[**Neutron 理解 (1): Neutron 所实现的虚拟化网络 [How Netruon Virtualizes Network]**](http://www.cnblogs.com/sammyliu/p/4622563.html)

学习 Neutron 系列文章：

（1）[Neutron 所实现的虚拟化网络](http://www.cnblogs.com/sammyliu/p/4622563.html%20)

（2）[Neutron OpenvSwitch + VLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4626419.html%20)

（3）[Neutron OpenvSwitch + GRE/VxLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4627230.html%20)

（4）[Neutron OVS OpenFlow 流表 和 L2 Population](http://www.cnblogs.com/sammyliu/p/4633814.html%20)

（5）[Neutron DHCP Agent](http://www.cnblogs.com/sammyliu/p/4419195.html)

（6）[Neutron L3 Agent](http://www.cnblogs.com/sammyliu/p/4636091.html%20)

（7）[Neutron LBaas](http://www.cnblogs.com/sammyliu/p/4656176.html%20)

（8）[Neutron Security Group](http://www.cnblogs.com/sammyliu/p/4658746.html%20)

（9）[Neutron FWaas 和 Nova Security Group](http://www.cnblogs.com/sammyliu/p/4675991.html)

（10）[Neutron VPNaas](http://www.cnblogs.com/sammyliu/p/4677386.html%20)

（11）[Neutron DVR](http://www.cnblogs.com/sammyliu/p/4713562.html)

（12）[Neutron VRRP](http://www.cnblogs.com/sammyliu/p/4692081.html)

（13）[High Availability （HA）](http://www.cnblogs.com/sammyliu/p/4741967.html)

**1. 为什么要网络虚拟化？**

个人认为，这里主要有两个需求：一个是数据中心的现有网络不能满足云计算的物理需求；另一个是数据中心的现有网络不能满足云计算的软件化即SDN要求。

**1.1 现有物理网络不能满足云计算的需求**

    互联网行业数据中心的基本特征就是服务器的规模偏大。进入云计算时代后，其业务特征变得更加复杂，包括：虚拟化支持、多业务承载、资源灵活调度等（如图1所示）。与此同时，互联网云计算的规模不但没有缩减，反而更加庞大。这就给云计算的网络带来了巨大的压力。



图1. 互联网云计算的业务特点

（1）**大容量的MAC表项和ARP表项**

    虚拟化会导致更大的MAC表项。假设一个互联网云计算中心的服务器有5000台，按照1:20的比例进行虚拟化，则有10万个虚拟机。通常每个虚拟机会配置两个业务网口，这样这个云计算中心就有20万个虚拟网口，对应的就是需要20万个MAC地址和IP地址。云计算要求资源灵活调度，业务资源任意迁移。也就是说任意一个虚拟机可以在整个云计算网络中任意迁移。这就要求全网在一个统一的二层网络中。全网任意交换机都有可能学习到全网所有的MAC表项。与此对应的则是，目前业界主流的接入交换机的MAC表项只有32K，基本无法满足互联网云计算的需求。另外，网关需要记录全网所有主机、所有网口的ARP信息。这就需要网关设备的有效ARP表项超过20万。大部分的网关设备芯片都不具备这种能力。

（2）**4K VLAN Trunk问题**

传统的大二层网络支持任意VLAN的虚拟机迁移到网络的任意位置，一般有两种方式。方式一：虚拟机迁移后，通过自动化网络管理平台动态的在虚拟机对应的所有端口上下发VLAN配置；同时，还需要动态删除迁移前虚拟机对应所有端口上的VLAN配置。这种方式的缺点是实现非常复杂，同时自动化管理平台对多厂商设备还面临兼容性的问题，所以很难实现。方式二：在云计算网络上静态配置VLAN，在所有端口上配置VLAN trunk all。这种方式的优点是非常简单，是目前主流的应用方式。但这也带来了巨大的问题：任一VLAN内如果出现广播风暴，则全网所有VLAN内的虚拟机都会受到风暴影响，出现业务中断。

（3）**4K VLAN上限问题**

云计算网络中有可能出现多租户需求。如果租户及业务的数量规模超出VLAN的上限（4K），则无法支撑客户的需求。

（4）**虚拟机迁移网络依赖问题**

VM迁移需要在同一个二层域内，基于IP子网的区域划分限制了二层网络连通性的规模。

（[资料来源](http://www.h3c.com.cn/About_H3C/Company_Publication/IP_Lh/2014/07/Home/Catalog/201501/852548_30008_0.htm)）

**1.2 云计算的 SDN 要求**

    数据中心（Data Center）中的物理网络是固定的、需要手工配置的、单一的、没有多租户隔离的网络。这是一个物理网络局部实例：

（图1， 来源：[Cisco 网站](http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/OpenStack/Grizzly/OpenStack_Red_Hat_RDO.html)。 TOR：Top On Rack 交换机。图中的一个服务器有三块网卡，分别连到连接数据网络和管理网络的交换机。）

    而云架构往往是多租户架构，这意味着多个客户会共享单一的物理网络。因此，除了提供基本的网络连接能力以外，云还需要提供网络在租户之间的隔离能力；同时云是自服务的，这意味着租户可以通过云提供的 API 来使用虚拟出的网络组建来设计，构建和部署各种他们需要的网络。

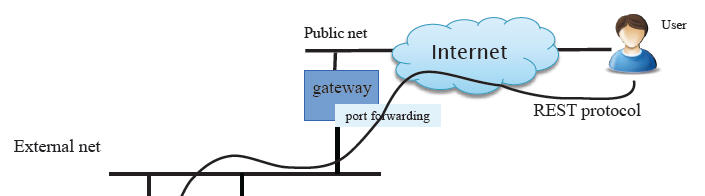
（左图-图3： 计算虚拟化和网络虚拟化。[来源](http://bradhedlund.com/2013/05/28/what-is-network-virtualization/)）                                                 （右图-图4：[来源](http://bradhedlund.com/2012/10/06/mind-blowing-l2-l4-network-virtualization-by-midokura-midonet/) MidoNet 提供的物理网络和逻辑网络的映射）

   OpenStack 云也不例外。 OpenStack 通过 Neutron 项目在物理网络环境之上提供满足多租户要求的虚拟网络和服务。Neutron 提供的网络虚拟化能力包括：

