns_servers: {get_param: DnsServers}
      members:
        -
          type: interface
          name: nic3
        -
          type: vlan
          vlan_id: {get_param: ExternalNetworkVlanID}
          addresses:
            -
              ip_netmask: {get_param: ExternalIpSubnet}
          routes:
            -
              default: true
              next_hop: {get_param:
ExternalInterfaceDefaultRoute}
      -
        type: ovs_bridge
        name: br-link0
        use_dhcp: false
        members:
          -
            type: interface
            name: nic4
      -
        type: ovs_bridge
        name: br-link1
        use_dhcp: false
        members:
          -
            type: interface
            name: nic5

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

```

### A.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
  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: ovs_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}
                -
                  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}
                addresses:
                  -
                    ip_netmask: {get_param: InternalApiIpSubnet}
              -
                type: vlan
                vlan_id: {get_param: TenantNetworkVlanID}
                addresses:

```



```

        -
            ip_netmask: {get_param: TenantIpSubnet}
    -
        type: ovs_user_bridge
        name: br-link0
        use_dhcp: false
        members:
            -
                type: ovs_dpdk_port
                name: dpdk0
                members:
                    -
                        type: interface
                        name: nic3
    -
        type: ovs_user_bridge
        name: br-link1
        use_dhcp: false
        members:
            -
                type: ovs_dpdk_port
                name: dpdk1
                members:
                    -
                        type: interface
                        name: nic4

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

```

#### A.2.6. overcloud\_deploy.sh

```

#!/bin/bash

cat >"$HOME/extra_env.yaml"<<EOF
---
parameter_defaults:
    Debug: true
EOF

openstack overcloud deploy --debug \
--templates \
--environment-file "$HOME/extra_env.yaml" \
--libvirt-type kvm \
--ntp-server clock.redhat.com \
-e /home/stack/<relative-directory>/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ovs-
dpdk.yaml \
--log-file overcloud_install.log &> overcloud_install.log

```

