[**Neutron 理解 (2): 使用 Open vSwitch + VLAN 组网 [Netruon Open vSwitch + VLAN Virutal Network]**](http://www.cnblogs.com/sammyliu/p/4626419.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. L2 基础知识**

**1.1 VLAN 基础知识**

**1.1.1 VLAN 的含义**

   LAN 表示 Local Area Network，本地局域网，通常使用 Hub 和 Switch 来连接LAN 中的计算机。一般来说，当你将两台计算机连入同一个 Hub 或者 Switch 时，它们就在同一个 LAN 中。同样地，你连接两个 Switch 的话，它们也在一个 LAN 中。一个 LAN 表示一个广播域，它的意思是，LAN 中的所有成员都会收到 LAN 中一个成员发出的广播包。可见，LAN 的边界在路由器或者类似的3层设备。

   VLAN 表示 Virutal LAN。一个带有 VLAN 功能的switch 能够同时处于多个 LAN 中。最简单地说，VLAN 是一种将一个交换机分成多个交换机的一种方法。比方说，你有两组机器，group A 和 B，你想配置成组 A 中的机器可以相互访问，B 中的机器也可以相互访问，但是A组中的机器不能访问B组中的机器。你可以使用两个交换机，两个组分别接到一个交换机。如果你只有一个交换机，你可以使用 VLAN 达到同样的效果。你在交换机上分配配置连接组A和B的机器的端口为 VLAN access ports。这个交换机就会只在同一个 VLAN 的端口之间转发包。

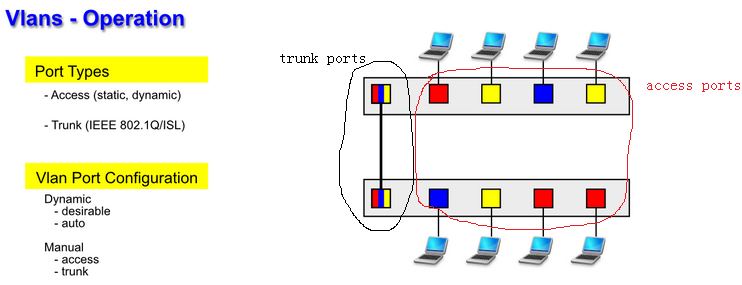
（图1）

    IEEE 802.1Q 标准定义了 VLAN Header 的格式。它在普通以太网帧结构的 SA （src addr）之后加入了 4bytes 的 VLAN Tag/Header 数据，其中包括 12-bits 的 VLAN ID。VLAN ID 最大值为4096，但是有效值范围是 1 - 4094。

（图2）

    带 VLAN 的交换机的端口分为两类：

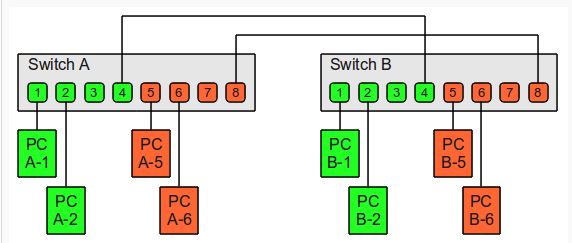
* Access port：这些端口被打上了 VLAN Tag。离开交换机的 Access port 进入计算机的以太帧中没有 VLAN Tag，这意味着连接到 access ports 的机器不会觉察到 VLAN 的存在。离开计算机进入这些端口的数据帧被打上了 VLAN Tag。
* Trunk port： 有多个交换机时，组A中的部分机器连接到 switch 1，另一部分机器连接到 switch 2。要使得这些机器能够相互访问，你需要连接两台交换机。 要避免使用一根电缆连接每个 VLAN 的两个端口，我们可以在每个交换机上配置一个 VLAN trunk port。Trunk port 发出和收到的数据包都带有 VLAN header，该 header 表明了该数据包属于那个 VLAN。因此，只需要分别连接两个交换机的一个 trunk port 就可以转发所有的数据包了。通常来讲，只使用 trunk port 连接两个交换机，而不是用来连接机器和交换机，因为机器不想看到它们收到的数据包带有 VLAN Header。

