Red Hat OpenStack Platform 13

网络功能虚拟化规划和配置指南

规划和配置网络功能虚拟化(NFV) OpenStack 部署

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 网络功能虚拟化规划和配置指南

规划和配置网络功能虚拟化(NFV) OpenStack 部署

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Network_Functions_Virtualization_Planning_and_Configuration_Guide.ent file |.

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, the Red Hat 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 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.

摘要

本指南包含重要的规划信息，并描述了在 Red Hat OpenStack Platform 部署中用于网络功能虚拟化基础架构(NFVi)的单一根输入/输出虚拟化(SR-IOV)和数据平面开发套件(DPDK)的配置程序。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。告诉我们我们如何使其更好。

使用直接文档反馈(DDF)功能

使用 **添加反馈** DDF 函数进行具体句子、段落或代码块的直接评论。

1. 以 *Multi-page HTML* 格式查看文档。
2. 确保您在文档右上角看到 **反馈** 按钮。
3. 用鼠标指针高亮显示您想评论的文本部分。
4. 单击**添加反馈**。
5. 将 **添加反馈** 字段填写您的意见。
6. 可选：添加您的电子邮件地址，以便文档团队可以与您联系以获取您的问题。
7. 点 **Submit**。

第 1 章 网络功能虚拟化概述

网络功能虚拟化(NFV)是一种软件解决方案，它根据基于云的通用基础架构对网络功能进行虚拟化。使用 NFV 时，通信服务提供商可以从传统硬件中移出。

有关 NFV 概念的高级概述，请参阅 网络功能虚拟化产品指南。

注意

OVS-DPDK 和 SR-IOV 配置取决于您的硬件和拓扑。本指南提供了 CPU 分配、内存分配和 NIC 配置可能因拓扑和用例而异的示例。

使用 Red Hat OpenStack Platform director 隔离特定的网络类型，如外部、项目、内部 API 等。您可以在一个网络接口上部署网络，或者通过多个主机网络接口进行分发。通过 Open vSwitch，您可以通过为单个网桥分配多个接口来创建绑定。使用模板文件在 Red Hat OpenStack Platform 安装中配置网络隔离。如果没有提供模板文件，则服务网络会在 provisioning 网络上部署。模板配置文件有两种类型：

network-environment.yaml

此文件包含用于 overcloud 节点的网络详细信息，如子网和 IP 地址范围。此文件还包含不同的设置，用于覆盖不同场景的默认值。

compute.yaml 和 controller.yaml

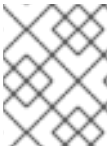
这些文件包含 overcloud 节点的主机网络接口配置。

host-config-and-reboot.yaml

此文件替换了已弃用的 **first-boot.yaml** 文件，并包含主机安装的配置。

这些 heat 模板文件位于 undercloud 节点上的 **/usr/share/openstack-tripleo-heat-templates/**。

硬件要求和软件要求部分提供了有关如何使用 Red Hat OpenStack Platform director 为 NFV 计划和配置 heat 模板文件的更多详细信息。



注意

您可以使用 YAML 文件概述 NFV 配置。如需有关 YAML 文件格式的更多信息，请参阅 Nutshell 中的 YAML

第 2 章 硬件要求

本节介绍 NFV 所需的硬件详细信息。

您可以通过选择类别并选择产品版本，使用红帽技术 生态系统 来检查认证硬件、软件、云提供商、组件列表。

有关 Red Hat OpenStack Platform 认证硬件的完整列表，请参阅 Red Hat OpenStack Platform 认证硬件。

2.1. 网络适配器支持

如需 NFV 测试的 NIC 列表，登录到客户门户网站的 Network Adapter 支持 页面。

如果您在 Mellanox ConnectX-4 或 ConnectX-5 网络接口上配置 OVS-DPDK，您必须在 **compute-ovs-dpdk.yaml** 文件中设置对应的内核驱动程序：

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

2.2. 发现 NUMA 节点拓扑

在计划部署时，您必须了解 Compute 节点的 NUMA 拓扑，以便可以对 CPU 和内存资源进行分区以获得最佳性能。要确定 NUMA 信息，您可以选择以下选项之一：

- 启用硬件内省以从裸机节点中检索 NUMA 信息。
- 登录到每个裸机节点以手动收集信息。

注意

您必须安装并配置 undercloud，然后才能通过硬件内省来检索 NUMA 信息。如需更多信息，请参阅 *Director 安装和使用* 指南。

检索硬件内省详细信息

裸机服务硬件检查多余(**inspection_extras**)默认情况下为检索硬件详情启用。您可以使用这些硬件详情来配置 overcloud。有关 **undercloud.conf** 文件中的 **inspection_extras** 参数的更多信息，请参阅 *Director 安装和使用 指南中的配置* Director。

例如，**numa_topology** 收集程序是这些硬件检查额外的部分，包括每个 NUMA 节点的以下信息：

- RAM（单位为 KB）
- 物理 CPU 内核数和同级线程数
- 和 NUMA 节点关联的 NIC

使用 **openstack baremetal introspection data save _UUID_ | jq .numa_topology** 命令，使用裸机节点的 *UUID* 检索此信息。

以下示例显示获取的裸机节点 NUMA 信息：

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

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

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

```
      "name": "ens1f0",
      "numa_node": 0
    },
    {
      "name": "eno4",
      "numa_node": 0
    },
    {
      "name": "eno1",
      "numa_node": 0
    },
    {
      "name": "eno3",
      "numa_node": 0
    },
    {
      "name": "eno2",
      "numa_node": 0
    }
  ]
}
```

2.3. 查看 BIOS 设置

以下列表描述了 NFV 所需的 BIOS 设置：

- **C3 Power State** ：Disabled.
- **C6 Power State** ：Disabled.
- **MLC Streamer** ：已启用.
- **MLC Spacial Prefetcher**: Enabled。
- **DCU Data Prefetcher**: Enabled。
- **DCA** ：已启用.
- **CPU Power 和性能** ：性能.
- **内存 RAS 和 Performance Config** → **NUMA Optimized**: Enabled。
- **turbo Boost** ：Disabled.
- **VT-d** ：如果需要 VFIO 功能，则已启用 Intel 卡。

2.4. 网络适配器快速数据路径特性支持矩阵

如需受支持版本的 FDP 列表，请登录客户门户网站的 快速数据路径 页面。

第 3 章 软件要求

本节介绍 NFV 所需的受支持配置和驱动程序以及订阅详情。

3.1. 注册并启用软件仓库

要安装 Red Hat OpenStack Platform，您必须使用 Red Hat Subscription Manager 注册 Red Hat OpenStack Platform director，并订阅所需的频道。详情请查看 注册您的系统。

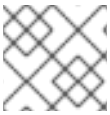
流程

1. 禁用默认存储库。

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

2. 通过网络功能虚拟化(NFV)为 Red Hat OpenStack Platform 启用所需的存储库。

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



注意

要注册 overcloud 节点，请参阅 Overcloud 注册。

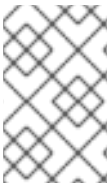
3.2. NFV 部署支持的配置

Red Hat OpenStack Platform 支持使用 director 的以下网络功能虚拟化(NFV)部署：

- 单根 I/O 虚拟化(SR-IOV)
- Open vSwitch 带有 Data Plane Development Kit (OVS-DPDK)

另外，您还可使用以下功能部署 Red Hat OpenStack Platform：

- 可组合角色
- 超融合基础架构 （支持限）
- 配置实时计算
- OVS 硬件卸载 （技术预览）



注意

红帽的嵌入式 OpenDaylight SDN 解决方案已在 OpenStack Platform (OSP) 14 中弃用。红帽将继续为 OpenDaylight 提供支持和程序错误修复，所有支持均以 OSP 13 生命周期结束(2021 年 6 月 27 日)。

3.3. 支持的驱动程序

有关支持的驱动程序的完整列表，请参阅 Red Hat OpenStack Platform 中的组件、插件和驱动程序支持。

有关使用 Red Hat OpenStack 的 NFV 部署测试的 NIC 列表，请参阅测试的 NIC。

3.4. 与第三方软件兼容

有关与红帽技术(Red Hat OpenStack Platform)进行测试、支持和认证的产品和服务的完整列表，请参见与 Red Hat OpenStack Platform 兼容的第三方软件 。您可以根据产品版本和软件类别过滤列表。

有关与红帽技术(Red Hat Enterprise Linux)进行测试、支持和认证的产品和服务的完整列表，请参见 与 Red Hat Enterprise Linux 兼容的第三方软件。您可以根据产品版本和软件类别过滤列表。

第 4 章 网络注意事项

undercloud 主机至少需要以下网络：

- Provisioning 网络 - 提供 DHCP 和 PXE 引导功能，以帮助发现裸机系统以便在 overcloud 中使用。
- 外部网络 - 单独的网络，用于远程连接所有节点。连接到此网络的接口需要可路由的 IP 地址（静态定义）或通过外部 DHCP 服务动态定义。

最小 overcloud 网络配置包括：

- 单 NIC 配置 - 一个 NIC 在原生 VLAN 中用于 Provisioning 网络，并用于 tagged VLAN（使用子网处理不同的 overcloud 网络类型）。
- dual NIC 配置 - 一个 NIC 用于 Provisioning 网络，另一个用于 External 网络的 NIC。
- dual NIC 配置 - 一个 NIC 用于原生 VLAN 上的 Provisioning 网络，另一个用于使用子网进行不同 overcloud 网络类型的 VLAN。
- 多 NIC 配置 - 每个 NIC 都使用一个子网来分别处理 overcloud 中不同的网络类型。

有关网络要求的更多信息，请参阅 网络要求。

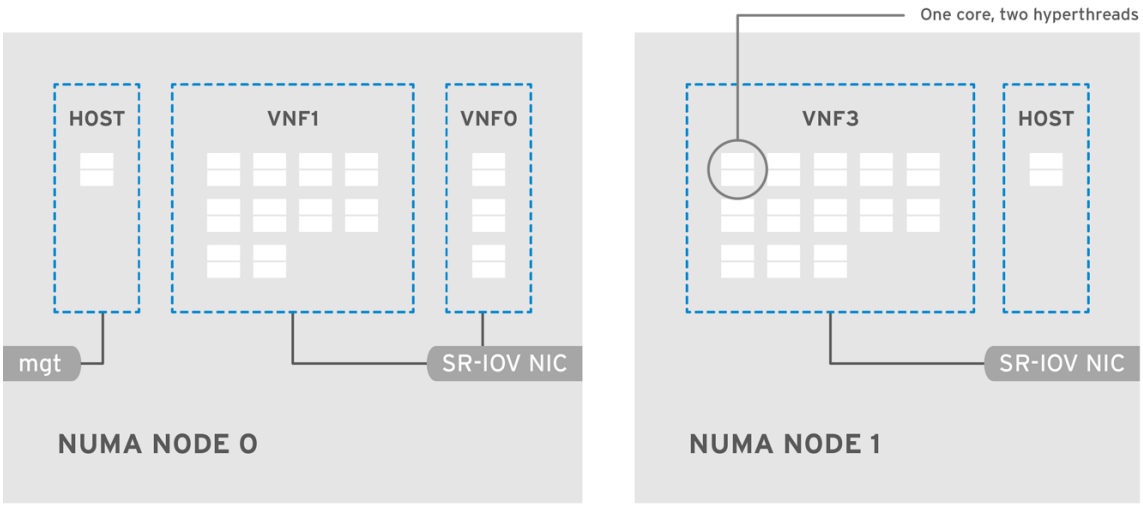
第 5 章 规划 SR-IOV 部署

根据您的 Compute 节点硬件设置单个参数，为 NFV 优化单根 I/O 虚拟化(SR-IOV)部署。

请参阅 发现您的 NUMA 节点拓扑，以评估您的硬件对 SR-IOV 参数的影响。

5.1. SR-IOV 部署的硬件分区

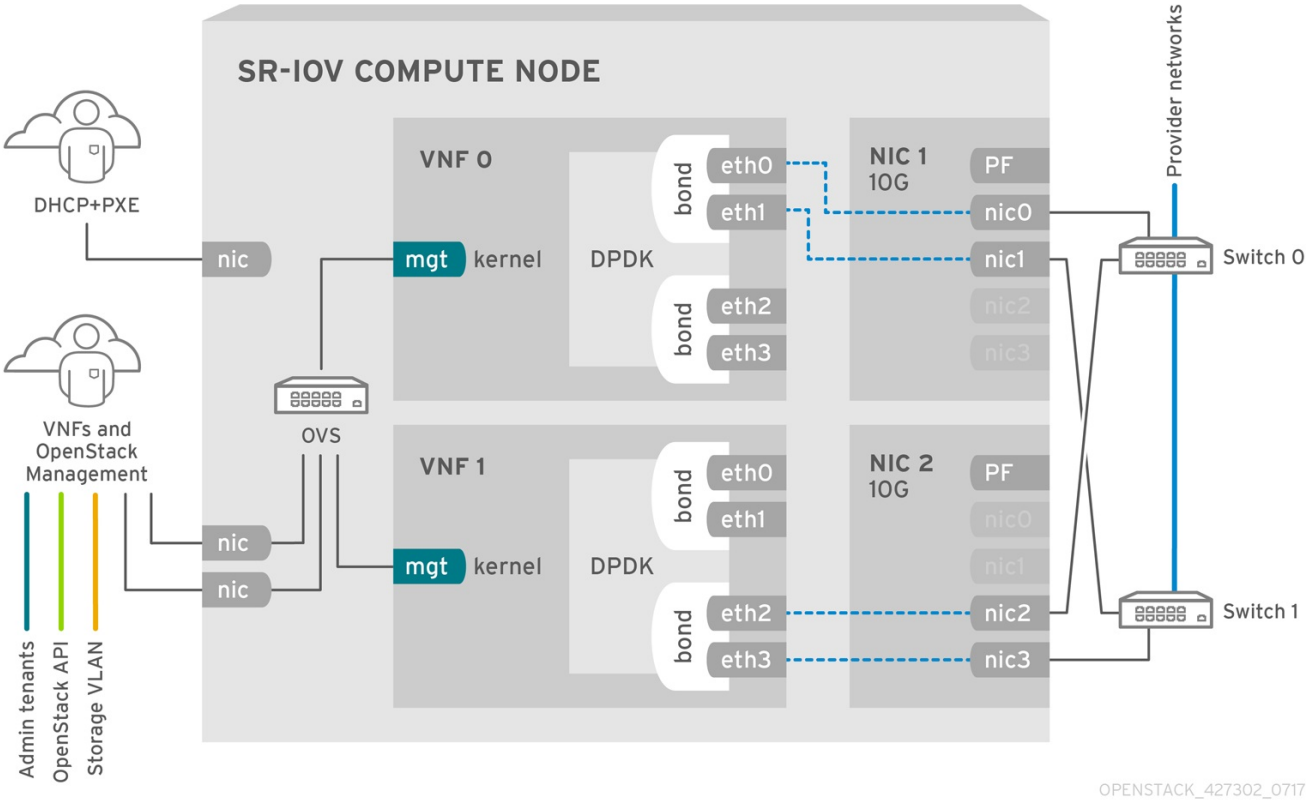
要使用 SR-IOV 实现高性能，您需要在主机和客户机之间分区资源。



典型的拓扑包括每个 NUMA 节点的 14 个核心，位于双插槽 Compute 节点上。支持超线程(HT)和非HT内核。每个内核都有两个同级线程。个核心专用于每个 NUMA 节点上的主机。VNF 处理 SR-IOV 接口绑定。所有中断请求(IRQ)在主机内核上路由。VNF 内核专用于 VNF。它们提供与其他 VNF 隔离以及与主机分开的隔离。每个 VNF 都必须在单个 NUMA 节点上使用资源。VNF 使用的 SR-IOV NIC 还必须与同一 NUMA 节点关联。此拓扑没有虚拟化开销。主机、OpenStack Networking (neutron)和计算(nova)配置参数在单一文件中公开，以简化一致性，避免导致抢占和数据包丢失。主机和虚拟机隔离取决于 **tuned** 配置文件，该配置集会根据要隔离的 CPU 列表处理引导参数和 OpenStack 修改。