* （1）二层到七层网络的虚拟化：L2（virtual switch）、L3（virtual Router 和 LB）、L4-7（virtual Firewall ）等
* （2）网络连通性：二层网络和三层网络
* （3）租户隔离性
* （4）网络安全性
* （4）网络扩展性
* （5）REST API
* （6）更高级的服务，包括 LBaaS，[FWaaS](https://wiki.openstack.org/wiki/Neutron/FWaaS)，VPNaaS 等。具体以后再慢慢分析。

**2. Neutron 网络虚拟化**

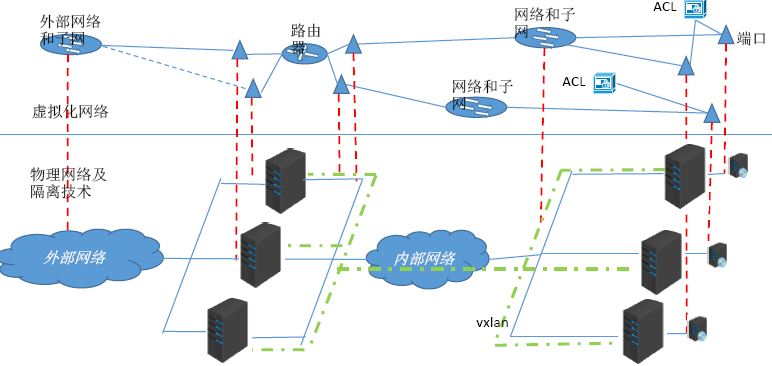
    在实际的数据中心中，网络可以分为三层：OpenStack Cloud network，机房intranet （external network），以及真正的外部网络即 Internet。External 网络和Internet 之间是数据中心的 Uplink 路由器，它负责通过 NAT 来进行两个网络之间的IP地址（即 floating IP 和 Internet/Public IP）转换，因此，这部分的控制不在 OpenStack 控制之下，也不在本文的主要探讨范围之内。



* OpenSack Cloud network：OpenStack 所管理的网络。
* External network：数据中心所管理的的公司网（Intranet） ，虚机使用的 Floating IP 是这个网络的地址的一部分。
* Internet：由各大电信运营商所管理的公共网络，使用公共IP。

这是 RedHat 提供的一个 OpenStack Cloud network 网络架构：

大概地分类的话，该网络管理网络 和 数据网络，数据网络中关键的是租户网络，用于租户虚机之间的通信。这部分也是 Neutron 所实现的网络虚拟化的核心。在目前的Neutron 实现中，Neutron 向租户提供虚拟的网络（network）、子网（subnet）、端口 （port）、交换机（switch）、路由器（router）等网络组件。下图显示了虚拟网络和物理网络的映射关系：



（图5.来源：2015 OpenStack技术大会-Neutron云计算网络虚拟化-龚永生.pdf）

**2.1 网络（L2 network）**

    网络（network）是一个隔离的二层网段，类似于物理网络世界中的虚拟 LAN (VLAN)。更具体来讲，它是为创建它的租户而保留的一个广播域，或者被显式配置为共享网段。端口和子网始终被分配给某个特定的网络。这里所谓的隔离，可以理解为几个含义：

* 跨网络的子网之间的流量必须走 L3 Virtual Router
* 每个网络使用自己的 DHCP Agent，每个 DHCP Agent 在一个 Network namespace 内
* 不同网络内的IP地址可以重复（overlapping）

根据创建网络的用户的权限，Neutron network 可以分为：

* Provider network：管理员创建的和物理网络有直接映射关系的虚拟网络。
* Tenant network：租户普通用户创建的网络，物理网络对创建者透明，其配置由 Neutron根据管理员在系统中的配置决定。

虚机可以直接挂接到 provider network 或者 tenant network 上  “VMs can attach directly to both tenant and provider networks, and to networks with any provider:network\_type value, assuming their tenant owns the network or its shared.”。

根据网络的类型，Neutron network 可以分为：

* VLAN network（虚拟局域网） ：基于物理 VLAN 网络实现的虚拟网络。共享同一个物理网络的多个 VLAN 网络是相互隔离的，甚至可以使用重叠的 IP 地址空间。每个支持 VLAN network 的物理网络可以被视为一个分离的 VLAN trunk，它使用一组独占的 VLAN ID。有效的 VLAN ID 范围是 1 到 4094。
* Flat network：基于不使用 VLAN 的物理网络实现的虚拟网络。每个物理网络最多只能实现一个虚拟网络。
* local network（本地网络）：一个只允许在本服务器内通信的虚拟网络，不知道跨服务器的通信。主要用于单节点上测试。
* GRE network （通用路由封装网络）：一个使用 GRE 封装网络包的虚拟网络。GRE 封装的数据包基于 IP 路由表来进行路由，因此 GRE network 不和具体的物理网络绑定。
* VXLAN network（虚拟可扩展网络）：基于 VXLAN 实现的虚拟网络。同 GRE network 一样， VXLAN network 中 IP 包的路由也基于 IP 路由表，也不和具体的物理网络绑定。

注：在AWS中，该概念对应 VPC 概念。AWS 对 VPC 的数目有一定的限制，比如每个账户在每个 region 上默认最多只能创建 5 个VPC，通过特别的要求最多可以创建 100 个。

**2.1.1 Provider network**

   Provider Network 是由 OpenStack 管理员创建的，直接对应于数据中心的已有物理网络的一个网段。这种网络有三个和物理网络有关属性：

* provider:network\_type (网络类型，包括 vxlan, gre, vlan, flat, local)
* provider:segmentation\_id (网段 ID， 比如 VLAN 的 802.1q tag, GRE 网络的 Tunnel ID, VXLAN 网络的 VNI)
* provider:physical\_network (物理网络的逻辑名称，比如 physnet1, ph-eth1, etc)

  这种网络是可以在多个租户之间共享的。这种网络通过计算和网络节点上指定的 bridge 直接接入物理网络，所以默认的情况下它们是可以路由的，因此也不需要接入 Neutron Virtual Router，也不需要使用 L3 agent。使用这种网络，必须预先在各计算和网络节点上配置指定的网桥。