   （图3）

**1.1.2 VLAN 的类型**

（1）基于端口的 VLAN （untagged VLAN - 端口属于一个VLAN，数据帧中没有VLAN tag）

    这种模式中，在交换机上创建若干个VLAN，在将若干端口放在每个VLAN 中。每个端口在某一时刻只能属于一个VLAN。一个 VLAN 可以包含所有端口，或者部分端口。每个端口有个PVID （port VLAN identifier)。这种模式下，一个端口上收到的 frame 是 untagged frame，因此它不包含任何有关 VLAN 的信息。VLAN 的关系只能从端口的 PVID 上看出来。交换机在转发 frame 时，只将它转发到相同 PVID 的端口。

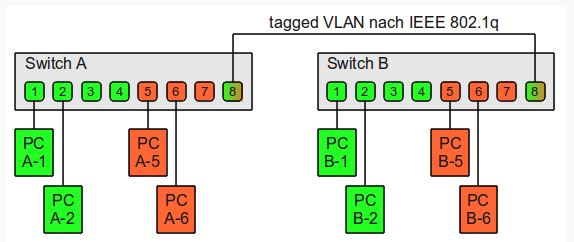


如上图所示，连接两个交换机的同一个 VLAN 中的两个计算机需要通信的话，需要在两个交换机之间连两根线：

* 一根从 Switch A 端口4 到 Switch B 端口 4 （VLAN 1）
* 一根从 Switch A 端口8 到 Switch B 端口 8 （VLAN 2）

（2）Tagged VLANs （数据帧中带有 VLAN tag）

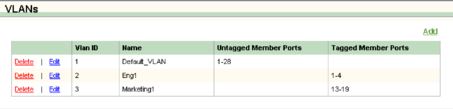
这种模式下，frame 的VLAN 关系是它自己携带的信息中保存的，这种信息叫 a tag or tagged header。当交换机收到一个带 VLAN tag 的帧，它只将它转发给具有同样 VID 的端口。一个能够接收或者转发 tagged frame 的端口被称为 a tagged port。所有连接到这种端口的网络设备必须是 802.1Q 协议兼容的。这种设备必须能处理 tagged frame，以及添加 tag 到其转发的 frame。



上图中，两个交换机上的端口8 支持 VLAN 1 和 2， 因此一根线就可以了实现跨交换机的同VLAN 内的计算机互相通信了。

在实际的VLAN 配置中的各种情况：

* 交换机的所有端口，部分是 tagged port，部分被添加到 VLAN 中。
* 一个 untagged port，不管它是一个基于端口的VLAN 的一个成员还是一个 tagged VLAN 中的一个成员，一个时刻只能在一个 VLAN 中。
* 一个 tagged port，可以是多个 VLAN 的成员。
* 一个 port，可以同时是一个 VLAN 的 untagged member，以及不同 VLAN 中的 tagged member。