5.2. NFV SR-IOV 部署的拓扑

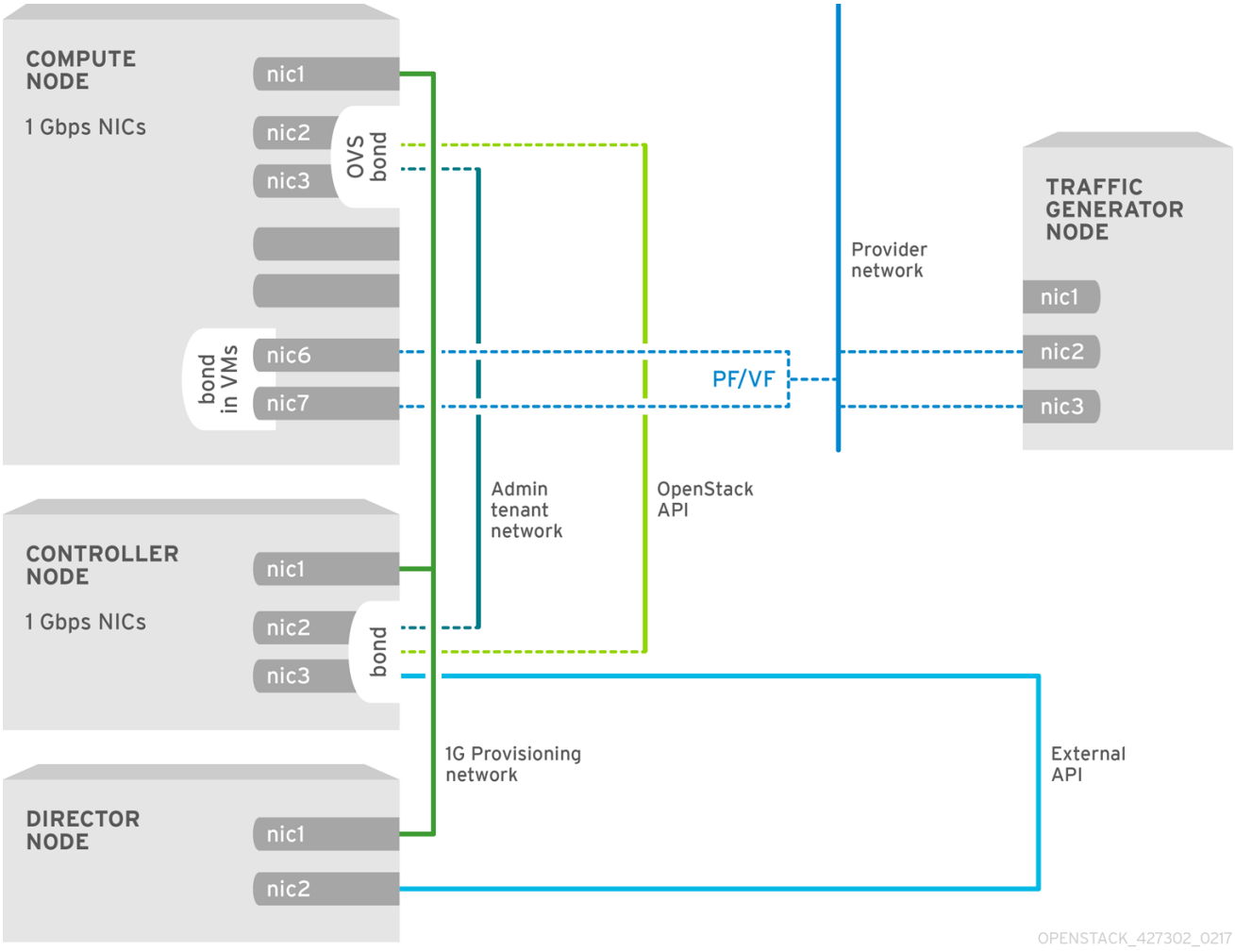
以下镜像有两个虚拟网络功能(VNF)，每个功能都带有 **mgt** 和 data plane 接口代表的管理接口。管理界面管理 **ssh** 访问等等。data plane 接口将 VNF 绑定到 Data Plane Development Kit (DPDK)以确保高可用性(VNF)使用 DPDK 库绑定 data plane 接口。该镜像也有两个冗余提供商网络。Compute 节点有两个常规 NIC 绑定在一起，并在 VNF 管理和红帽 OpenStack Platform API 管理间共享。



该镜像显示了一个 VNF，它利用了应用级别的 DPDK，并可以访问单一根 I/O 虚拟化(SR-IOV)虚拟功能(VF)和物理功能(PF)，以提高性能（取决于 fabric 配置）。DPDK 提高了性能，而 VF/PF DPDK 绑定支持故障转移（可用性）。VNF 供应商必须确保其 DPDK 轮询模式驱动程序(PMD)支持作为 VF/PF 公开的 SR-IOV 卡。管理网络使用 Open vSwitch (OVS)，以便 VNF 使用标准 virtIO 驱动程序来查看"mgmt"网络设备。操作员可以使用该设备来发起与 VNF 连接，并确保其 DPDK 应用正确绑定两个 VF/PF。

5.2.1. 没有 HCI 的 NFV SR-IOV

您可以参阅以下镜像中的 NFV 用例的单个根 I/O 虚拟化(SR-IOV)的拓扑。它由具有 1 Gbps NIC 和 Director 节点的计算和控制器节点组成。

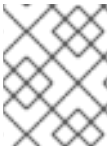

COMPUTE NODE
1 Gbps NICs
nic1
nic2
nic3
OVS bond
bond in VMs
nic6
nic7
CONTROLLER NODE
1 Gbps NICs
nic1
nic2
nic3
bond
DIRECTOR NODE
nic1
nic2
Admin tenant network
OpenStack API
1G Provisioning network
Provider network
PF/VF
TRAFFIC GENERATOR NODE
nic1
nic2
nic3
External API
OPENSTACK_427302_0217

第 6 章 部署 SR-IOV 技术

通过允许 OpenStack 中的实例通过虚拟资源直接访问共享 PCIe 资源，单个根 I/O 虚拟化(SR-IOV)允许接近裸机性能。

6.1. 前提条件

- 在部署 overcloud 之前安装和配置 undercloud。详情请查看 Director 安装和使用指南。



注意

不要手动编辑由 Director heat 模板修改的 **/etc/tuned/cpu-partitioning-variables.conf** 中的值。

6.2. 配置 SR-IOV

注意

以下示例的 CPU 分配、内存分配和 NIC 配置可能与您拓扑和用例的不同。

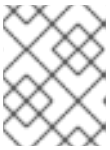
1. 生成内置的 **ComputeSriov**，以定义 OpenStack 集群中的节点，以运行 **NeutronSriovAgent**、**NeutronSriovHostConfig** 和默认计算服务。

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

2. 在生成 **overcloud_images.yaml** 时包括 **neutron-sriov.yaml** 和 **roles_data.yaml** 文件，以便 SR-IOV 容器已准备好。

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

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



注意

push-destination IP 地址是您之前使用 **undercloud.conf** 配置文件中的 **local_ip** 参数设置的地址。

如需有关容器镜像准备的更多信息，请参阅 Director 安装和使用。

3. 要将 **KernelAgs** 和 **TunedProfile** 参数包含在部署脚本中，请将 **/usr/share/openstack-tripleo-heat-templates/environments** 中的 **host-config-and-reboot.yaml** 文件包含在内。

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

4. 根据集群需求以及硬件配置，在 **parameter_defaults** 下配置 SR-IOV 节点的参数。这些设置通常添加到 **network-environment.yaml** 文件中。

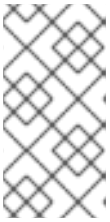
```
  NeutronNetworkType: 'vlan'
  NeutronNetworkVLANRanges:
    - tenant:22:22
    - tenant:25:25
  NeutronTunnelTypes: ''
```

5. 在同一文件中，为 SR-IOV 计算节点配置特定于角色的参数。

注意

NeutronSriovNumVFs 参数很快将被弃用，而是使用网络配置模板中的 **numvfs** 属性。部署后，红帽不支持修改 **NeutronSriovNumVFs** 参数，也无法修改 **numvfs** 参数。已知在运行中的环境中更改任何参数，会导致在该 PF 上具有 SR-IOV 端口的所有运行实例永久中断。除非硬重启这些实例，否则 SR-IOV PCI 设备将无法对实例可见。

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



注意

在配置 PCI 透传时，不要使用 **devname** 参数，因为 NIC 的设备名称可能会改变。反之，使用 **vendor_id** 和 **product_id**，因为它们更稳定，**或者使用** NIC 的地址。有关如何配置 **NovaPCIPassthrough** 的更多信息，请参阅有关 配置 **NovaPCIPassthrough**的指南。

6. 在 **compute.yaml** 网络配置模板中配置启用的 SR-IOV 接口。确保接口被配置为独立 NIC，用于创建 SR-IOV 虚拟功能(VF)：

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

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

7. 确保默认过滤器列表包含值 **AggregateInstanceExtraSpecsFilter**。

```
NovaSchedulerDefaultFilters:
['AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthroughFilter','AggregateInstanceExtraSpecsFilter']
```

8. 部署 overcloud。

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

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

6.3. 配置硬件卸载（技术预览）

Open vSwitch (OVS)硬件卸载是技术预览，不建议用于生产环境部署。有关技术预览功能的更多信息，请参阅 覆盖范围详情。

OVS 硬件卸载配置的步骤共享许多与配置 SR-IOV 相同的步骤。

流程

1. 生成 **ComputeSriov** 角色：

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

2. 将 **OvsHwOffload** 参数添加到特定于角色的参数下，值为 **true**。

3. 要将 neutron 配置为使用 iptables/hybrid 防火墙驱动程序实现，请包括：
NeutronOVSFirewallDriver: iptables_hybrid。有关 **NeutronOVSFirewallDriver** 的更多信息，*请参阅高级 Overcloud 自定义指南中的* 使用 Open vSwitch 防火墙。

4. 配置 **physical_network** 参数以匹配您的环境。

   - 对于 VLAN，请在部署后将 **physical_network** 参数设置为您在 neutron 中创建的网络名称。此值也应位于 **NeutronBridgeMappings** 中。

   - 对于 VXLAN，将 **physical_network** 参数设置为 **null**。
   例如：

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

   - 将 **<vendor-id** > 替换为物理 NIC 的供应商 ID。

   - 将 **<product-id** > 替换为 NIC VF 的产品 ID。

   - 将 **<** address> 替换为物理 NIC 的地址。
   有关如何配置 **NovaPCIPassthrough** 的更多信息，请参阅有关 配置 **NovaPCIPassthrough**的指南。

5. 确保默认过滤器列表包含 **NUMATopologyFilter** ：

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

6. 在 **compute-sriov.yaml** 配置文件中配置用于硬件卸载的一个或多个网络接口：

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

注意

- 在配置 Open vSwitch 硬件卸载时，不要使用 **NeutronSriovNumVFs** 参数。虚拟功能的数量使用 **os-net-config** 使用的网络配置文件中的 **numvfs** 参数指定。红帽不支持在更新或重新部署过程中修改 **numvfs** 设置。
- 不要将 Mellanox 网络接口配置为 nic-config 接口类型 **ovs-vlan**，因为这会阻止 VXLAN 等隧道端点会因为驱动程序限制而传递流量。

7. 在 **overcloud deploy** 命令中包含 **ovs-hw-offload.yaml** 文件：

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

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

6.3.1. 验证 OVS 硬件卸载

1. 确认 PCI 设备处于 **switchdev** 模式：

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

2. 验证 OVS 中是否启用了卸载：

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

3. 确认 NIC 上启用了硬件卸载：

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

6.4. 为 SR-IOV 部署实例

建议使用主机聚合来分隔高性能计算主机。有关创建主机聚合和相关类别的信息，请参阅创建主机聚合。

注意

您应该使用主机聚合将 CPU 固定实例与未固定实例分开。不使用 CPU 固定的实例不会遵循使用 CPU 固定的实例重新提供要求。

通过执行以下步骤为单根 I/O 虚拟化(SR-IOV)部署实例：

1. 创建类别。

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

2. 创建网络。

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

3. 创建端口。

- 使用 vnic-type 直接创建 SR-IOV 虚拟功能(VF)端口。

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

- **使用以下命令创建带有硬件卸载的虚拟功能。**

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

- **使用 vnic-type direct-physical 创建 SR-IOV PF 端口。**

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

4. **部署实例**

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

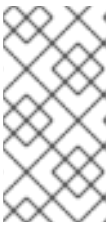
6.5. 创建主机聚合

使用 CPU 固定和大页部署客户机以提高性能。您可以通过与类别元数据匹配的聚合元数据，在主机子集上调度高性能实例。

流程

1. **您可以在部署前通过 heat 参数 NovaSchedulerDefaultFilters 在 nova.conf 配置文件中配置 AggregateInstanceExtraSpecs 值 和其他必要的过滤器。**