  虽然可以创建 GRE 和 VXLAN 类型的 Provider network， 但是（个人认为）Provider network 只对 Flat 和 VLAN 类型的网络才有意义，因为 Provider network 的一个重要属性是 provider:physical\_network，而这个参数对其他网络类型没有有意义。

创建 provider network：

* local 类型的：neutron net-create NAME --provider:network\_type local
* flat 类型的：neutron net-create NAME --provider:network\_type flat --provider:physical\_network PHYS\_NET\_NAME
* vlan 类型的：neutron net-create NAME --provider:network\_type vlan --provider:physical\_network PHYS\_NET\_NAME --provider:segmentation\_id VID
* gre 类型的：neutron net-create NAME --provider:network\_type gre --provider:segmentation\_id TUNNEL\_ID
* vxlan 类型的：neutron net-create NAME --provider:network\_type vxlan --provider:segmentation\_id TUNNEL\_ID

**2.1.3 Tenant network**

    Tenant network 是由 tenant 的普通用户创建的网络。默认情况下，这类用户不能创建共享的 tenant network（因此 Nuetron Server 的policy 设置了"create\_network:shared": "rule:admin\_only"。），因此这种网络是完全隔离的，也不可以被别的 tenant 共享。

    Tenant network 也有 local，flat，vlan，gre 和 vxlan 等类型。但是，tenant 普通用户创建的 Flat 和 VLAN tenant network 实际上还是 Provider network，所以真正有意义的是 GRE 和 VXLAN 类型，这种网络和物理网络没有绑定关系。

    创建 tenant network 的过程：

（0）管理员在 neutron 配置文件中配置 tenant\_network\_types，其值可以设为一个所支持的网络类型列表，比如 “vlan,gre,vxlan”。其默认值为 “local“，因此需要改变。该值表明该 OpenStack 云中允许被创建的 tenant network 类型。

（1）运行命令 neutron net-create <net\_name>

（2）neutron server 逐一根据该配置项尝试创建 network segment，成功则立即返回。

[复制代码](javascript:void(0);)

def allocate\_tenant\_segment(self, session):

for network\_type in self.tenant\_network\_types: #挨个试

driver = self.drivers.get(network\_type)

segment = driver.obj.allocate\_tenant\_segment(session)

if segment: #返回第一个成功的 segment

return segment

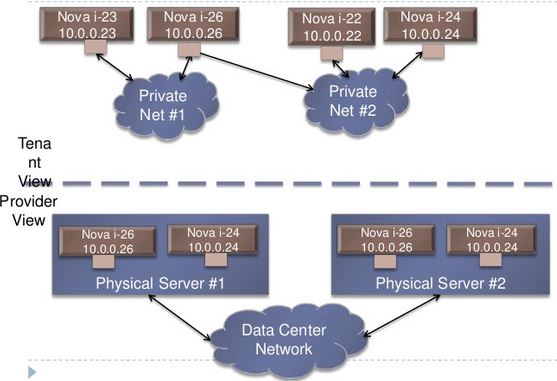
raise exc.NoNetworkAvailable()

[复制代码](javascript:void(0);)

    创建每种网络时，使用不同的配置项：

|  |  |  |  |
| --- | --- | --- | --- |
| 网络类型 | 配置项 | 说明 | 实例 |
| vlan | * network\_vlan\_ranges = physnet1:1000:2999,physnet2 | * 指定所使用的物理网络的标签和支持的 VLAN ID 范围 | network\_vlan\_ranges = default:2000:3999 integration\_bridge = br-int bridge\_mappings = default:br-eth1 |
| flat | * flat\_networks = physnet1,physnet2 | * 指定所使用的物理网络的 label |  |
| gre | * tunnel\_id\_ranges =a list of <tun\_min>:<tun\_max> * local\_ip = <ip> | * 一组可用的 GRE ID 区间列表； * 建立 GRE Tunnel 使用的 IP 地址 | enable\_tunneling = true tunnel\_id\_ranges = 1:1000 integration\_bridge = br-int tunnel\_bridge = br-tun local\_ip = 10.0.0.3 |
| vxlan | * vni\_ranges = a list of *<vni\_min>:<vni\_max>* * local\_ip = <ip> * vxlan\_group = 239.1.1.1 | * 一组可用的 VNI 区间列表； * 建立 VxLAN Tunnel 使用的 IP 地址 * VXLAN 组播组 | enable\_tunneling = true tunnel\_type = vxlan integration\_bridge = br-int tunnel\_bridge = br-tun local\_ip = 10.0.0.3 tunnel\_types = vxlan |
| 所有 | * integration\_bridge * bridge\_mappings = default:br-eth1 * tunnel\_bridge * enable\_tunneling = False | * 指定intergration bridge 的名称，默认为 br-int； * 指定物理网络label 和服务器上的 bridge 对应关系； * 指定 turnnel bridge 的名称，默认为 br-tun * 是否使用 tunneling 类型的网络，包括 GRE 和 VxLAN。使用的话，ML2 agent 会在每个计算和网络节点创建名称为 tunnel\_bridge 的 tunnel bridge。 |  |