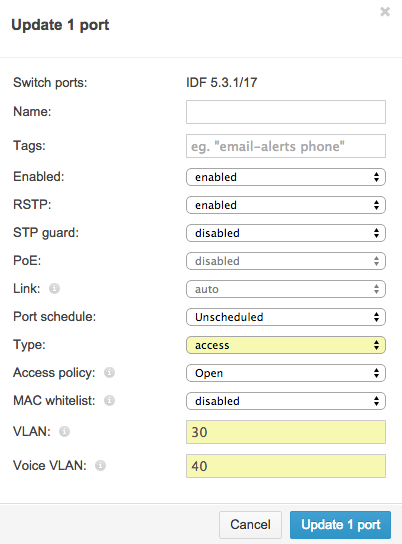


**1.1.3 交换机端口类型**

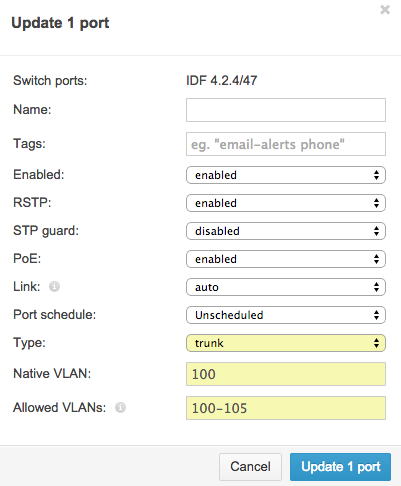
    以太网端口有三种链路类型：Access、Hybrid和Trunk。

* Access类型的端口只能属于1个VLAN，一般用于连接计算机的端口

这种类型的端口允许接收没有打标签的帧，再发出去时将会被打上标签。

（[图片来源](https://documentation.meraki.com/MS/Port_and_VLAN_Configuration/Switch_Ports)）

* Trunk类型的端口可以属于多个VLAN，可以接收和发送多个VLAN的报文，一般用于交换机之间连接的端口

（[图片来源](https://documentation.meraki.com/MS/Port_and_VLAN_Configuration/Switch_Ports)）

在配置 trunk 口时，可以指定允许接收的VLAN 的 ID 区间，还可以配置一个 Native VLAN （缺省VID，也称为 PVID）。当设置了 PVID 时，没有打标签的进来的帧将被打上PVID 的 tag再被发出去。

* Hybrid类型的端口可以属于多个VLAN，可以接收和发送多个VLAN的报文，可以用于交换机之间连接，也可以用于连接用户的计算机。Hybrid端口和Trunk端口的不同之处在于Hybrid端口可以允许多个VLAN的报文发送时不打标签，而Trunk端口只允许缺省VLAN的报文发送时不打标签。

各种类型：

* Access (接收) Tagged = PVID 不接收 注：部分高端产品可能接收。
* Access (接收) Tagged =/ PVID 不接收 注：部分高端产品可能接收。
* Access (接收) Untagged 接收 增加tag＝PVID 从PC
* Access (发送) Tagged = PVID 转发 删除tag
* Access (发送) Tagged =/ PVID 不转发 不处理
* Access (发送) Untagged 无此情况
* Trunk (接收) Tagged = PVID 接收 不修改tag
* Trunk (接收) Tagged =/ PVID 接收 不修改tag
* Trunk (接收) Untagged 接收 增加tag＝PVID
* Trunk (发送) Tagged = PVID If Passing then 转发 删除tag
* Trunk (发送) Tagged =/ PVID If Passing then 转发 不修改tag
* Trunk (发送) Untagged 无此情况
* Hybrid (接收) Tagged = PVID 接收 不修改tag 对端是trunk
* Hybrid (接收) Tagged =/ PVID 接收 不修改tag 对端是trunk
* Hybrid (接收) Untagged 接收 增加tag＝PVID 类Trunk
* Hybrid (发送) Tagged = PVID Tag 和 untag 中列出的vlan可以passing 看Tag项和untag项
* Hybrid (发送) Tagged =/ PVID Tag 和 untag 中列出的vlan可以passing 看Tag项和untag项
* Hybrid (发送) Untagged 无此情况

解释：

* 主机只能处理标准以太帧（没打标签的），交换机内部的帧都是打了标签的。
* 收报文：Acess端口1 收到一个报文,判断是否有VLAN信息：如果没有则打上端口的PVID，并进行交换转发,如果有则直接丢弃（缺省）
* 发报文：Acess端口1 将报文的VLAN信息剥离，直接发送出去
* 收报文：trunk端口： 1、收到一个报文，判断是否有VLAN信息：如果没有则打上端口的PVID，并进行交换转发，如果有判断该trunk端口是否允许该 VLAN的数据进入：如果可以则转发，否则丢弃
* 发报文：trunk端口： 1、比较端口的PVID和将要发送报文的VLAN信息，如果两者相等则剥离VLAN信息，再发送，如果不相等则直接发送
* 收报文： hybrid端口： 1、收到一个报文 2、判断是否有VLAN信息：如果没有则打上端口的PVID，并进行交换转发，如果有则判断该hybrid端口是否允许该VLAN的数据进入：如果可以则转发，否则丢弃
* 发报文：hybrid端口：1、判断该VLAN在本端口的属性（disp interface 即可看到该端口对哪些VLAN是untag， 哪些VLAN是tag）2、如果是untag则剥离VLAN信息，再发送，如果是tag则直接发送