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



注意

要将 AggregateInstanceExtraSpecsFilter 配置添加到退出的集群，您可以在 heat 模板中添加此参数，然后再次运行原始部署脚本。

2. **为单根 I/O 虚拟化(SR-IOV)创建一个聚合组，并添加相关主机。定义与定义的类别元数据匹配的 metadata，如 sriov=true。**

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

3. **创建类别。**

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

4. **设置其他类别属性。请注意，定义的元数据 sriov=true 与 SR-IOV 聚合上定义的元数据匹配。**

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

第 7 章 规划 OVS-DPDK 部署

要利用 NFV 的 Data Plane Development Kit (OVS-DPDK)部署优化 Open vSwitch，您应该了解 OVS-DPDK 如何使用计算节点硬件(CPU、NUMA 节点、内存、NIC)以及决定基于 Compute 节点的各种 OVS-DPDK 参数的注意事项。

重要

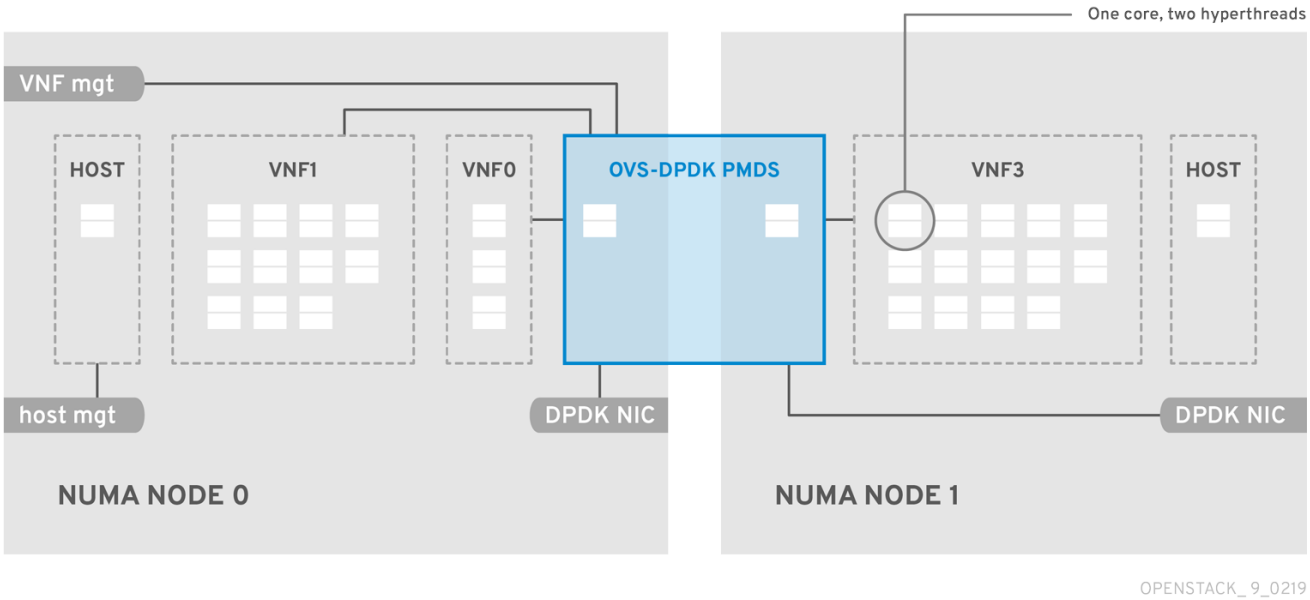
使用 OVS-DPDK 和 OVS 原生防火墙（基于 conntrack 的有状态防火墙）时，您只能跟踪使用 ICMPv4、ICMPv6、TCP 和 UDP 协议的数据包。OVS 会将所有其他网络流量标记为无效。

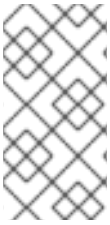
有关 CPU 和 NUMA 拓扑的高级介绍，请参阅 NFV 性能注意事项。

7.1. 带有 CPU 分区和 NUMA 拓扑的 OVS-DPDK

OVS-DPDK 对主机、客户机和 OVS-DPDK 本身的硬件资源分区。OVS-DPDK 轮询模式驱动程序 (PMD)运行 DPDK 活跃循环，这需要专用内核。这意味着，CPU 和 Huge Pages 列表专用于 OVS-DPDK。

示例分区包括每个双插槽 Compute 节点上的每个 NUMA 节点的 16 个内核。流量需要额外的 NIC，因为无法在主机和 OVS-DPDK 之间共享 NIC。



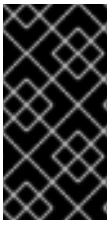


注意

DPDK PMD 线程必须在两个 NUMA 节点上保留，即使 NUMA 节点没有关联的 DPDK NIC。

OVS-DPDK 性能还取决于为 NUMA 节点保留本地内存块。使用与用于内存和 CPU 固定的相同 NUMA 节点关联的 NIC。同时确保绑定中的两个接口来自同一 NUMA 节点上的 NIC。

7.2. 工作流和派生参数概述



重要

该功能在此发行版本中作为*技术预览*提供，因此不享有红帽的全面支持。它只应用于测试，不应部署在生产环境中。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

您可以使用 OpenStack Workflow (mistral)服务根据可用的裸机恢复节点的功能来派生参数。OpenStack 工作流使用 .yaml 文件来定义一组要执行的任务。您可以在 tripleo-common/workbooks/ 目录中使用预定义的工作簿中执行生成的 _params.yaml。本工作簿提供了从裸机内省检索的结果获取每个支持的参数的工作流。derived _params.yaml 工作流使用来自 tripleo-common/workbooks/derive_params_formulas.yaml 的公式来计算派生的参数。

注意

您可以在 derived _params_formulas.yaml 中修改公式以适应您的环境。

derived_params.yaml 工作簿假定 *给定可组合角色* 的所有节点具有相同的硬件规格。工作流会考虑 flavor-profile 关联和 nova 放置调度程序，以匹配与角色关联的节点，并使用第一个与角色匹配的内省数据。

如需了解有关 OpenStack 工作流的详细信息，请参阅对工作流进行故障排除 和执行。

您可以使用 -p 或 --plan-environment-file 选项，将自定义 plan_environment.yaml 文件添加到 openstack overcloud deploy 命令中。自定义 plan_environment.yaml 文件提供工作簿中的列表，以及要传递给工作簿中的任何输入值。触发的工作流将派生的参数合并到自定义 plan_environment.yaml 中，它们可用于 overcloud 部署。您可以使用这些派生的参数结果来准备 overcloud 镜像。

如需了解如何在 部署中使用 --plan-environment-file 选项的详细信息，请参阅计划 环境元数据。

7.3. 派生的 OVS-DPDK 参数

derived _params.yaml 中的工作流派生了与使用 ComputeNeutronOvsDpdk 服务的匹配角色关联的 DPDK 参数。

以下是工作流可以为 OVS-DPDK 自动派生的参数列表：

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

OvsDpdkMemoryChannels 参数无法从内省内存银行数据衍生，因为内存插槽名称的格式在不同的硬件环境中不一致。

在大多数情况下，OvsDpdkMemoryChannels 应该为 4 （默认）。使用硬件手册来决定每个套接字的内存频道数量，并使用这个值覆盖默认值。

有关配置详情，请参阅 第 8.1 节 “使用工作流推断 DPDK 参数”。

7.4. 手动计算 OVS-DPDK 参数的概述

本节论述了如何使用 Data Plane Development Kit (OVS-DPDK)在 director network_environment.yaml HEAT 模板中使用参数来配置 CPU 和内存以实现最佳性能。使用这些信息评估您的 Compute 节点上的硬件支持，以及该硬件分区以优化 OVS-DPDK 部署的方式。

注意

如果您使用 derived_parameters.yaml 工作流自动生成这些值，则不需要手动计算这些参数。查看 工作流和派生参数概述

注意

在分配 CPU 内核时，始终对物理核心的 CPU 同级线程（逻辑 CPU）在一起。

请参阅 发现 NUMA 节点拓扑，以确定 Compute 节点上的 CPU 和 NUMA 节点。您可以使用这些信息映射 CPU 和其他参数来支持主机、客户机实例和 OVS-DPDK 进程的需求。

7.4.1. CPU 参数

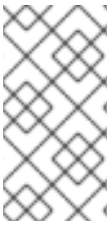
OVS-DPDK 使用以下 CPU 分区参数：

OvsPmdCoreList

提供用于 DPDK 轮询模式驱动程序(PMD)的 CPU 内核。选择与 DPDK 接口本地 NUMA 节点关联的 CPU 内核。OvsPmdCoreList 用于 Open vSwitch 中的 pmd-cpu-mask 值。

- 对同级的线程连接在一起。
- 从 OvsDpdkCoreList中排除所有内核
- 避免为两个 NUMA 节点上的第一个物理内核分配逻辑 CPU （线程同级），因为它们应该用于 OvsDpdkCoreList 参数。
- 性能取决于为此 PMD 核心列表分配的物理内核数。在与 DPDK NIC 关联的 NUMA 节点上，分配所需的内核。

- 对于使用 DPDK NIC 的 NUMA 节点：
  - 根据性能要求确定所需的物理内核数量，并包含每个物理内核的所有同级线程（逻辑 CPU）。
- 对于没有 DPDK NIC 的 NUMA 节点：
  - 分配一个物理内核（逻辑 CPU）的同级线程（逻辑 CPU），排除了 NUMA 节点的第一个物理核心。



注意

DPDK PMD 线程必须在两个 NUMA 节点上保留，即使 NUMA 节点没有关联的 DPDK NIC。

NovaVcpuPinSet

为 CPU 固定设置内核。Compute 节点将这些核心用于客户机实例。NovaVcpuPinSet 用作 nova.conf 文件中的 vcpu_pin_set 值。

- 从 OvsPmdCoreList 和 OvsDpdkCoreList 中排除所有核心。
- 包括所有剩余内核。
- 对同级的线程连接在一起。

NovaComputeCpuSharedSet

设置用于仿真程序线程的内核。这将定义 nova.conf 参数 cpu_shared_set 的值。此参数的建议值与 OvsDpdkCoreList 设置的值匹配。

IsolCpusList

与主机进程隔离的一组 CPU 核心。此参数用作 tuned-profiles-cpu-partitioning 组件的 cpu-partitioning-variable.conf 文件中的 isolated_cores 值。

- 匹配 OvsPmdCoreList 和 NovaVcpuPinSet 中的内核数。

- 对同级的线程连接在一起。

OvsDpdkCoreList

为非数据路径 OVS-DPDK 进程（如处理程序和 revalidator 线程）提供 CPU 内核。这个参数不会影响多 NUMA 节点硬件上的整个数据路径性能。此参数用于 Open vSwitch 中的 dpdk-lcore-mask 值，这些核心与主机共享。

- 为每个 NUMA 节点分配第一个物理内核（和同级线程）（即使 NUMA 节点没有关联的 DPDK NIC）。

- 这些内核必须从 OvsPmdCoreList 和 NovaVcpuPinSet 中的内核数相互排斥。

DerivePciWhitelistEnabled

要为虚拟机保留虚拟功能(VF)，请使用 NovaPCIPassthrough 参数创建通过 Nova 传递给 Nova 的 VF 列表。列表中排除的虚拟机可用于主机。

红帽建议将 DerivePciWhitelistEnabled 值从 true 的默认值改为 false，然后在 NovaPCIPassthrough 参数中手动配置列表。

对于列表中的每个 VF，使用解析到地址值的正则表达式填充 address 参数。

以下是手动列表创建过程的示例。如果在名为 eno2 的设备中启用 NIC 分区，使用以下命令列出 VF 的 PCI 地址：

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

在本例中，VF 0、4 和 6 供 eno2 用于 NIC 分区。手动配置 NovaPCIPassthrough 以包括 VF 1-3、5 和 7，因此排除 VFs 0、4 和 6，如下例所示：

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

7.4.2. 内存参数

OVS-DPDK 使用下列内存参数：

OvsDpdkMemoryChannels

映射每个 NUMA 节点的 CPU 中的内存频道。Open vSwitch 使用 OvsDpdkMemoryChannels 参数作为 other_config:dpdk-extra="-n <value>" 值。

- 使用 dmidecode -t memory 或您的硬件手册来确定可用的内存通道数。
- 使用 ls /sys/devices/system/node/node* -d 确定 NUMA 节点的数量。
- 将可用内存通道数除以 NUMA 节点数。

NovaReservedHostMemory

以 MB 为单位保留主机上任务的内存。Compute 节点使用这个值作为 nova.conf 中的 reserved_host_memory_mb 值。

- 使用静态推荐值 4096 MB。

OvsDpdkSocketMemory

指定从每个 NUMA 节点预先分配巨页池的内存量（以 MB 为单位）。这个值被 Open vSwitch 用作 other_config:dpdk-socket-mem 值。

- 提供 作为用逗号分开的列表。从 NUMA 节点上的每个 NIC 的 MTU 值计算 OvsDpdkSocketMemory 值。

- 对于没有 DPDK NIC 的 NUMA 节点，请使用 1024 MB (1GB)的静态建议。

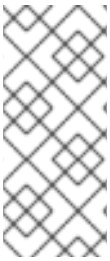
- 以下驱除 OvsDpdkSocketMemory 的值如下：

  - MEMORY_REQD_PER_MTU = (ROUNDUP_PER_MTU + 800) * (4096 * 64) Bytes

    - 800 是开销值。

    - 4096 * 64 是 mempool 中的数据包数量。

- 为 NUMA 节点上设置的每个 MTU 值添加 MEMORY_REQD_PER_MTU，再添加另一个 512 MB 作为缓冲区。将值设为 1024 的倍数。



注意

如果 MTU 大小不是 1500，您可能会在 /var/log/messages 中创建 内存池 错误消息。如果出现在实例启动时发生，可以忽略此错误消息。要避免此消息，请将 1500 MTU 的额外 OvsDpdkSocketMemory 增加到您的 OvsDpdkSocketMemory 计算。