**2.1.4 Provider network 和 Tenant network 的区别**



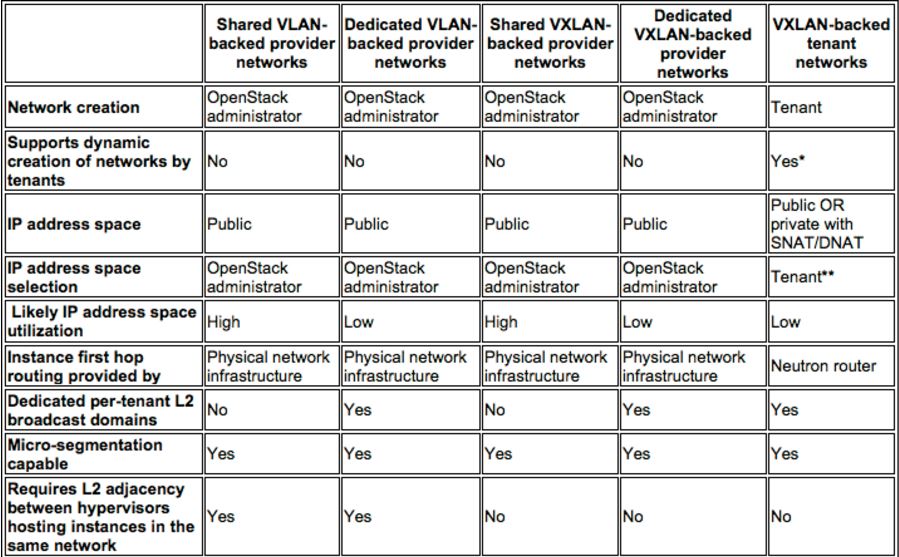
（图6.来源：google）

几个区别：

* Provider network 是由 Admin 用户创建的，而 Tenant network 是由 tenant 普通用户创建的。
* Provider network 和物理网络的某段直接映射，比如对应某个 VLAN，因此需要预先在物理网络中做相应的配置。而 tenant network 是虚拟化的网络，Neutron 需要负责其路由等三层功能。
* 对 Flat 和 VLAN 类型的网络来说，只有 Provider network 才有意义。即使是这种类型的 tenant network，其本质上也是对应于一个实际的物理段。
* 对 GRE 和 VXLAN 类型的网络来说，只有 tenant network 才有意义，因为它本身不依赖于具体的物理网络，只是需要物理网络提供 IP 和 组播即可。
* Provider network 根据 admin 用户输入的物理网络参数创建；而 tenant work 由 tenant 普通用户创建，Neutron 根据其网络配置来选择具体的配置，包括网络类型，物理网络和 segmentation\_id。
* 创建 Provider network 时允许使用不在配置项范围内的 segmentation\_id。

这个文档 <http://superuser.openstack.org/articles/tenant-networks-vs-provider-networks-in-the-private-cloud-context/> 对这两个概念有很全面的阐述。

这个文档 <http://superuser.openstack.org/articles/provisioning-neutron-networks-with-the-nsx-v-plugin/> 也有一些说明。



**2.1.5 多网段 Provider network （multi-segment provider network）**

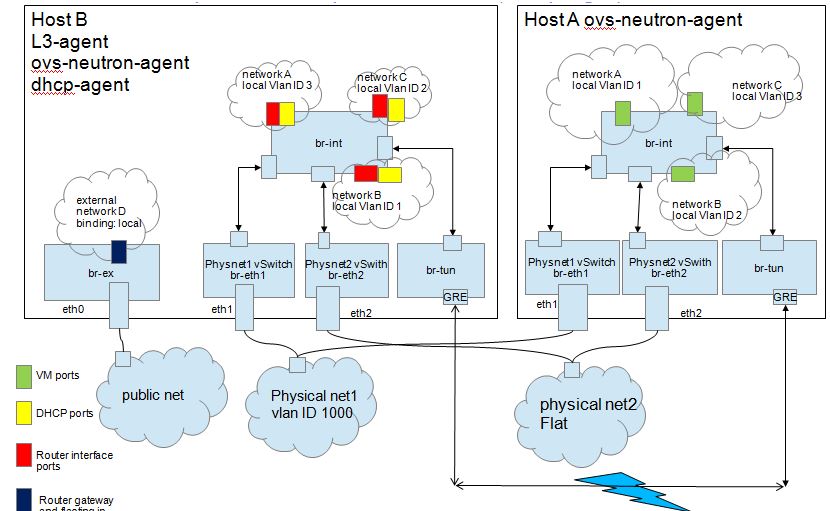
    默认情况下，用户只能创建单网段 provider network。这个 [Blueprint](https://blueprints.launchpad.net/neutron/+spec/ml2-multi-segment-api) 使得 Neutron 能够支持创建多网段网络。这种网络由相同或者不同网络类型的多个网段（network segments）组成一个广播域。一个 Port 只能在一个 segment 中分配。

    一个实例：

s1@controller:~$ neutron net-create multinet --segments type=dict list=true provider:physical\_network=physnet1,provider:segmentation\_id=153,provider:network\_type=vlan provider:physical\_network='',provider:segmentation\_id=801,provider:network\_type=vxlan

Created a new network:  
+-----------------+-------------------------------------------------------------------------------------------------------------+  
| Field           | Value                                                                                                       |  
+-----------------+-------------------------------------------------------------------------------------------------------------+  
| admin\_state\_up  | True                                                                                                        |  
| id                     | 206ae34c-7993-4128-b14d-196e084fbefe                                                                        |  
| name                | multinet                                                                                                    |  
| router:external  | False                                                                                                       |  
| segments          | {"provider:network\_type": "vlan", "provider:physical\_network": "physnet1", "provider:segmentation\_id": 153} |  
|                         | {"provider:network\_type": "vxlan", "provider:physical\_network": null, "provider:segmentation\_id": 801}      |  
| shared              | False                                                                                                       |  
| status               | ACTIVE                                                                                                      |  
| subnets             |                                                                                                             |  
| tenant\_id          | 74c8ada23a3449f888d9e19b76d13aab                                                                            |  
+-----------------+-------------------------------------------------------------------------------------------------------------+

下图中的网络包括类型分别为 flat，gre 和 vlan 的三个网段：



（图7。来源 google）

**2.1.6 Mirantis 描述的网络**

<https://docs.mirantis.com/openstack/fuel/fuel-9.0/pdf/Mirantis-OpenStack-9.0-PlanningGuide.pdf>

Mirantis 将逻辑网络分为以下三大类：

* public network （公共网络）
  + 虚机通过 pulic network 访问 internet。它分配外部的 IP 地址给网络节点，以及在使用 DVR 时分配给计算节点，然后虚机通过 SNAT 来访问外网。
  + 它也向 public endpoints 提供虚拟IP，用于外网访问 openstack service api。
  + 管理员制定一个相邻的地址区间给浮动IP使用。
  + 需要将 public network 与其它网络隔离。
* internal network （内部网络）
  + internal network 是出了 public network 和 provate network 的其他网络的统称，包括存储，管理，PXE 网络等。
    - 内部网络 - PXE 网络，用于环境部署
    - 存储网络 - 用于存储访问
    - 管理网络 - 包括数据库，MQ，HA 服务，以及计算和存储节点之间的 iSCSI 网络。
  + 内部网络将各节点连接起来。openstack 环境中的各个模块之间的交互通过内部网络进行。
  + 不要将 internal network 和 private network 混淆，private network 只用于虚机之间的通信。
  + 出于安全考虑，需要将内部网络从 public 和 private network 隔离出来。
* Private network （私有网络）
  + 用于虚机之间的通信，包括 VLAN, GRE/VXLAN 等类型的虚拟网络都会运行在私有网络之上。