|  |  |  |  |
| --- | --- | --- | --- |
|  | tagged （进） | untagged（进） | 出 （交换机在做交换时，只会把帧发给包含其 VID 的端口） |
| Access 端口 | 丢弃 | 打上 PVID | 剥离 VID，此时的帧为标准以太网帧 |
| Trunk 端口 | 如果是允许的，则不变；否则丢弃 | 打上 PVID | 如果 VID 与 PVID 不同，则透传；如果 VID 与 PVID 相同，则剥离 VID |

**1.1.4 VLAN 的不足**

1. VLAN 使用 12-bit 的 VLAN ID，所以 VLAN 的第一个不足之处就是它最多只支持 4096 个 VLAN 网络（当然这还要除去几个预留的），对于大型数据中心的来说，这个数量是远远不够的。
2. VLAN 是基于 L2 的，所以很难跨越 L2 的边界，在很大程度上限制了网络的灵活性。
3. VLAN 操作需手工介入较多，这对于管理成千上万台机器的管理员来说是难以接受的。

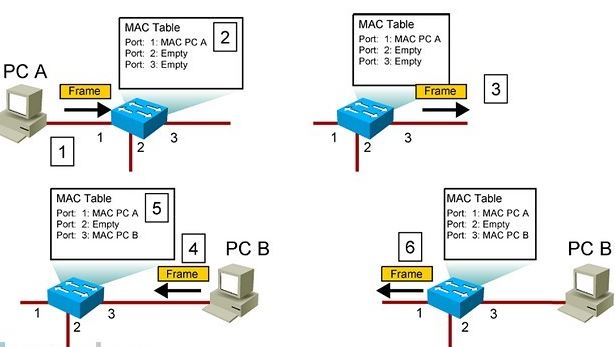
**1.2   二层交换的基础知识**

**1.2.1 二层交换机最基本的功能**

二层交换机最基本的功能包括：

* MAC 地址学习：当交换机从它的某个端口收到数据帧时，它将端口的 ID 和帧的源 MAC 地址保存到它的内部MAC表中。这样，当将来它收到一个要转发到该 MAC 地址的帧时，它就知道直接从该端口转发出去了。
* 数据帧转发：交换机在将从某个端口收到数据帧，再将其从某个端口转发出去之前，它会做一些逻辑判断：
  + 如果帧的目的 MAC 地址是广播或者多播地址的话，将其从交换机的所有端口（除了传入端口）上转发。
  + 如果帧的目的MAC地址在它的内部MAC表中能找到对应的输出端口的话（MAC 地址学习过程中保存的），将其从该端口上转发出去。
  + 对其它情况，将其从交换机的所有端口（除了传入端口）上转发。
* 加 VLAN 标签/去 VLAN 标签：
  + 帧接收：从 trunk port 上收到的数据帧必须是加了标签的。从 access port 上收到的数据帧必须是没有加标签的，否则该帧将会被抛弃。
  + 帧处理：根据上述转发流程决定其发出的端口。
  + 帧发出：从 trunk port 发出的帧是加了标签的。从 access port 上发出的帧必须是没加标签的。

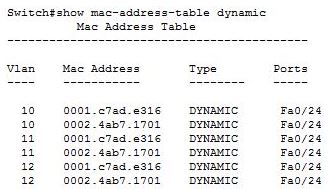
 默认情况下，交换机的所有端口都处于VLAN 1 中，也就相当于没有配置 VLAN。该机制说明如下：