Calculation 示例 - MTU 2000 和 MTU 9000

DPDK NIC dpdk0 和 dpdk1 位于相同的 NUMA 节点上，分别使用 MTU 9000 和 2000 进行配置。生成内存所需的计算示例如下：

1. 将 MTU 值向下舍入到最接近 1024 字节。

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

2. 根据这些循环的字节值计算每个 MTU 值所需的内存。

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

3. 计算所需的总内存总量（以字节为单位）。

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

此计算代表(MTU 为 9000 所需要的内存)+(MTU 为 2000)+(512 MB 缓冲所需的内存)。

4. 将所需的总内存转换为 MB。

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

5. 将该值向上舍入到最接近的 1024。

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

6. 使用这个值设置 OvsDpdkSocketMemory。

```
OvsDpdkSocketMemory: "4096,1024"
```

Calculation 示例 - MTU 2000

DPDK NIC dpdk0 和 dpdk1 位于相同的 NUMA 节点上，分别使用 MTU 2000 和 2000 进行配置。生成内存所需的计算示例如下：

1. 将 MTU 值向下舍入到最接近 1024 字节。

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

2. 根据这些循环的字节值计算每个 MTU 值所需的内存。

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

3. 计算所需的总内存总量（以字节为单位）。

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

此计算代表(MTU 为 2000)+(512 MB 缓冲)所需的内存。

4. 将所需的总内存转换为 MB。

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

5. 将该值向上舍入到最接近的 1024。

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

6. 使用这个值设置 OvsDpdkSocketMemory。

```
OvsDpdkSocketMemory: "2048,1024"
```

7.4.3. 网络参数

OvsDpdkDriverType

设置 DPDK 使用的驱动程序类型。使用 vfio-pci 的默认值。

NeutronDatapathType

OVS 网桥的数据路径类型。DPDK 使用 netdev 的默认值。

NeutronVhostuserSocketDir

为 OVS 设置 vhost-user 套接字目录。将 /var/lib/vhost_sockets 用于 vhost 客户端模式。

7.4.4. 其他参数

NovaSchedulerDefaultFilters

提供有序的过滤器列表，计算节点用于为请求的客户机实例查找匹配的 Compute 节点。

VhostuserSocketGroup

设置 vhost-user 套接字目录组。默认值为 qemu。*VhostuserSocketGroup* 应设置为 hugetlbfs，以便 ovs-vswitchd 和 qemu 进程可以访问用来配置 virtio-net 设备的共享巨页和 unix 套接字。此值特定于角色，应当应用到利用 OVS-DPDK 的任何角色。

KernelArgs

在引导时为 Compute 节点提供多个内核参数。根据您的配置添加以下内容：

- hugepagesz ：设置 CPU 中巨页的大小。这个值可能因 CPU 硬件而异。为 OVS-DPDK 部署(默认_hugepagesz=1GB hugepagesz=1G)设置为 1G。检查 pdpe1gb CPU 标记，以确保您的 CPU 支持 1G。

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

- hugepages count ：设置可用巨页的数量。这个值取决于可用的主机内存量。使用大多数可用内存（排除 NovaReservedHostMemory）。您还必须配置与 Compute 节点关联的 OpenStack 类别中的巨页数值。

- IOMMU: 对于 Intel CPU，添加 "intel_iommu=on iommu=pt"'

- isolcpus ：设置要调整的 CPU 内核。这个值与 IsolCpusList 匹配。

7.4.5. 实例额外规格

在 NFV 环境中部署实例之前，先创建一个类别，它将使用 CPU 固定、仿真程序线程固定和大页面。

hw:cpu_policy

将此参数的值设为 dedicated，以便 guest 将使用固定 CPU。从带有此参数集的类别创建的实例将有效过量使用 1:1。默认为 共享。

hw:mem_page_size

将此参数设置为带有标准后缀的特定值的有效字符串，例如： 4KB、8MB 或 1GB。使用 1GB 匹配 hugepagesz 引导参数。虚拟机的巨页数量是引导参数减去 OvsDpdkSocketMemory。其他有效的参数值包括：

- small （默认）- 使用最小页面大小

- large - 只使用大页大小。(x86 架构上的 2MB 或 1GB)
- any - 计算驱动程序可能会尝试大页，但如果没有可用，则默认为 small。

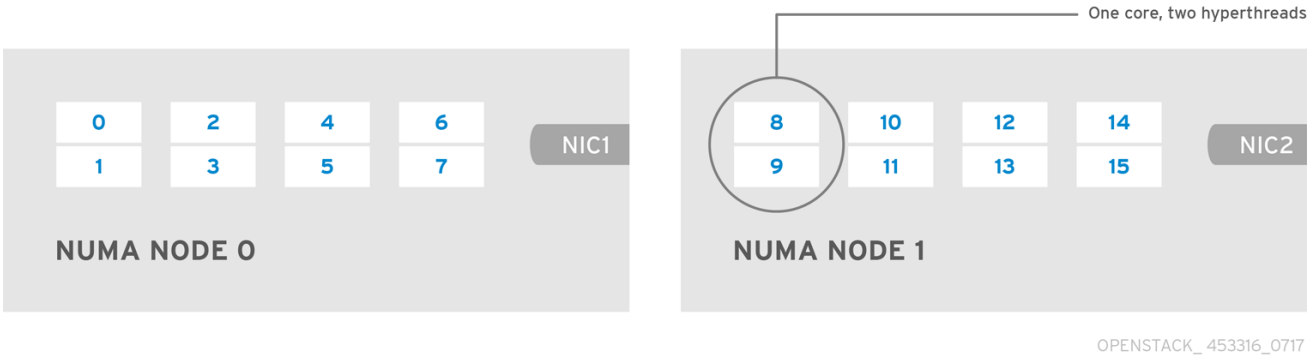
hw:emulator_threads_policy

将此参数的值设为 share，以便仿真程序线程锁定在 heat 参数(NovaComputeCpuSharedSet 中已标识)的 CPU。如果仿真程序线程在 vCPU 上运行，用于轮询模式驱动程序(PMD)或实时处理，您可以遇到数据包丢失或丢失的期限。

7.5. 两个 NUMA 节点示例 OVS-DPDK 部署

此 Compute 节点示例包括两个 NUMA 节点，如下所示：

- NUMA 0 具有内核 0-7。同级线程对有(0,1)、(2,3)、(4,5)和(6,7)
- NUMA 1 有内核 8-15。同级线程对是(8,9),(10,11)、(12,13)和(14,15)。
- 每个 NUMA 节点连接到物理 NIC (NUMA 0 上的 NIC1 和 NUMA 1 上的 NIC2)。



注意

对于非数据路径 DPDK 进程(OvsDpdkCoreList)，在每个 NUMA 节点上保留第一个物理内核（包括线程对(0、1 和 8)。

本例还假设一个 1500 MTU 配置，因此 OvsDpdkSocketMemory 对所有用例都相同：

```
OvsDpdkSocketMemory: "1024,1024"
```

NIC 1 用于 DPDK，对 PMD 有一个物理内核

在这种情况下，您可以为 PMD 分配 NUMA 0 上的一个物理内核。您还必须在 NUMA 1 上分配一个物理内核，即使该 NUMA 节点的 NIC 上没有启用 DPDK。剩余的内核（没有为 OvsDpdkCoreList保留）已分配给客户机实例。生成的参数设置有：

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

NIC 1 用于 DPDK，用于 PMD 的两个物理内核

在这种情况下，您可以为 PMD 分配两个物理内核在 NUMA 0 上。您还必须在 NUMA 1 上分配一个物理内核，即使该 NUMA 节点的 NIC 上没有启用 DPDK。剩余的内核（没有为 OvsDpdkCoreList保留）已分配给客户机实例。生成的参数设置有：

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

NIC 2 用于 DPDK，对 PMD 有一个物理内核

在这种情况下，您可以为 PMD 为 NUMA 1 分配一个物理内核。您还必须在 NUMA 0 上分配一个物理内核，即使该 NUMA 节点的 NIC 上没有启用 DPDK。剩余的内核（没有为 OvsDpdkCoreList保留）已分配给客户机实例。生成的参数设置有：

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

NIC 2 用于 DPDK，用于 PMD 的两个物理内核

在这种情况下，您可以为 PMD 在 NUMA 1 上分配两个物理内核。您还必须在 NUMA 0 上分配一个物理内核，即使该 NUMA 节点的 NIC 上没有启用 DPDK。剩余的内核（没有为 OvsDpdkCoreList保留）已分配给客户机实例。生成的参数设置有：

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

用于 DPDK 的 NIC 1 和 NIC2，用于 PMD 的两个物理内核

在这种情况下，您可以为 PMD 在每个 NUMA 节点上分配两个物理内核。剩余的内核（没有为 OvsDpdkCoreList保留）已分配给客户机实例。生成的参数设置有：