当 Neutron使用 VLAN 模式时候的网卡分配方案：

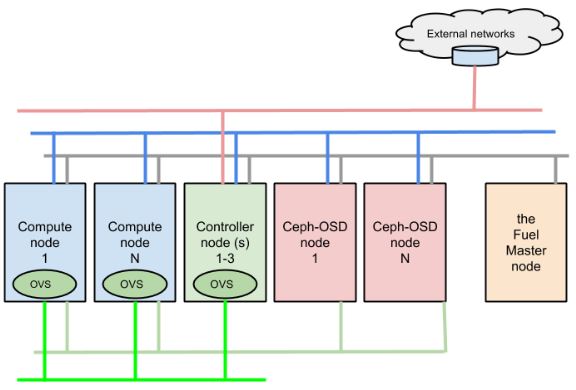
* 三网卡方案：

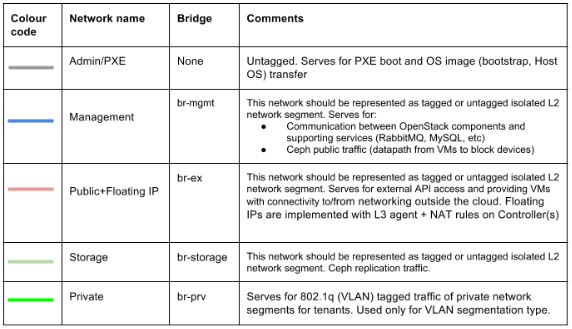
eth0 - PXE 网络  
eth1 - 公共网络（untagged），管理网络（tag=102），存储网络（tag=103）  
eth2 - 私有网络

* 四网卡方案

eth0 - PXE 网络  
eth1 - 公共网络（untagged），管理网络（tag=102）  
eth2 - 私有网络  
eth3 - 存储网络

这是 OVS + VLAN 使用三块网卡时的配置示例：





当 Neutron使用 隧道（GRE或者 VXLAN） 模式时候的网卡分配方案：

* 两网卡方案

eth0 - PXE 网络  
eth1 - 公共网络（untagged），管理网络（tag=102），存储网络（tag=103）

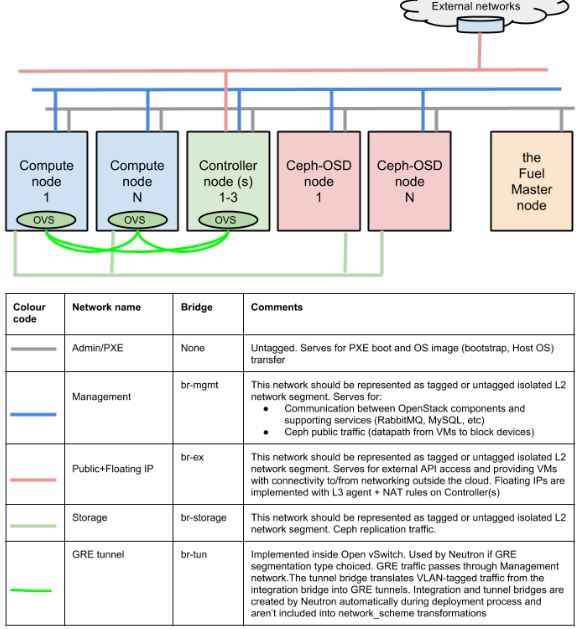
* 三网卡方案：

eth0 - PXE 网络  
eth1 - 公共网络（untagged），管理网络（tag=102）  
eth2 - 存储网络

* 四网卡方案

eth0 - PXE 网络  
eth1 - 管理网络  
eth2 - 公共网络  
eth3 - 存储网络

 这是 OVS + GRE 配置示例：



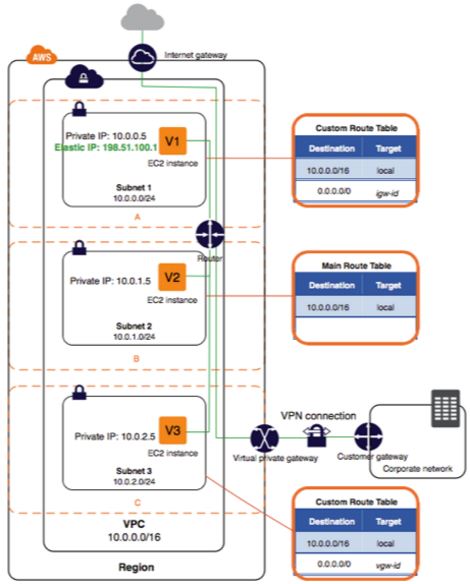
备注：这表格中提到 GRE网络和管理网络混在一起，这是 mirantis Fuel 6.0从UI上配置的一个已知问题，详情在这里 <https://bugs.launchpad.net/fuel/+bug/1285059>。其实Mrantis也是不推荐这么配置的。

**2.2 子网 （subnet）**

    子网是一组 IPv4 或 IPv6 地址以及与其有关联的配置。它是一个地址池，OpenStack 可从中向虚拟机 (VM) 分配 IP 地址。每个子网指定为一个无类别域间路由 (Classless Inter-Domain Routing) 范围，必须与一个网络相关联。除了子网之外，租户还可以指定一个网关、一个域名系统 (DNS) 名称服务器列表，以及一组主机路由。这个子网上的 VM 实例随后会自动继承该配置。

   在 OpenStack Kilo 版本之前，用户需要自己输入 CIDR。Kilo 版本中实现了这个 [Blueprint](https://blueprints.launchpad.net/neutron/+spec/subnet-allocation)，使得 Neutron 能够从用户指定的 CIDR Pool 中自动分配 CIDR。