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

注意

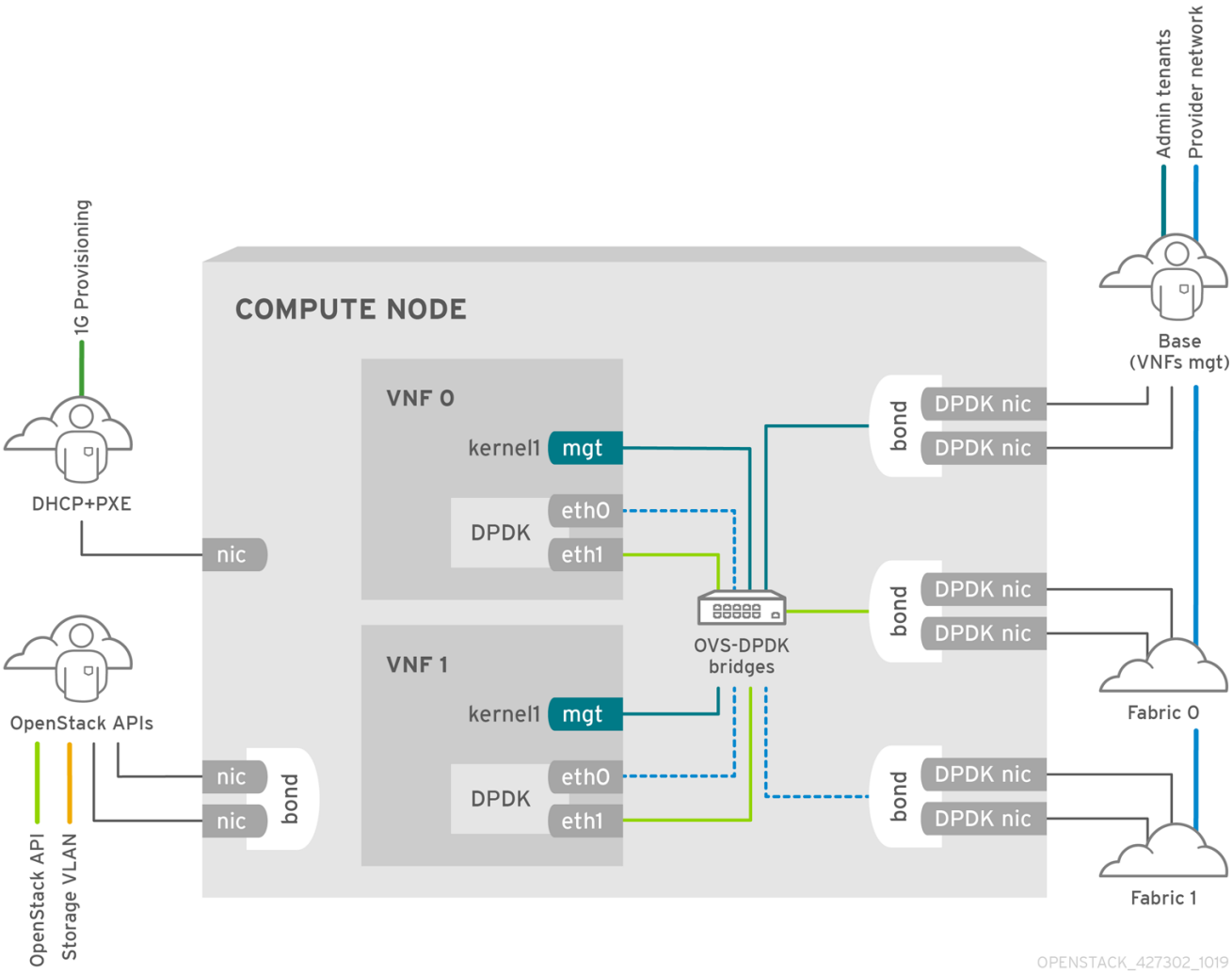
红帽建议每个 NUMA 节点使用 1 个物理内核。

7.6. NFV OVS-DPDK 部署的拓扑

这个示例部署显示 OVS-DPDK 配置，它由两个虚拟网络功能(VNF)组成，各自有两个接口：

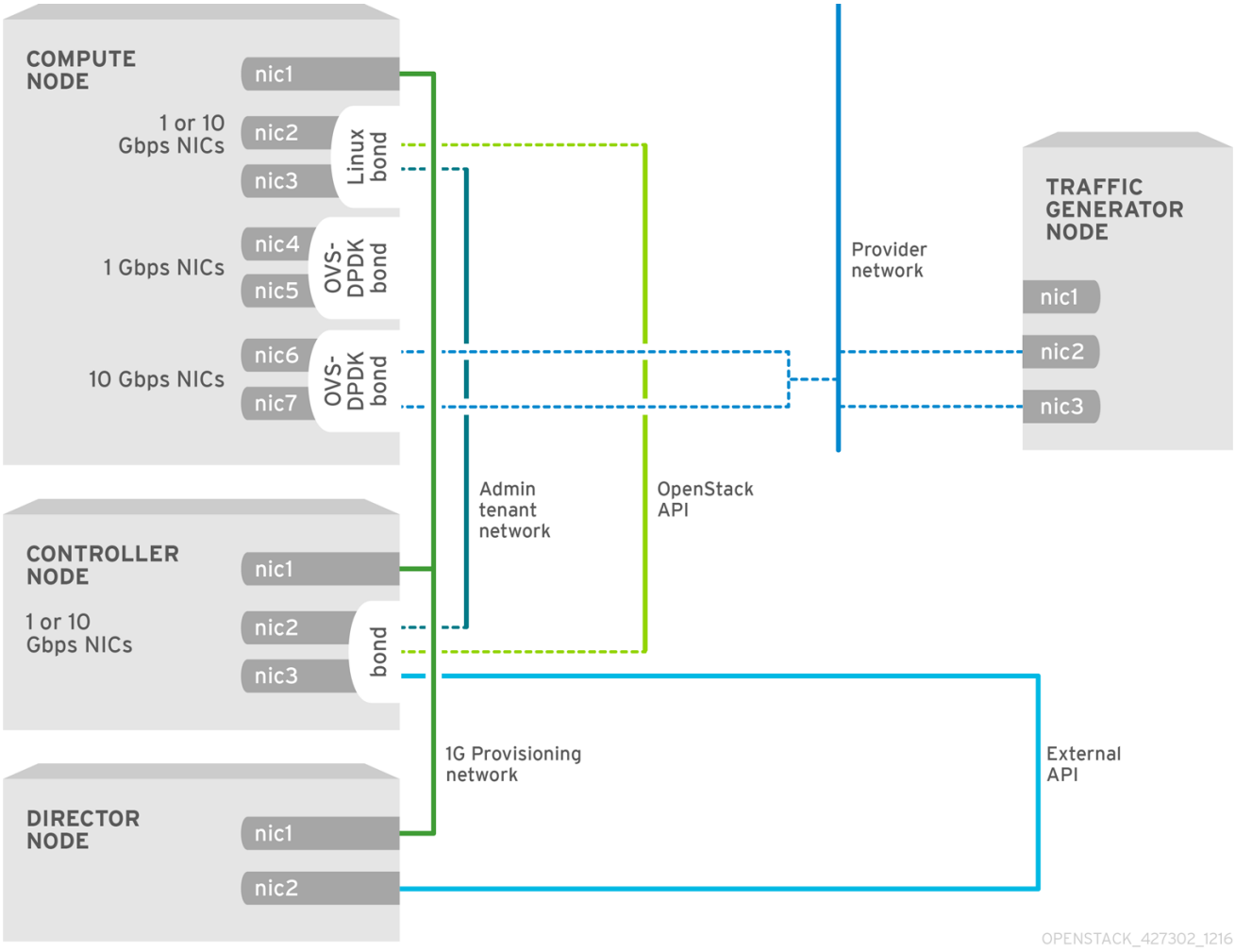
- 管理接口，由 mgt 表示。
- data plane 接口。

在 OVS-DPDK 部署中，VNF 使用支持物理接口的、内置 DPDK 来运行。OVS-DPDK 在 vSwitch 级别上启用绑定。为了提高 OVS-DPDK 部署的性能，建议您分隔内核和 OVS-DPDK NIC。要分隔管理(mgt)网络，连接到虚拟机的 Base 提供商网络，请确保您有额外的 NIC。Compute 节点由两个常规 NIC 组成，用于 Red Hat OpenStack Platform API 管理，它们可以被 Ceph API 使用，但不能与任何 OpenStack 项目共享。



NFV OVS-DPDK 拓扑

下图显示了 NFV 用例的 OVS_DPDK 的拓扑：它由具有 1 个或 10 Gbps NIC 和 Director 节点的 Compute 和 Controller 节点组成。


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

第 8 章 配置 OVS-DPDK 部署

本节在 Red Hat OpenStack Platform 环境中使用 Open vSwitch (OVS-DPDK)部署 DPDK。overcloud 通常由预定义角色的节点组成，如 Controller 节点、计算节点和不同的存储节点类型。这些默认角色各自包含 director 节点上的核心 Heat 模板中定义的一组服务。

您必须先安装和配置 undercloud，然后才能部署 overcloud。详情请查看 Director 安装和使用指南。

重要

您必须确定 network-environment.yaml 文件中设置的 OVS-DPDK 参数的最佳值，以优化 OVS-DPDK 的 OpenStack 网络。

注意

不要编辑或更改 etc/tuned/cpu-partitioning-variables.conf 中的 isolated_cores 或其他值，这些 director heat 模板修改。

8.1. 使用工作流推断 DPDK 参数

重要

该功能在此发行版本中作为*技术预览*提供，因此不享有红帽的全面支持。它只应用于测试，不应部署在生产环境中。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

有关 DPDK 的 Mistral 工作流概述，请参阅 第 7.2 节 “工作流和派生参数概述”。

前提条件

您必须启用裸机内省，包括启用硬件检查额外的(inspection_extras)来提供此工作流检索的数据。默认情况下启用硬件检查额外的功能。请参阅 检查节点的硬件。

为 DPDK 定义工作流和输入参数

下表列出了您可以提供给 OVS-DPDK 工作流的输入参数：

num_phy_cores_per_numa_node_for_pmd

此输入参数指定与 DPDK NIC 关联的 NUMA 节点所需的最小内核数。为不与 DPDK NIC 关联的其他 NUMA 节点分配一个物理内核。这个参数应设置为 1。

huge_page_allocation_percentage

此输入参数指定配置为巨页所需的内存总量（不包括 NovaReservedHostMemory）。KernelArgs 参数使用基于指定的 huge_page_allocation_percentage 计算的巨页来派生。这个参数应设置为 50。

工作流使用这些输入参数以及裸机内省详情来计算适当的 DPDK 参数值。

为 DPDK 定义工作流和输入参数：

1. 将 /usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-params.yaml 文件复制到一个本地目录，并将输入参数设置为适合您的环境。

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

2. 运行 openstack overcloud deploy 命令并包含以下内容：

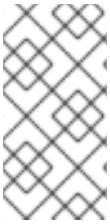
- update-plan-only 选项
- 特定于您的环境的角色文件和所有环境文件
- plan-environment-derived-parms.yaml 文件，使用 --plan-environment-file 可选参数

```
$ openstack overcloud deploy --templates --update-plan-only \
```

```
-r /home/stack/roles_data.yaml \
-e /home/stack/<environment-file> \
... #repeat as necessary ...
-p /home/stack/plan-environment-derived-params.yaml
```

此命令的输出显示派生结果，结果也合并到 plan-environment.yaml 文件中。

```
Started Mistral Workflow tripleo.validations.v1.check_pre_deployment_validations. Execution ID: 55ba73f2-2ef4-4da1-94e9-eae2fdc35535
Waiting for messages on queue 472a4180-e91b-4f9e-bd4c-1fbdfbcf414f with no timeout.
Removing the current plan files
Uploading new plan files
Started Mistral Workflow tripleo.plan_management.v1.update_deployment_plan. Execution ID: 7fa995f3-7e0f-4c9e-9234-dd5292e8c722
Plan updated.
Processing templates in the directory /tmp/tripleoclient-SY6RcY/tripleo-heat-templates
Invoking workflow (tripleo.derive_params.v1.derive_parameters) specified in plan-environment file
Started Mistral Workflow tripleo.derive_params.v1.derive_parameters. Execution ID: 2d4572bf-4c5b-41f8-8981-c84a363dd95b
Workflow execution is completed. result:
ComputeOvsDpdkParameters:
 IsolCpusList: 1-7,17-23,9-15,25-31
 KernelArgs: default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt intel_iommu=on isolcpus=1-7,17-23,9-15,25-31
 NovaReservedHostMemory: 4096
 NovaVcpuPinSet: 2-7,18-23,10-15,26-31
 OvsDpdkCoreList: 0,16,8,24
 OvsDpdkMemoryChannels: 4
 OvsDpdkSocketMemory: 1024,1024
 OvsPmdCoreList: 1,17,9,25
```



注意

OvsDpdkMemoryChannels 参数不能从内省详情衍生。在大多数情况下，这个值应该是 4。

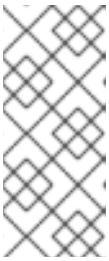
使用 Derived 参数部署 Overcloud

使用这些派生参数部署 overcloud：

1. 将派生的参数从 plan-environment.yaml 复制到 network-environment.yaml 文件。

```
# DPDK compute node.
ComputeOvsDpdkParameters:
  KernelArgs: default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt intel_iommu=on isolcpus=1-7,17-23,9-15,25-31
```

```
TunedProfileName: "cpu-partitioning"
IsolCpusList: "1-7,17-23,9-15,25-31"
NovaVcpuPinSet: ['2-7,18-23,10-15,26-31']
NovaReservedHostMemory: 4096
OvsDpdkSocketMemory: "1024,1024"
OvsDpdkMemoryChannels: "4"
OvsDpdkCoreList: "0,16,8,24"
OvsPmdCoreList: "1,17,9,25"
```



注意

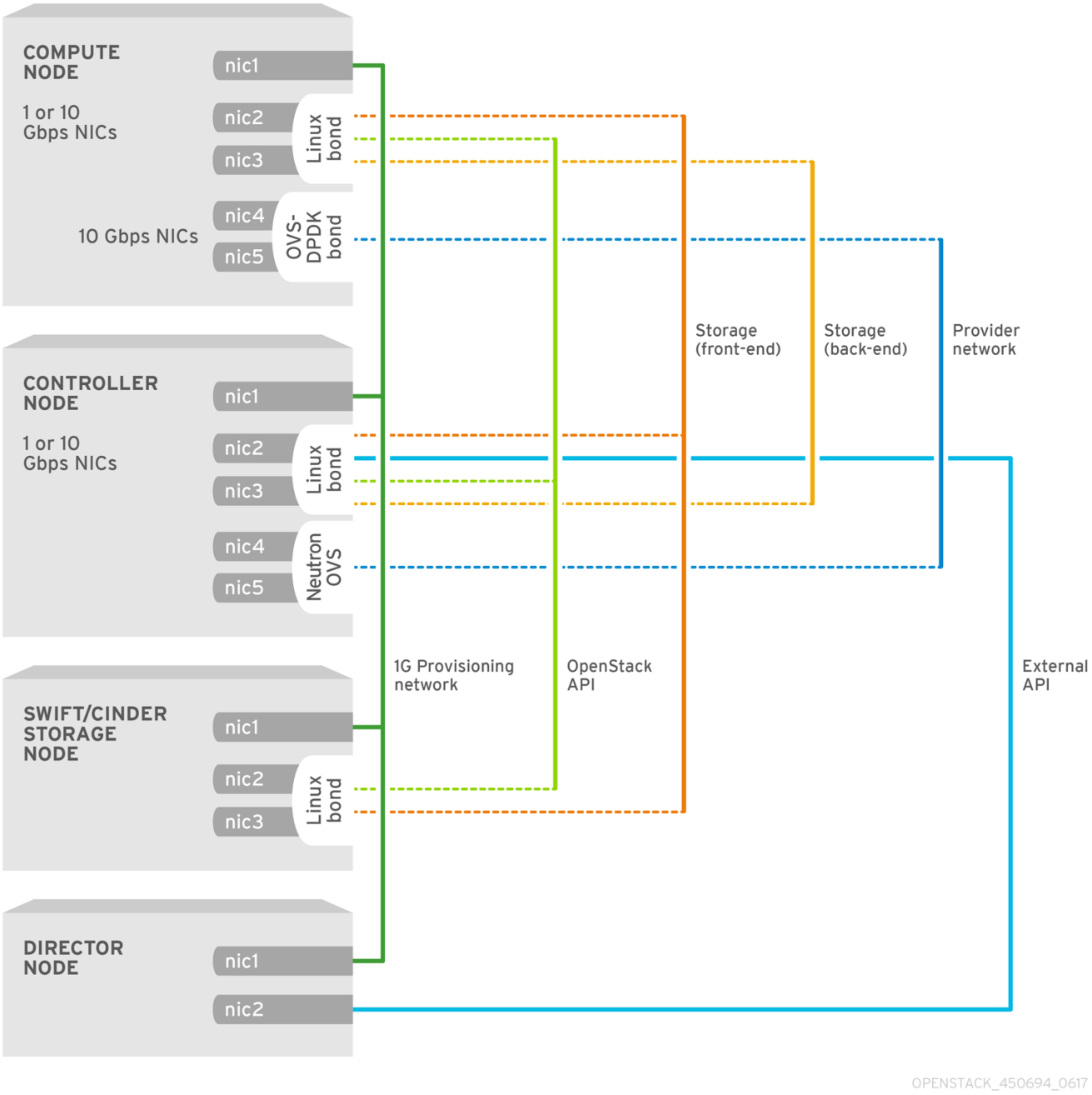
这些参数专门用于角色 ComputeOvsDpdk，且不适用于其他角色，包括 同一集群中 可能存在的 Compute 或 ComputeSriov。您可以在全局范围内应用这些参数，但任何全局参数都会被特定于角色的参数覆盖。

2. 使用角色文件以及特定于您的环境的所有环境文件，部署 overcloud。有关详细信息，请参阅部署 Overcloud。

8.2. OVS-DPDK 拓扑

在 Red Hat OpenStack Platform 中，您可以使用可组合角色功能创建自定义部署角色，从每个角色添加或删除服务。如需有关可组合角色的更多信息，请参阅可组合角色和服务。

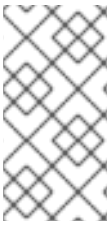
此镜像显示了一个带有 Data Plane Development Kit (OVS-DPDK)拓扑的 Open vSwitch 示例，它有两个绑定端口用于 control plane 和数据平面：



配置 OVS-DPDK 包括以下任务：

- **如果使用可组合角色，请复制并修改 roles_data.yaml 文件，以便为 OVS-DPDK 添加自定义角色。**
- **更新适当的 network-environment.yaml 文件，使其包含内核参数和 DPDK 参数的参数。**
- **更新 compute.yaml 文件，使其包含 DPDK 接口参数的网桥。**

- 更新 controller.yaml 文件，使其包含 DPDK 接口参数的相同网桥详情。
- 运行 overcloud_deploy.sh 脚本，以使用 DPDK 参数部署 overcloud。



注意

本指南提供了 CPU 分配、内存分配和 NIC 配置可能因拓扑和用例而异的示例。请参阅网络功能虚拟化产品指南和 第 2 章 *硬件要求*，以了解硬件和配置选项。

在开始这个过程前，请确保至少满足以下条件：

- OVS 2.9
- DPDK 17
- 测试的 NIC。有关 NFV 测试的 NIC 列表，请参阅 第 2.1 节 “网络适配器支持”。

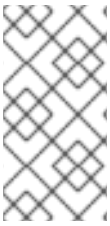
注意

Red Hat OpenStack Platform 在 OVS-DPDK 部署的 OVS 客户端模式中运行。

8.3. 为 OVS-DPDK 接口设置 MTU 值

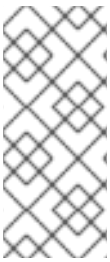
Red Hat OpenStack Platform 支持 Open vSwitch 带有 Data Plane Development Kit (OVS-DPDK) 的巨型帧。要为巨型帧设置最大传输单元(MTU)值：

- 在 network-environment.yaml 文件中为 networking 设置全局 MTU 值。
- 在 compute.yaml 文件中设置物理 DPDK 端口 MTU 值。vhost 用户界面也使用这个值。
- 在 Compute 节点上的任何客户端实例中设置 MTU 值，以确保您的配置中有可比较的 MTU 值从结尾。



注意

VXLAN 数据包在标头中包含额外的 50 字节。根据这些额外标头字节计算您的 MTU 要求。例如，MTU 值为 9000，表示 VXLAN 隧道 MTU 8950 用于这些额外字节。



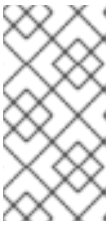
注意

您不需要物理 NIC 的任何特殊配置，因为 NIC 由 DPDK PMD 控制，并且具有 compute.yaml 文件设置的相同 MTU 值。您不能设置大于物理 NIC 支持的最大值的 MTU 值。

为 OVS-DPDK 接口设置 MTU 值：

1. 在 network-environment.yaml 文件中设置 NeutronGlobalPhysnetMtu 参数。

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



注意

确保 network-environment.yaml 文件中的 NeutronDpdkSocketMemory 值足够大，以支持巨型帧。详情请查看 第 7.4.2 节 “内存参数”。

2. 将网桥上的 MTU 值设置为 controller.yaml 文件中的 Compute 节点。

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

3. 在 compute.yaml 文件中为 OVS-DPDK 绑定设置 MTU 值：

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

8.4. 为安全组配置防火墙

dataPlane 接口在有状态的防火墙中需要高程度的性能。为保护这些接口，请考虑将 lecommunications grade 防火墙部署为虚拟网络功能(VNF)。

通过将 NeutronOVSFirewallDriver 参数设置为 openvswitch，可以配置 controlPlane 接口。这会将 OpenStack 网络配置为使用基于流的 OVS 防火墙驱动程序。这是在 parameter_defaults 下的 network-environment.yaml 文件中设置。

例如：

```
parameter_defaults:
  NeutronOVSFirewallDriver: openvswitch
```

当使用 OVS 防火墙驱动程序时，务必要为数据平面接口禁用它。这可以通过 openstack port set 命令完成此操作。

例如：

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

8.5. 为 OVS-DPDK 接口设置多队列

要为 Compute 节点上的 Data Plane Development Kit (OVS-DPDK)中的接口设置相同的队列，请按如下所示修改 compute.yaml 文件：

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

8.6. 部署 OVERCLOUD

1. 确保 DPDK 计算角色的参数在 network-environment.yaml 中填充。如果需要，它们可以从派生的 OVS-DPDK 参数复制：

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

2. 使用 openstack overcloud deploy 命令部署 overcloud。

- 包含角色文件以及特定于您的环境的所有环境文件。

- 通过将 /usr/share/openstack-tripleo-heat-templates/environments 中的 host-config-and-reboot.yaml 文件从 /usr/share/openstack-tripleo-heat-templates/environments 中包含 host-config-and-reboot.yaml 文件来应用 KernelArgs 和 TunedProfile 参数：

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

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

8.7. 已知限制

使用用于 NFV 用例的 Red Hat OpenStack Platform 配置 OVS-DPDK 时有一些限制：

- 对 control plane 网络使用 Linux 绑定。确保绑定中使用的 PCI 设备都位于同一个 NUMA 节点上，以获得最佳性能。红帽不支持 Neutron Linux 网桥配置。
- 在带有 OVS-DPDK 的主机上运行每个实例都需要巨页。如果客户机中没有巨页，则接口会出现但无法正常工作。
- 使用 OVS-DPDK 时，服务性能降级使用 tap 设备，如分布式虚拟路由(DVR)。生成的性能不适用于生产环境。
- 使用 OVS-DPDK 时，请确保同一计算节点上所有网桥都是 ovs_user_bridge 类型。在同一节点上混合 ovs_bridge 和 ovs_user_bridge 会对性能造成影响，不被支持。

8.8. 创建类别并部署 OVS-DPDK 实例

使用 Data Plane Development Kit (OVS-DPDK)为带有 NFV 的 Red Hat OpenStack Platform 部署配置 Open vSwitch 后，您可以创建一个类别并部署一个实例：

1.

创建聚合组并为 OVS-DPDK 添加相关主机。定义与定义的类别元数据匹配的元数据，如 dpdk=true。

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

注意

您应该使用主机聚合将 CPU 固定实例与未固定实例分开。不使用 CPU 固定的实例不会遵循使用 CPU 固定的实例重新提供要求。

2. 创建类别。

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

3. 设置其他类别属性。请注意，定义的元数据 dpdk=true 与 DPDK 聚合中定义的元数据匹配。

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

有关性能改进的仿真程序线程策略的详情，请参阅：配置 Emulator Threads，以便在 Dedicated 物理 CPU 上运行。

4. 创建网络。

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

5. 可选：如果您将多队列用于 OVS-DPDK，请在要用于创建实例的镜像上设置 hw_vif_multiqueue_enabled 属性：

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

6. 部署实例。

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

8.9. 对配置进行故障排除

本节介绍了使用 Data Plane Development Kit (DPDK-OVS)配置对 Open vSwitch 进行故障排除的步骤。

1. 检查网桥配置，并确认网桥是使用 datapath_type=netdev 创建的。

```
# 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. 确认 docker 容器 neutron_ovs_agent 配置为自动启动。

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

如果容器启动时遇到问题，您可以查看相关的消息。

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

3. **确认 ovs-dpdk 的 PMD CPU 掩码已固定到 CPU。如果是 HT，请使用同级 CPU。**

   例如，使用 CPU4 ：

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

   使用 CPU 4 和 20：

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

   显示其状态：

```
# 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 9 章 调优 RED HAT OPENSTACK PLATFORM 环境

9.1. 可信虚拟功能

您可以配置物理功能(PF)来信任虚拟功能(VF)，以便 VF 能够执行一些特权操作。例如，您可以使用此配置来允许 VF 启用混杂模式或更改硬件地址。

9.1.1. 提供信任

前提条件

- 操作安装 Red Hat OpenStack Platform director

流程

完成以下步骤，使用必要的参数部署 overcloud，以启用虚拟功能的物理功能信任：

1. 在 parameter_defaults 部分下添加 NeutronPhysicalDevMappings 参数，以在逻辑网络名称和物理接口之间建立链接。

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

2. 将新属性"trusted"添加到与 SR-IOV 相关的现有参数。

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

注意

您必须包含 "true" 值的引号。

重要

仅在可信环境中完成以下步骤。此步骤将允许非管理员用户绑定可信端口。

3. 修改权限，以允许用户创建和更新端口绑定。

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

9.1.2. 使用可信虚拟功能

在完全部署的 **overcloud** 上执行以下命令，以利用可信的虚拟功能。

创建可信 **VF** 网络

1. 创建类型为 **vlan** 的网络。

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

2. 创建子网。

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

3. 创建一个端口，将 **vnic-type** 选项设置为 **direct**，并将 **binding-profile** 选项设置为 **true**。

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

4. 创建一个实例绑定，将其绑定到之前创建的可信端口。

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

验证 **hypervior** 上的可信虚拟功能配置

在托管新创建的实例的计算节点上，运行以下命令：

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

查看 **ip link** 命令的输出，并验证虚拟功能的信任状态 是否在 上信任。示例输出包含包含两个端口的环境的详细信息。请注意，**vf 6** 包含 上的文本信任。

9.2. 配置 RX/TX 队列大小

出于多种原因，您可以遇到每秒 **350**万数据包率的高数据包丢失，例如：

- 网络中断
- **a SMI**
- 虚拟功能中的数据包处理延迟

为防止数据包丢失，将队列大小从默认 **512** 增加到最大 **1024**。

前提条件

- 要配置 **RX**，请确保您有 **libvirt v2.3** 和 **QEMU v2.7**。
- 要配置 **TX**，请确保您有 **libvirt v3.7** 和 **QEMU v2.10**。

流程

- **要增加 RX 和 TX 队列大小，请在相关 director 角色的 parameter_defaults: 部分包括以下行：以下是 ComputeOvsDpdk 角色示例：**

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

测试

- **您可以在 nova.conf 文件中观察 RX 队列大小和 TX 队列大小的值：**

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

- **您可以在计算主机上由 libvirt 生成的 VM 实例 XML 文件中检查 RX 队列大小和 TX 队列大小。**

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

要验证 RX 队列大小和 TX 队列大小的值，请在 KVM 主机上使用以下命令：

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

- **您可以检查改进的性能，比如 3.8 mpps/core 在 0 帧丢失。**

9.3. 为 NFV 工作负载启用 RT-KVM

本节介绍了为 Red Hat OpenStack Platform 安装和配置 Red Hat Enterprise Linux 7.5 Real Time KVM (RT-KVM)的步骤。Red Hat OpenStack Platform 提供实时功能，提供一个新的实时 Compute 节

点角色，这些角色为 Real-Time 置备 Red Hat Enterprise Linux，以及额外的 RT-KVM 内核模块，以及自动配置 Compute 节点。

9.3.1. 规划您的 RT-KVM Compute 节点

您必须使用 Red Hat 认证的服务器作为 RT-KVM Compute 节点。详情请查看 Red Hat Enterprise Linux for Real Time 7 认证的服务器。

如需有关如何为 RT-KVM 启用 rhel-7-server-nfv-rpms 存储库的详细信息，请参阅 注册和更新 undercloud，并确保您的系统处于最新状态。

注意

您需要单独订阅 Red Hat OpenStack Platform for Real Time SKU，才能访问此软件仓库。

构建实时镜像

使用以下步骤为 Real-time Compute 节点构建 overcloud 镜像：

1. 要初始化 stack 用户以使用 director 命令行工具，请运行以下命令：

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

2. 在 undercloud 上安装 libguestfs-tools 软件包以获取 virt-customize 工具：

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

重要

如果在 undercloud 上安装 libguestfs-tools 软件包，请禁用 iscsid.socket 以避免与 undercloud 上的 tripleo_iscsid 服务冲突：

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

3. 提取镜像：

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

4. **复制默认镜像：**

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

5. **注册您的镜像，以启用与您的自定义相关的红帽软件仓库。在以下示例中，用有效的凭证替换 [username] 和 [password]。**

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

注意

在历史记录文件中删除凭证，只要它们用于命令提示符。您可以使用 history -d 命令（后跟行号）删除历史记录中的个别行。

6. **从您的帐户的订阅中查找池 ID 列表，并将适当的池 ID 附加到您的镜像。**

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

7. **添加带有 NFV 的 Red Hat OpenStack Platform 所需的存储库。**

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

8. **创建一个脚本，以在镜像上配置实时功能。**

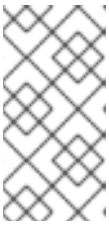
```
(undercloud) [stack@undercloud-0 ~]$ cat <<'EOF' > rt.sh
 #!/bin/bash
```

```
set -eux

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

9. 运行脚本以配置 RT 镜像：

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



注意

您可能会在 rt.sh 脚本输出中看到以下错误： **grubby fatal error: unable to find a suitable template**。您可以安全地忽略这个错误。

10. 您可以通过检查从上一个命令创建的 **virt-customize.log** 文件，检查使用 **rt.sh** 脚本安装的软件包。

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

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

11. 重新标记 **SELinux**：

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

12. 提取 **vmlinuz** 和 **initrd**：

注意

vmlinuz 和 initramfs 文件名中的软件版本因内核版本而异。在文件名中使用相关的软件版本，如 image/boot/vmlinuz-3.10.0-862.rt56.804.el7x86_64，或者使用通配符符号 *。

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

13. **上传镜像：**

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

现在，您可以在选择 Compute 节点上的 ComputeOvsDpdkRT 可组合角色中使用实时镜像。

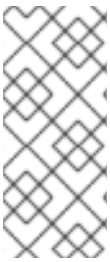
修改 RT-KVM Compute 节点上的 BIOS 设置

要减少 RT-KVM Compute 节点上的延迟，您必须修改 BIOS 设置。您应该在 Compute 节点 BIOS 设置中禁用以下所有选项：

- **电源管理**
- **超线程**
- **CPU 睡眠状态**
- **逻辑处理器**

有关这些设置的描述以及禁用它们的影响，请参阅设置 BIOS 参数。有关如何更改 BIOS 设置的详情，请查看您的硬件厂商文档。

9.3.2. 使用 RT-KVM 配置 OVS-DPDK



注意

您必须确定 network-environment.yaml 文件中设置的 OVS-DPDK 参数的最佳值，以优化 OVS-DPDK 的 OpenStack 网络。详情请查看 第 8.1 节 “使用工作流推断 DPDK 参数”。

9.3.2.1. 生成 ComputeOvsDpdk 可组合角色

您可以使用 ComputeOvsDpdkRT 角色指定使用实时计算节点的 Compute 节点。

为 ComputeOvsDpdkRT 角色生成 roles_data.yaml。

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

9.3.2.2. 配置 OVS-DPDK 参数

重要

尝试在没有适当值的情况下部署 Data Plane Development Kit (DPDK)会导致部署失败，或导致部署不稳定。您必须确定 network-environment.yaml 文件中设定的 OVS-DPDK 参数的最佳值，以优化 OVS-DPDK 的 OpenStack 网络。详情请查看 第 8.1 节 “使用工作流推断 DPDK 参数”。

1. 为您在 resource_registry 下使用的 OVS-DPDK 角色添加 nic 配置：

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

2. 在 parameter_defaults 下，设置 OVS-DPDK 和 RT-KVM 参数：

```
# DPDK compute node.
ComputeOvsDpdkRTParameters:
  KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
```