   注：在AWS中，该概念对应其 Subnet 概念。AWS 对 Subnet 的数目有一定的限制，默认地每个 VPC 内最多只能创建 200 个 Subnet。从CIDR角度看，一个 VPC 有一个比较大的网段，其中的每个 Subnet 分别拥有一个较小的网段。这种做法和OpenStack 有区别，OpenStack 中创建网络时不需要指定 CIDR。



**2.3 端口 （Port）**

    一个 Port 代表虚拟网络交换机（logical network switch）上的一个虚机交换端口（virtual switch port）。虚机的网卡（VIF - Virtual Interface）会被连接到 port 上。当虚机的 VIF 连接到 Port 后，这个 vNIC 就会拥有 MAC 地址和 IP 地址。Port 的 IP 地址是从 subnet 中分配的。

**2.4 虚机交换机 （Virtual switch）**

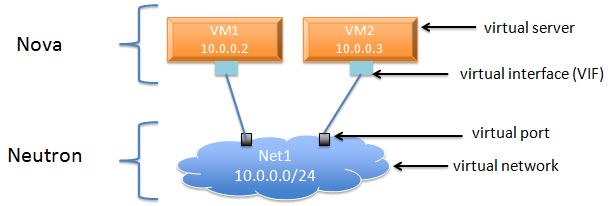
  Neutron 默认采用开源的 Open vSwitch 作为其虚机交换机，同时还支持使用 Linux bridge。

**2.5 虚拟路由器 （Virtual router）**

  一个 Virtual router 提供不同网段之间的 IP 包路由功能，由 Nuetron L3 agent 负责管理。

**2.6 各组件之间的关系**

  OpenStack 实际上并未增加网络功能。路由、交换和名称解析是由底层的网络基础架构处理的。OpenStack 的作用是将这些组件的管理捆绑在一起，并将它们连接到计算工作负载。

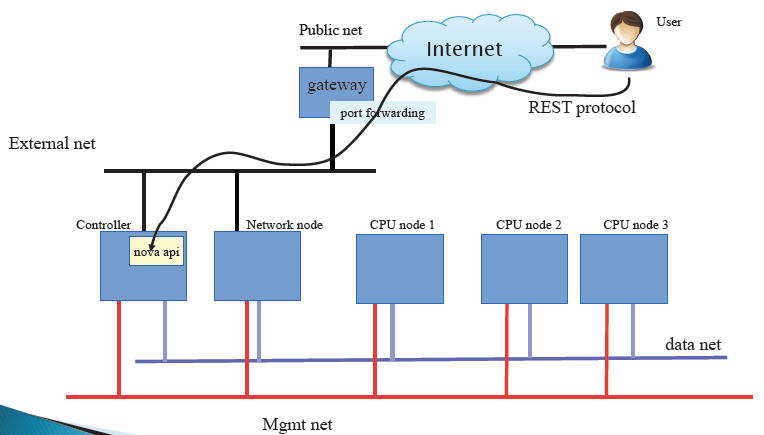


（图8.来源 google）

**3. Neutron中的网络连通性**

  如上面第二章节所述，一个标准 OpenStack 环境中的物理网络配置往往包括：

* Internet（Pulic network）：传统意义上的公共网络，使用往往由电信运营商提供的公共IP。
* 外部网络（External network）：数据中心 Intranet，从这里分配浮动IP地址。
* OpenStack 内部网络：
  + 管理网络（management network）：提供 OpenStack 各个组件之间的内部通信，以及 API 访问端点（Endpoint）。为安全考虑，该网络必须限制在数据中心之内。
  + API 网络：其实这不是一个单独的网络，而是包含在外部和内部网络中。API 的 Endpoint 包括 publicurl 和 internalurl，其中，publicurl  包含的是 externa network 的 IP 地址，internal network 包含的是 management network IP 地址。为了简单起见，提供给内外网络访问的API的 publicurl 和 internalurl 相同，而只给内部网络访问的 API 只使用 internalurl。
  + 数据网络（data network）：除管理网络以外的其它网络，往往还可以细分为下面几种。它们可以合为一种，也可以从性能方面考虑分离出一种或几种作为单独的网络。
    - 租户网络（Tenant network）：提供虚机在计算节点之间，以及计算节点和网络节点之间的通信。同样这也是数据中心的内部网络。
    - 存储访问网络（storage access network）：访问存储的网络。
    - 存储后端网络（storage backend network）：比如 Ceph 和 Swift 集群用于后端数据复制的网络。
* 除了以上网络外，往往还有各种功能网络，包括 IPMI 网络，PXE 网络，监控网络等等。这部分就略去不谈了。



这几种网络，在物理交换机上，往往都使用 VLAN 来做网络隔离。现在讨论的只是租户网络即虚机之间通信的网络，在 Neutron 的实现看来，该网络的连通性包括几个层次：

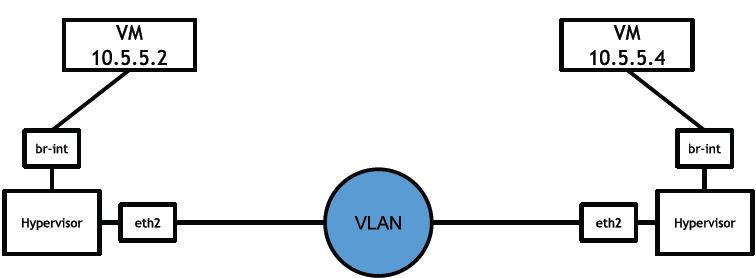
* 同主机和不同主机上一个网段内的虚机之间的连接性：虚拟二层网络，走物理二层（VLAN）或者三层（GRE/VxLAN）网络。
* 不同网段内的虚机之间的连通性：经过物理（VLAN）或者 Neutron Virtual router
* 虚机和外部网络之间的连通性：经过物理路由器（给 VLAN 虚拟网络实用的物理交换机连接的路由器）或者 Neutron Virtual router

为了支持这些网络连通性，Neutron 需要实现跨主机的二层网络和虚拟路由器。可以参考 [学习OpenStack之（5）：在Mac上部署Juno版本OpenStack 四节点环境](http://www.cnblogs.com/sammyliu/default.html?page=6)。