```
intel_iommu=on isolcpus=1-7,17-23,9-15,25-31"
    TunedProfileName: "realtime-virtual-host"
    IsolCpusList: "1-7,17-23,9-15,25-31"
    NovaVcpuPinSet: ['2-7,18-23,10-15,26-31']
    NovaReservedHostMemory: 4096
    OvsDpdkSocketMemory: "1024,1024"
    OvsDpdkMemoryChannels: "4"
    OvsDpdkCoreList: "0,16,8,24"
    OvsPmdCoreList: "1,17,9,25"
    VhostuserSocketGroup: "hugetlbfs"
  ComputeOvsDpdkRTImage: "overcloud-realtime-compute"
```

9.3.2.3. 准备容器镜像。

准备容器镜像：

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

9.3.2.4. 部署 overcloud

为 **ML2-OVS 部署 overcloud：**

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

9.3.3. 启动 RT-KVM 实例

要在启用了实时的 Compute 节点上启动 RT-KVM 实例：

1. **在 overcloud 上创建 RT-KVM 类别：**

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

2. 启动 **RT-KVM** 实例：

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

3. （可选）验证实例是否使用分配的仿真程序线程：

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

9.4. 配置一个 NUMA 感知的 VSWITCH （技术预览）

重要

该功能在此发行版本中作为*技术预览*提供，因此不享有红帽的全面支持。它只应用于测试，不应部署在生产环境中。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

在实现 **NUMAaware vSwitch** 前，检查硬件配置的以下组件：

- 物理网络的数量。
- PCI 卡的放置。

- 服务器的物理架构。

内存映射的 I/O (MMIO)设备（如 PCIe NIC）与特定 NUMA 节点相关联。当虚拟机和 NIC 位于不同的 NUMA 节点上时，性能会有很大降低。要提高性能，请在同一 NUMA 节点上对齐 PCIe NIC 放置和实例处理。

使用此功能来确保共享物理网络的实例位于同一 NUMA 节点上。要优化数据中心硬件，您可以使用多种网络、不同的网络类型或绑定来利用加载共享虚拟机。

重要

要正确架构 NUMA 节点负载共享和网络访问，您必须了解 PCIe 插槽和 NUMA 节点的映射。有关具体硬件的详情，请参考您的厂商文档。

为防止跨 NUMA 配置，请将虚拟机放在正确的 NUMA 节点上，方法是向 Nova 提供 NIC 的位置。

前提条件

- 您已启用过滤器 "NUMATopologyFilter"

流程

- 设置一个新的 NeutronPhysnetNUMANodesMapping 参数，将物理网络映射到与物理网络关联的 NUMA 节点。

- 如果使用隧道，如 VxLAN 或 GRE，还必须设置 NeutronTunnelNUMANodes 参数。

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

下面是一个与 NUMA 节点 0 连接的两个物理网络的示例：

- 与 NUMA 节点 0 关联的项目网络

- 个管理网络，没有任何关联性

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

测试

- 观察文件 /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf 中的配置

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

- 使用 lscpu 命令确认新配置：

```
$ lscpu
```

- 启动一个虚拟机，将 NIC 附加到适当的网络

9.5. 在 NFVI 环境中配置服务质量(QOS)

有关配置 QoS 的详情，请参阅配置 Real-Time Compute。支持仅限于 SR-IOV 和 OVS-DPDK 出口接口的 QoS 规则类型 bandwidth-limit。

9.6. 使用 HCI 和 DPDK 部署 OVERCLOUD

您可以通过共同定位和配置 Compute 和 Ceph Storage 服务以优化资源使用量来部署 NFV 基础架构。

有关超融合基础架构(HCI)的更多信息，请参阅： Hyper Converged Infrastructure Guide

前提条件

- Red Hat OpenStack Platform 13.12 维护阶段于 2019 年 12 月 19 日或更新版本。

- Ceph 12.2.12-79 （硬件）或更新.
- ceph-ansible 3.2.38 或更新版本。

流程

1. 在 undercloud 上安装 ceph-ansible。

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

2. 为 ComputeHCI 角色生成 roles_data.yaml 文件。

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

3. 使用 openstack flavor create 和 openstack flavor set 命令创建和配置一个新类别。有关创建类别的更多信息，*请参阅高级 Overcloud 自定义指南中的创建新角色。*

4. 使用您生成的自定义 roles_data.yaml 文件部署 overcloud。

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

9.6.1. NUMA 节点配置示例

为提高性能，请将租户网络和 Ceph 对象服务守护进程(OSD)放入一个 NUMA 节点，如 NUMA-0，以及 VNF 以及任何其他 NUMA 节点（如 NUMA-1）。

CPU 分配：

NUMA-0	NUMA-1
Ceph OSD 数量 * 4 HT	用于 VNF 和非NFV 虚拟机的客户机 vCPU
DPDK lcore - 2 HT	DPDK lcore - 2 HT
DPDK PMD - 2 HT	DPDK PMD - 2 HT

CPU 分配示例：

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

9.6.2. ceph 配置文件示例

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

```
  ceph_osd_numactl_opts: "-N 0 --preferred=0"                    # 3
CephAnsibleDisksConfig:
  osds_per_device: 1
  osd_scenario: lvm
  osd_objectstore: bluestore
  devices:
    - /dev/sda
    - /dev/sdb
    - /dev/sdc
```

使用下列参数，为 ceph OSD 进程分配 CPU 资源。根据此超融合环境中的工作负载和硬件调整值。

1

ceph_osd_docker_cpuset_cpus：为 SSD 磁盘分配 4 个 CPU 线程，或者为 HDD 磁盘为每个 OSD 分配 4 个 CPU 线程。包含与 ceph 关联的 NUMA 节点的内核数和同级线程，以及三个列表中未找到的 CPU：NovaVcpuPinSet、OvsDpdkCoreList 和 OvsPmdCoreList。

2

ceph_osd_docker_cpu_limit：将此值设置为 0， 以将 ceph OSD 固定到 ceph_osd_docker_cpuset_cpus 中的 CPU 列表。

3

ceph_osd_numactl_opts：将此值设置为跨 NUMA 操作 的首选 值，作为 precaution。

9.6.3. DPDK 配置文件示例

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

isolcpus=2,46,3,47,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,49,51,53,55,57,59,61,6
3,65,67,69,71,73,75,77,79,81,83,85,87"
    TunedProfileName: "cpu-partitioning"
    IsolCpusList:                                 # 2
      "2,46,3,47,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,49,51,
      53,55,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87"
    VhostuserSocketGroup: hugetlbfs
    OvsDpdkSocketMemory: "4096,4096"                        # 3
    OvsDpdkMemoryChannels: "4"

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

1

KernelArgs：要计算 大页，请从总内存中减去 NovaReservedHostMemory 参数的值。

2

IsolCpusList：分配您想要使用此参数与主机进程隔离的一组 CPU 核心。将 OvsPmdCoreList 参数的值添加到 NovaVcpuPinSet 参数的值，以计算 IsolCpusList 参数的值。

3

OvsDpdkSocketMemory：指定以 MB 为单位的内存大小，以预先从每个 NUMA 节点的巨页池中分配，使用 OvsDpdkSocketMemory 参数进行预先分配。有关计算 OVS-DPDK 参数的更多信息，请参阅： ovsdpdk 参数

4

OvsPmdCoreList：指定用于 DPDK 轮询模式驱动程序(PMD)的 CPU 核心。选择与 DPDK 接口本地 NUMA 节点关联的 CPU 内核。为每个 NUMA 节点分配 2 HT 同级线程，以计算 OvsPmdCoreList 参数的值。

5

OvsDpdkCoreList：使用此参数为非数据路径 OVS-DPDK 进程（如 handler 和 revalidator 线程）指定 CPU 核心。为每个 NUMA 节点分配 2 HT 同级线程，以计算 OvsDpdkCoreList 参数的值。

9.6.4. nova 配置文件示例

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

1

NovaReservedHugePages：从带有 NovaReservedHugePages 参数的巨页池中分配内存（以 MB 为单位）。它与 OvsDpdkSocketMemory 参数的值总量相同。

2

- 每个 OSD 5 GB。
- 每个虚拟机的 0.5 GB 开销.
- 4GB 用于常规主机处理。确保您分配足够的内存，以防止通过跨 NUMA OSD 操作导致潜在的性能下降。

3

NovaVcpuPinSet：使用 NovaVcpuPinSet 参数列出 OvsPmdCoreList 中没有找到的 CPU、OvsDpdkCoreList 或 Ceph_osd_docker_cpuset_cpus。CPU 必须与 DPDK NIC 位于同一个 NUMA 节点。

9.6.5. 建议配置 HCI-DPDK 部署

表 9.1. 用于 HCI 部署的可调参数

块设备类型	OSD、内存、vCPU 每个设备
NVMe	内存：每个 OSD 5GB 每个设备的 OSD: 4 个 vCPU：3 个
SSD	内存：每个 OSD 5GB 每个设备的 OSD: 1 个 个 vCPU：4 个
HDD	内存 : 每个 OSD 5GB 每个设备的 OSD: 1 个 个 vCPU: 1

对以下功能使用相同的 NUMA 节点：

- 磁盘控制器
- 存储网络
- 存储 CPU 和内存

为 **DPDK 提供商网络的以下功能分配另一个 NUMA 节点：**

- **NIC**
- **PMD CPU**
- **套接字内存**

第 10 章 示例：使用 VXLAN 隧道配置 OVS-DPDK 和 SR-IOV

本节论述了如何使用 OVS-DPDK 和 SR-IOV 接口部署 Compute 节点。集群将安装有 ML2/OVS 和 VXLAN 隧道。

重要

您必须确定 network-environment.yaml 文件中设置的 OVS-DPDK 参数的最佳值，以优化 OVS-DPDK 的 OpenStack 网络。详情请参阅使用 工作流 Deriving DPDK 参数。

10.1. 配置角色

通过复制并编辑 /usr/share/openstack-tripleo-heat-templates 中找到的默认 roles_data.yaml 文件来配置自定义角色。

在本例中，会创建 ComputeOvsDpdkSriov 角色。有关在 Red Hat OpenStack Platform 中创建角色的详情，请参考 高级 Overcloud 自定义。有关本例使用的特定角色的详情，请参阅 roles_data.yaml。

10.2. 配置 OVS-DPDK 参数

重要

您必须确定 network-environment.yaml 文件中设置的 OVS-DPDK 参数的最佳值，以优化 OVS-DPDK 的 OpenStack 网络。有关详细信息，请参阅 网络功能虚拟化规划和配置。

1. 将 OVS-DPDK 的自定义资源添加到 resource_registry 下：

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

2. 在 parameter_defaults 下，将隧道类型和网络类型设置为 vxlan。

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

3. 在 parameters_defaults 下，设置网桥映射：

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

4. 在 parameter_defaults 下，为 ComputeOvsDpdkSriov 角色设置特定于角色的参数：

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

注意

要防止创建客户机期间失败，请在每个 NUMA 节点上至少分配一个具有同级线程的 CPU。在示例中，OvsPmdCoreList 参数的值代表来自 NUMA 0 的核心 2 和 22，以及来自 NUMA 1 的核心 3 和 23。

注意

巨页由虚拟机消耗，以及使用 OvsDpdkSocketMemory 参数的 OVS-DPDK。要计算虚拟机的巨页数量，请从 boot 参数值中减去 OvsDpdkSocketMemory 值。您还必须将 hw:mem_page_size=1GB 添加到与 DPDK 实例关联的类别中。

注意

OvsDPDKCoreList 和 OvsDpdkMemoryChannels 是必需的，必须正确设置才能防止失败。

5. 为 **SR-IOV 配置特定于角色的参数：**

```
  ############################
  #  SR-IOV configuration  #
  ############################
  NeutronMechanismDrivers: ['openvswitch','sriovnicswitch']
  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFil
ter','ImagePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','P
ciPassthroughFilter','NUMATopologyFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthrough_filter.PciP
assthroughFilter"]
  NovaPCIPassthrough:
    - vendor_id: "8086"
      product_id: "1528"
      address: "0000:06:00.0"
      trusted: "true"
      physical_network: "sriov-1"
    - vendor_id: "8086"
      product_id: "1528"
      address: "0000:06:00.1"
      trusted: "true"
      physical_network: "sriov-2"
```

10.3. 配置 CONTROLLER 节点

1. 为隔离的网络创建 **control plane Linux 绑定。**

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

2. 将 **VLAN 分配给此 Linux 绑定。**

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

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

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

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

3. 创建 OVS 网桥，以访问 neutron-dhcp-agent 和 neutron-metadata-agent 服务。

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

10.4. 为 DPDK 和 SR-IOV 配置 COMPUTE 节点

从默认的 compute.yaml 文件创建 computeovsdpdksriov.yaml 文件，并进行以下更改：

1. 为隔离的网络创建 control plane Linux 绑定。

```
- 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. 将 VLAN 分配给此 Linux 绑定。

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

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

3. 使用 DPDK 端口设置网桥以链接到控制器。

```
- type: ovs_user_bridge
  name: br-link0
  use_dhcp: false
  ovs_extra:
    - str_replace:
        template: set port br-link0 tag=_VLAN_TAG_
        params:
          _VLAN_TAG_:
```

```
          get_param: TenantNetworkVlanID
      addresses:
        - ip_netmask:
            get_param: TenantIpSubnet
      members:
        - type: ovs_dpdk_bond
          name: dpdkbond0
          mtu: 9000
          rx_queue: 2
          members:
            - type: ovs_dpdk_port
              name: dpdk0
              members:
                - type: interface
                  name: nic7
            - type: ovs_dpdk_port
              name: dpdk1
              members:
                - type: interface
                  name: nic8
```

注意

要包含多个 DPDK 设备，请为您要添加的每个 DPDK 设备重复 类型 代码部分。

注意

在使用 OVS-DPDK 时，同一计算节点上的 所有 网桥都应是 ovs_user_bridge 类型。director 可以接受配置，但 Red Hat OpenStack Platform 不支持同一节点上的 ovs_bridge 和 ovs_user_bridge。

10.5. 部署 OVERCLOUD

运行 overcloud_deploy.sh 脚本来部署 overcloud。

```
#!/bin/bash

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