**3.1 虚拟二层网络的实现**

    所谓虚拟二层网络，就是提供给虚机的一种虚拟网络，使得虚机可以和同网段中的其它虚机就像在物理二层网络一样在网络二层直接通信，不管目的虚机处于什么物理位置。也就是说，对虚机来说，物理三层网络对它是透明的。

**3.1.1 使用 VLAN 实现虚拟二层网络**



（图10.来源：google）

这种实现方式基于物理网络上的使用 VLAN 的交换机。请参考 （2）[Neutron OpenvSwitch + VLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4626419.html%20)。

**3.1.2 基于 GRE/VxLAN 实现的二层网络 （L2）**

    除了 VLAN 方式的物理的二层网络，另一种方式使用Tunnel/Overlay 方式实现虚机二层网络。那Overlay如何应对云计算网络的挑战呢？

（1）首先，Overlay的核心是重新构建一个逻辑网络平面，其技术手段的关键是采用隧道技术实现L2oIP的封装。通过隧道实现各虚拟机之间的二层直连。这样网络只看见Overlay边缘节点的MAC地址，物理网络学习到的MAC表项非常有限，现有接入交换机32K的MAC表项足以满足需求（如下图所示）。对应的Overlay边缘节点实现基于会话的地址学习机制，也就是说只学习有交互流量的虚拟机MAC地址。这样也严格限制了边缘节点的地址表项。



Overlay网络如何应对云计算网络的挑战

（2）其次，Overlay网络仅仅是一个逻辑上的二层直连网络。其依赖的物理网络，是一个传统的路由网络。这个路由网络是一个三层到边缘的网络。也就是说二层广播域被缩小到极致。这样，网络风暴潜在的风险大幅度降低。同时，对于一些需要依赖二层广播的协议报文，例如：ARP报文，Overlay网络通过ARP代理等方式实现协议的内容透传，不会影响协议报文的正常运作。

（3）再次，针对4K VLAN上限问题，Overlay网络通过L2oIP的封装字段，提供24bits长度的隔离ID，最大可以支持16M租户或业务。

（4）最后，针对网络虚拟化问题。Overlay网络本身就是一个从物理网络中抽离的逻辑网络，通过名址分离使得内层IP地址完全作为一个寻址标签，不再具备路由功能，可以实现不同subnet之间二层互通，保证二层网络的连通性不受IP地址段的限制。

（[资料来源](http://www.h3c.com.cn/About_H3C/Company_Publication/IP_Lh/2014/07/Home/Catalog/201501/852548_30008_0.htm)）

  在 Neutron 中，Neutron 将虚机发出的数据帧打包，走三层物理网络到达目的虚机的主机，解包给虚机。这种实现使得两个虚机的通信看起来是二层网络通信。

（图11.来源：google）

请参考：（3）[Neutron OpenvSwitch + GRE/VxLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4627230.html%20)

**3.2 虚拟路由器 （virtual router）**

  跨子网的通信需要走虚拟路由器。同物理路由器一样，虚拟路由器由租户创建，拥有多个 virtual interface，连接一个租户的子网，以及外部网络。它具有以下特性：

* 一个 VR 只属于创建它的租户，只用于该租户的子网之间和子网与外网的路由
* 同一网络内的若干子网可以挂在一个 VR 上
* 同一租户的不同网络的没有 IP 地址重叠的子网可以挂在一个 VR 上
* 不同租户之间的内网之间是不能使用 VR 的
* 同一租户的不同网络内的有 IP 地址重叠的两个子网不能使用同一个 VR（添加子网到 VR 时会报错）
* 在网络节点上，一个 VR 运行在一个 Network namespace 内，该namespace 的名称包含该 VR 的 UUID

具体会在后面的文章中分析。

（图12。来源：google）

请参考：（6）[Neutron L3 Agent](http://www.cnblogs.com/sammyliu/p/4636091.html%20)

**3.3 DHCP 服务**

    DHCP 服务是网络环境中必须有的。Neutron 提供基于 Dnamasq 实现的虚机 DHCP 服务，向租户网络内的虚机动态分配固定 IP 地址。它具有以下特性：

* 一个网络可以有多个运行在不同物理网络节点上的 DHCP Agent，同时向网络内的虚机提供服务
* 一个 DHCP Agent 只属于一个网络，在网络节点上运行在一个 network namespace 内
* 网络内的子网共享该 DHCP Agent

具体请参考 （5）[Neutron DHCP Agent](http://www.cnblogs.com/sammyliu/p/4419195.html)。

**4. Neutron 租户网络的隔离性（isolation of tenant network）**

    Neutron 实现了不同层次的租户网络隔离性：

* 租户之间的网络是三层隔离的，连通过 VR 做路由都不行，实在要连通的话，需要走物理网络
* 一个租户内的不同网络之间二层隔离的，需要通过 VR 做三层连通
* 一个网络内的不同子网也是二层隔离的，需要通过 VR 做三层连通

    Neutron 对每个租户网络（tenant network）都分配一个 segmentation\_id，其特点包括：

* 每个 tenant network 都有一个这种 ID
* 每个租户网络的 ID 在全部的租户范围内都是唯一的
* 一个 ID 代表一个广播域
* 一个 ID 使得同一网络内的两个虚机之间好像建立了一个虚拟通道（tunnel）一样
* 不同 ID 的网络 tunnel 之间是互相隔离的
* 根据物理实现不同，该ID被实现为几种不同的形式：
  + VLAN ID
  + GRE Tunnel ID
  + VxLAN VNI

以下图描述的 GRE 类型的网络为例：

（图13。[图片来源](http://pinrojas.com/2014/07/29/theres-real-magic-behind-openstack-neutron/)。）

* （1）计算节点的 br-int 上，neutron 为每个虚机连接 OVS 的 access port 分配了内部的 VLAN Tag。这种 tag 限制了网络流量只能在 tenant network 之内。
* （2）计算节点的 br-tun 上，neutron 将内部的 VLAN Tag 转化为 GRE Tunnel ID，是的不同 network 的流量走不通的 tunnel。
* （3）网络节点的 br-tun 上，neutron 将 GRE Tunnel ID 转发了一一对应的 内部 VLAN Tag，使得网络流被不同的服务处理。
* （4）网络节点的 br-int 上连接的 DHCP 和 L3 agent 使用 Linux network namespace 进行隔离。