```
hybrid/overcloud_images.yaml \
-e /home/stack/ospd-13-vxlan-dpdk-sriov-ctlplane-dataplane-bonding-hybrid/network-
environment.yaml \
--log-file overcloud_install.log &> overcloud_install.log
```

第 11 章 使用 NFV 升级红帽 OPENSTACK 平台

有关使用配置 OVS-DPDK 升级 Red Hat OpenStack Platform (RHOSP)的更多信息，请参阅 框架中的准备网络功能虚拟化(NFV) *升级(13 至 16.1)* 指南。有关升级 RHOSP 的更多信息，请参阅升级 Red Hat OpenStack Platform 指南。

第 12 章 性能

Red Hat OpenStack Platform director 配置 Compute 节点，以强制资源分区和微调，以实现客户机虚拟网络功能(VNF)的行率性能。网络功能虚拟化(NFV)用例中的主要性能因素是吞吐量、延迟和严重程度。

data plane 开发套件(DPDK)加速 Open vSwitch (OVS)实现物理 NIC 和虚拟机之间的高性能数据包切换。OVS 2.7 嵌入了 DPDK 16.11 的支持，并包含对 vhost-user 多队列的支持，从而实现可扩展的性能。OVS-DPDK 为客户端 VNF 提供行速率性能。

单根 I/O 虚拟化(SR-IOV)网络提供了增强的性能特征，包括为特定网络和虚拟机改进吞吐量。

性能调优的其他重要功能包括大页面、NUMA 对齐、主机隔离和 CPU 固定。VNF 类别需要巨页和仿真程序线程隔离才能提高性能。主机隔离和 CPU 固定功能可提高 NFV 性能，并防止出现大量数据包丢失。

如需了解 CPU 和 NUMA 拓扑的高级介绍，请参阅 NFV 性能注意事项和配置 Emulator Threads 在 Dedicated 物理 CPU 上运行。

第 13 章 查找更多信息

下表提供了其他 Red Hat 文档供参考：

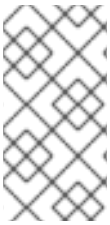
Red Hat OpenStack Platform 文档套件可在以下文档中找到： Red Hat OpenStack Platform 文档套件

表 13.1. 可用文档列表

组件	参考
Red Hat Enterprise Linux	Red Hat OpenStack Platform 需要运行在 Red Hat Enterprise Linux 7.4 上。有关安装 Red Hat Enterprise Linux 的详情，请参考相应的安装指南： Red Hat Enterprise Linux 文档套件。
Red Hat OpenStack Platform	要安装 OpenStack 组件及其依赖项，请使用 Red Hat OpenStack Platform director。director 使用基本的 OpenStack 安装作为 undercloud 来安装、配置和管理最终 overcloud 中的 OpenStack 节点。请注意，除了部署的 overcloud 所需的环境之外，还需要一台额外的主机来安装 undercloud。具体步骤请查看 Red Hat OpenStack Platform Director 安装和使用。 有关使用 Red Hat OpenStack Platform director （如网络隔离、存储配置、SSL 通信和常规配置方法）为 Red Hat OpenStack Platform 环境配置高级功能的详情，请参考 高级 Overcloud 自定义。
NFV 文档	有关 NFV 概念的高级概述，请参阅 网络功能虚拟化产品指南。

附录 A. DPDK SRIOV YAML 文件示例

本节提供了示例 YAML 文件，作为在同一计算节点上添加单一根 I/O 虚拟化(SR-IOV)和 Data Plane Development Kit (DPDK)接口的参考。



注意

这些模板来自完全配置的环境中，包含与 NFV 相关的参数，它们可能不相关或适合您的部署。

A.1. VXLAN DPDK SR-IOV YAML 文件示例

A.1.1. roles_data.yaml

```
###############################################################################
# File generated by TripleO                                                   #
###############################################################################
###############################################################################
# Role: Controller                                                            #
###############################################################################
- name: Controller
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
  # For systems with both IPv4 and IPv6, you may specify a gateway network for
  # each, such as ['ControlPlane', 'External']
  default_route_networks: ['External']
  HostnameFormatDefault: 'controller-%index%'
  # Deprecated & backward-compatible values (FIXME: Make parameters consistent)
  # Set uses_deprecated_params to True if any deprecated params are used.
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
```

```
- OS::TripleO::Services::AodhListener
- OS::TripleO::Services::AodhNotifier
- OS::TripleO::Services::AuditD
- OS::TripleO::Services::BarbicanApi
- OS::TripleO::Services::BarbicanBackendSimpleCrypto
- OS::TripleO::Services::BarbicanBackendDogtag
- OS::TripleO::Services::BarbicanBackendKmip
- OS::TripleO::Services::BarbicanBackendPkcs11Crypto
- OS::TripleO::Services::CACerts
- OS::TripleO::Services::CeilometerApi
- OS::TripleO::Services::CeilometerCollector
- OS::TripleO::Services::CeilometerExpirer
- OS::TripleO::Services::CeilometerAgentCentral
- OS::TripleO::Services::CeilometerAgentNotification
- OS::TripleO::Services::CephExternal
- OS::TripleO::Services::CephMds
- OS::TripleO::Services::CephMgr
- OS::TripleO::Services::CephMon
- OS::TripleO::Services::CephRbdMirror
- OS::TripleO::Services::CephRgw
- OS::TripleO::Services::CertmongerUser
- OS::TripleO::Services::CinderApi
- OS::TripleO::Services::CinderBackendDellPs
- OS::TripleO::Services::CinderBackendDellSc
- OS::TripleO::Services::CinderBackendDellEMCUnity
- OS::TripleO::Services::CinderBackendDellEMCVMAXISCSI
- OS::TripleO::Services::CinderBackendDellEMCVNX
- OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI
- OS::TripleO::Services::CinderBackendNetApp
- OS::TripleO::Services::CinderBackendScaleIO
- OS::TripleO::Services::CinderBackendVRTSHyperScale
- OS::TripleO::Services::CinderBackup
- OS::TripleO::Services::CinderHPELeftHandISCSI
- OS::TripleO::Services::CinderScheduler
- OS::TripleO::Services::CinderVolume
- OS::TripleO::Services::Clustercheck
- OS::TripleO::Services::Collectd
- OS::TripleO::Services::Congress
- OS::TripleO::Services::Docker
- OS::TripleO::Services::Ec2Api
- OS::TripleO::Services::Etcd
- OS::TripleO::Services::ExternalSwiftProxy
- OS::TripleO::Services::Fluentd
- OS::TripleO::Services::GlanceApi
- OS::TripleO::Services::GlanceRegistry
- OS::TripleO::Services::GnocchiApi
- OS::TripleO::Services::GnocchiMetricd
- OS::TripleO::Services::GnocchiStatsd
- OS::TripleO::Services::HAproxy
- OS::TripleO::Services::HeatApi
- OS::TripleO::Services::HeatApiCloudwatch
- OS::TripleO::Services::HeatApiCfn
- OS::TripleO::Services::HeatEngine
- OS::TripleO::Services::Horizon
- OS::TripleO::Services::Ipsec
- OS::TripleO::Services::IronicApi
```

```
- OS::TripleO::Services::IronicConductor
- OS::TripleO::Services::IronicPxe
- OS::TripleO::Services::Iscsid
- OS::TripleO::Services::Keepalived
- OS::TripleO::Services::Kernel
- OS::TripleO::Services::Keystone
- OS::TripleO::Services::LoginDefs
- OS::TripleO::Services::ManilaApi
- OS::TripleO::Services::ManilaBackendCephFs
- OS::TripleO::Services::ManilaBackendIsilon
- OS::TripleO::Services::ManilaBackendNetapp
- OS::TripleO::Services::ManilaBackendUnity
- OS::TripleO::Services::ManilaBackendVNX
- OS::TripleO::Services::ManilaBackendVMAX
- OS::TripleO::Services::ManilaScheduler
- OS::TripleO::Services::ManilaShare
- OS::TripleO::Services::Memcached
- OS::TripleO::Services::MistralApi
- OS::TripleO::Services::MistralEngine
- OS::TripleO::Services::MistralExecutor
- OS::TripleO::Services::MistralEventEngine
- OS::TripleO::Services::MongoDb
- OS::TripleO::Services::MySQL
- OS::TripleO::Services::MySQLClient
- OS::TripleO::Services::NeutronApi
- OS::TripleO::Services::NeutronBgpVpnApi
- OS::TripleO::Services::NeutronSfcApi
- OS::TripleO::Services::NeutronCorePlugin
- OS::TripleO::Services::NeutronDhcpAgent
- OS::TripleO::Services::NeutronL2gwAgent
- OS::TripleO::Services::NeutronL2gwApi
- OS::TripleO::Services::NeutronL3Agent
- OS::TripleO::Services::NeutronLbaasv2Agent
- OS::TripleO::Services::NeutronLbaasv2Api
- OS::TripleO::Services::NeutronLinuxbridgeAgent
- OS::TripleO::Services::NeutronMetadataAgent
- OS::TripleO::Services::NeutronML2FujitsuCfab
- OS::TripleO::Services::NeutronML2FujitsuFossw
- OS::TripleO::Services::NeutronOvsAgent
- OS::TripleO::Services::NeutronVppAgent
- OS::TripleO::Services::NovaApi
- OS::TripleO::Services::NovaConductor
- OS::TripleO::Services::NovaConsoleauth
- OS::TripleO::Services::NovaIronic
- OS::TripleO::Services::NovaMetadata
- OS::TripleO::Services::NovaPlacement
- OS::TripleO::Services::NovaScheduler
- OS::TripleO::Services::NovaVncProxy
- OS::TripleO::Services::Ntp
- OS::TripleO::Services::ContainersLogrotateCrond
- OS::TripleO::Services::OctaviaApi
- OS::TripleO::Services::OctaviaDeploymentConfig
- OS::TripleO::Services::OctaviaHealthManager
- OS::TripleO::Services::OctaviaHousekeeping
- OS::TripleO::Services::OctaviaWorker
- OS::TripleO::Services::OVNDBs
```

```
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::SkydiveAnalyzer
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::SwiftProxy
    - OS::TripleO::Services::SwiftDispersion
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::Ptp
  ##############################################################################
  # Role: ComputeOvsDpdkSriov                                                  #
  ##############################################################################
  - name: ComputeOvsDpdkSriov
    description: |
      Compute OvS DPDK+SR-IOV Role
    CountDefault: 1
    networks:
      - InternalApi
      - Tenant
      - Storage
    HostnameFormatDefault: 'compute-%index%'
    disable_upgrade_deployment: True
    ServicesDefault:
      - OS::TripleO::Services::Aide
      - OS::TripleO::Services::AuditD
      - OS::TripleO::Services::CACerts
      - OS::TripleO::Services::CephClient
      - OS::TripleO::Services::CephExternal
      - OS::TripleO::Services::CertmongerUser
      - OS::TripleO::Services::Collectd
      - OS::TripleO::Services::ComputeCeilometerAgent
      - OS::TripleO::Services::ComputeNeutronCorePlugin
      - OS::TripleO::Services::ComputeNeutronL3Agent
      - OS::TripleO::Services::ComputeNeutronMetadataAgent
      - OS::TripleO::Services::ComputeNeutronOvsDpdk
      - OS::TripleO::Services::Docker
      - OS::TripleO::Services::Fluentd
      - OS::TripleO::Services::Ipsec
```

```
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::NeutronSriovHostConfig
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
  # Currently when attempting to deploy firewall service it notifies that it was specified multiple
times
  # - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Ptp
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
```

A.1.2. network-environment.yaml

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

parameter_defaults:
  # Customize all these values to match the local environment
  InternalApiNetCidr: 10.10.10.0/24
  TenantNetCidr: 10.10.2.0/24
  StorageNetCidr: 10.10.3.0/24
  StorageMgmtNetCidr: 10.10.4.0/24
  ExternalNetCidr: 172.20.12.112/28
  # CIDR subnet mask length for provisioning network
  ControlPlaneSubnetCidr: '24'
  InternalApiAllocationPools: [{'start': '10.10.10.10', 'end': '10.10.10.200'}]
  TenantAllocationPools: [{'start': '10.10.2.100', 'end': '10.10.2.200'}]
  StorageAllocationPools: [{'start': '10.10.3.100', 'end': '10.10.3.200'}]
  StorageMgmtAllocationPools: [{'start': '10.10.4.100', 'end': '10.10.4.200'}]
  # Use an External allocation pool which will leave room for floating IPs
  ExternalAllocationPools: [{'start': '172.20.12.114', 'end': '172.20.12.125'}]
  # Set to the router gateway on the external network
```

```
  ExternalInterfaceDefaultRoute: 172.20.12.126
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.168.24.1
  # Generally the IP of the Undercloud
  EC2MetadataIp: 192.168.24.1
  InternalApiNetworkVlanID: 10
  TenantNetworkVlanID: 11
  StorageNetworkVlanID: 12
  StorageMgmtNetworkVlanID: 13
  ExternalNetworkVlanID: 14
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["8.8.8.8","8.8.4.4"]
  # May set to br-ex if using floating IPs only on native VLAN on bridge br-ex
  NeutronExternalNetworkBridge: """"
  # The tunnel type for the project network (vxlan or gre). Set to '' to disable tunneling.
  NeutronTunnelTypes: 'vxlan'
  # The project network type for Neutron (vlan or vxlan).
  NeutronNetworkType: 'vxlan,vlan'
  # The OVS logical->physical bridge mappings to use.
  NeutronBridgeMappings: 'dpdk-mgmt:br-link0'
  # The Neutron ML2 and OpenVSwitch vlan mapping range to support.
  NeutronNetworkVLANRanges: 'dpdk-mgmt:22:22,dpdk-mgmt:25:25,sriov-1:600:600,sriov-
2:601:601'
  # Nova flavor to use.
  OvercloudControllerFlavor: controller
  OvercloudComputeOvsDpdkSriovFlavor: compute
  #Number of nodes to deploy.
  ControllerCount: 3
  ComputeOvsDpdkSriovCount: 2
  # NTP server configuration.
  NtpServer: clock.redhat.com
  # MTU global configuration
  NeutronGlobalPhysnetMtu: 9000
  # Configure the classname of the firewall driver to use for implementing security groups.
  NeutronOVSFirewallDriver: openvswitch
  SshServerOptions:
    UseDns: 'no'

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

  ############################
```

```
  # SR-IOV configuration #
  ##############################
  NeutronMechanismDrivers: ['openvswitch','sriovnicswitch']
  NovaSchedulerDefaultFilters:
['RetryFilter','AvailabilityZoneFilter','RamFilter','ComputeFilter','ComputeCapabilitiesFilter','Im
agePropertiesFilter','ServerGroupAntiAffinityFilter','ServerGroupAffinityFilter','PciPassthrough
Filter','NUMATopologyFilter']
  NovaSchedulerAvailableFilters:
["nova.scheduler.filters.all_filters","nova.scheduler.filters.pci_passthrough_filter.PciPassthro
ughFilter"]
  NovaPCIPassthrough:
    - vendor_id: "8086"
      product_id: "1528"
      address: "0000:06:00.0"
      physical_network: "sriov-1"
    - vendor_id: "8086"
      product_id: "1528"
      address: "0000:06:00.1"
      physical_network: "sriov-2"
  NeutronPhysicalDevMappings:
    - sriov1:p7p3
    - sriov2:p7p4
  NeutronSriovNumVFs: "p7p3:5,p7p4:5"
```

A.1.3. controller.yaml

```
heat_template_version: queens

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

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  StorageIpSubnet:
    default: ''
```

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

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
```

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

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

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

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

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

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

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

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

A.1.4. compute-ovs-dpdk.yaml

```
heat_template_version: queens

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

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

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

```
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-
config.sh
          params:
            $network_config:
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                defroute: false

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

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

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

```
            get_param: InternalApiIpSubnet

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

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

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

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

outputs:
```

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

A.1.5. overcloud_deploy.sh

```
#!/bin/bash

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