请参考：

（2）[Neutron OpenvSwitch + VLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4626419.html%20)

（3）[Neutron OpenvSwitch + GRE/VxLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4627230.html%20)

**5. Neutron 租户网络的安全性（security）**

除了租户的隔离性以外，

* Neutron 还提供数据网络与外部网络的隔离性。默认情况下，所有虚机通往外网的流量全部走网络节点上的 L3 agent。在这里，内部的固定 IP 被转化为外部的浮动 IP 地址。这种做法一方面保证了网络包能够回来，另一方面也隐藏了内部的 IP 地址。
* Neutron 还是用 Linux iptables 特性，实现其 Security Group 特性，从而保证访问虚机的安全性。
* Neutron利用网络控制节点上的 network namespace 中的 iptables，实现了进出租户网络的网络包防火墙，从而保证了进出租户网络的安全性。

请参考下面的文章：

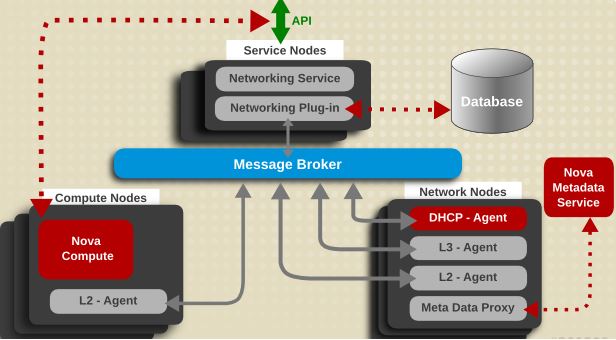
（8）[Neutron Security Group](http://www.cnblogs.com/sammyliu/p/4658746.html%20)

（9）[Neutron FWaas 和 Nova Security Group](http://www.cnblogs.com/sammyliu/p/4675991.html)

**6. Neutron 租户网络的可用性（HA）和扩展性（Scalability）**

OpenStack 云中可能用于成千上万台虚机，成千上万个租户，因此，Neutron 的数据网络的可用性和扩展性非常重要。Neutron 中，这些特性包括几个层次：

（1）软件架构上，Neutron 实现OpenStack 标准的去中心化架构和插件机制，有效地保证了其扩展性。



（图14.来源于 google）

（2）Neutron 为每个 provider/tenant network 分配一个唯一的 segmentation\_id。该 ID 的数目是影响可扩展性的因素之一。在规模不大的私有云中，往往是用 VLAN 模式，简单、可靠、能够满足规模要求；而在大型的私有云或者公有云中，往往使用 VxLAN。

|  |  |  |
| --- | --- | --- |
| 网络类型 | ID 数目 | 说明 |
| flat | 1 | 通常不用于数据网络，因为其不具备租户隔离型。 |
| vlan | 4094 | 802.1Q 定义了 VLAN ID 占 12-bit |
| GRE | 16777216 | RFC 2637 定义的 GRE 的 ID 占 24-bit |
| VxLAN | 16777216 | RFC 7348 定义的 VxLAN 的 ID 占 24-bit |

（3）分布式 Virtual Router （DVR） 和 分布式 DHCP agent

默认情况下，L3 agent 和 DHCP agent 部署在网络节点上，这在大规模的云环境中可能会存在性能问题。这是 [Blueprint](https://wiki.openstack.org/wiki/Neutron/DVR)。

 |

（左图-图15，未使用DVR。右图-图16，使用了DVR-分布式虚拟路由器。[来源](https://wiki.openstack.org/wiki/Neutron/DVR)。）

通过使用 DVR，L3 转发和 NAT 会被分布在计算节点上，这使得计算节点变成了网络节点，这样集中式的网络节点的负载就被分担了。这个 [Blueprint](https://blueprints.launchpad.net/neutron/+spec/distributed-dhcp) 实现了分布式的 DHCP Agent，这进一步释放了集中式网络节点的压力。

（4）L2 Population 和 ARP Responder：这两个功能大大减少了网络的复杂性，提交了网络效率，从而促进了扩展性。

（4）高可用：虽然 Neutron 在高可用实现方面走在了 OpenStack 所有组件的前列，目前已经实现了多种 L3 Agent 的 HA 方案，但是，这些方案目前还不是很成熟，大都是在 Juno 版本中添加的，而且现在还在持续优化中，离正式进入生产环境还有一些距离。但是，我们可以相信，再经过两三个版本，这些功能就能够进入生产环境。

请参考这个文章：

（4）[Neutron OVS OpenFlow 流表 和 L2 Population](http://www.cnblogs.com/sammyliu/p/4633814.html%20)

（11）[Neutron DVR](http://www.cnblogs.com/sammyliu/p/4713562.html)

（12）[Neutron VRRP](http://www.cnblogs.com/sammyliu/p/4692081.html)

（13）[High Availability （HA）](http://www.cnblogs.com/sammyliu/p/4741967.html)

**7. Neutron 扩展服务（extension service）**

    在实际的网络中，除了网络的核心功能以外，还有一些普遍应用的网络服务，比如 VPN， Load Balancing 和 Firewall 等。Neutron 针对这些普遍的服务，也提供了参考实现。需要注意的是，目前这些实现更多类似于一种原型实现（PoC），其目的是向给产品实现者提供一种参考架构，因此，这些功能往往还不能直接应用到生产系统。正是考虑到这些特点，目前 Neutron 项目组将这些代码挪出了 Neutron的主代码库。

具体请参考：

（7）[Neutron LBaas](http://www.cnblogs.com/sammyliu/p/4656176.html%20)

（9）[Neutron FWaas 和 Nova Security Group](http://www.cnblogs.com/sammyliu/p/4675991.html)

（10）[Neutron VPNaas](http://www.cnblogs.com/sammyliu/p/4677386.html%20)

**8. Neutron REST API**

    Neutron 的 neutron-server 模块提供 REST API 来操作上文提到的各种网络组件，包括 network，subnet，port，router 等。可参考上图13。具体的 API 可以参考 [这里](http://developer.openstack.org/api-ref-networking-v2.html)和[这里](http://developer.openstack.org/api-ref-networking-v2-ext.html)。

持续更新：

2015/11/12 （2， 3章节）