 （图4）

1. PC A 发一个帧到交换机的 1 端口，其目的MAC地址为 PC B 的 MAC。
2. 交换机比较其目的 MAC 地址和它的内部 MAC Table，发现它不存在（此时表为空）。在决定泛洪之前，它把端口 1 和 PC A 的 MAC 地址存进它的 MAC Table。
3. 交换机将帧拷贝多份，分别从2和3端口发出。
4. PC B 收到该帧以后，发现其目的 MAC 地址和他自己的 MAC 地址相同。它发出一个回复帧进入端口3。
5. 交换机将 PC B 的 MAC地址和端口3 存在它的 MAC 表中。
6. 因为该帧的目的地址为PC A 的 MAC 地址它已经在 MAC 表中，交换机直接将它转发到端口1，达到PC A。

配置了 VLAN 的交换机的该机制类似，只不过：

（1）MAC 表格中每一行有不同的 VLAN ID。做比较的时候，拿传入帧的目的 MAC 地址和 VLAN ID 和此表中的行数据相比较。如果都相同，则选择其 Ports 作为转发出口端口。

 （图5）

（2）如果没有吻合的表项，则将此帧从所有有同样 VLAN ID 的 Access ports 和 Trunk ports 转发出去。

**1.2.2 Address Resolution Protocol (ARP) 原理**

二层网络使用 MAC （media access control address）地址作为硬件的唯一标识。基于 TCP/IP 协议的软件使用 ARP 来将 IP 地址转化为 MAC 地址。

1. 目的 IP 地址在同一网段的话

（图6）

    该示例中，Host A 和 B 在同一个网段中。A 的 IP 地址是 10.0.0.99，B 的 IP 地址是 10.0.0.100。当 A 要和 B 通信时，A 需要知道 B 的 MAC 地址。该过程经过以下步骤：

（1）A 上的 IP 协议栈知道通过B 的 IP 地址可以直接到达 B。A 检查它的本地 ARP 缓存来看B 的 MAC 地址是否已经存在。

（2）如果A 没有发现B 的 MAC 地址，它发出一个 ARP 广播请求，来询问“10.0.0.100 的 MAC 地址是什么？”，该数据包：

SRC MAC： A 的 MAC

DST MAC：FF:FF:FF:FF:FF:FF

SRA IP： A 的 IP

DST IP: B 的 IP

（3）该网段中所有的电脑都将收到该包，并且会检查 DST IP 和自己的IP 是否相同。如果不同，则丢弃该包。Host B 发现其IP 地址和 DST IP 相同，它将 A 的 IP/MAP 地址加入到自己的ARP 缓存中。

（4）B 发出一个 ARP 回复消息

SRC MAC： B 的 MAC

DST MAC：A 的 MAC

SRA IP： B 的 IP

DST IP: A 的 IP

（5）交换机直接将该包交给 host A。A 收到后，将 B 的 MAC/IP 地址缓存到 ARP 缓存中。

（6）A 使用 B 的 MAC 作为目的 MAC 地址发出 IP 包。

2. 目的IP 地址不在同一个网段的话

（图7）

本例子中，A 的地址是 10.0.0.99， B 的地址是 192.168.0.99。Router 的 interface 1 和 A 在同一个网段，其IP 地址为10.0.0.1；interface 2 和 B 在同一个网段，其IP地址为 192.168.0.1。

A 使用下面的步骤来获取 Router 的 interface 1 的 MAC 地址。

（1）根据其路由表，A 上的 IP 协议知道需要通过它上面配置的 gateway 10.0.0.1 才能到达到 B。经过上面例子中的步骤，A 会得到 10.0.0.1 的 MAC 地址。

（2）当 A 收到 Router interface 1 的 MAC 地址后，A 发出了给B 的数据包：

SRC MAC： A 的 MAC

DST MAC：Router 的 interface 1 的 MAC 地址

SRA IP： A 的 IP

DST IP: B 的 IP

（3）路由器的 interface1 收到该数据包后，根据其路由表，首先经过同样的ARP 过程，路由器根据 B 的 IP 地址通过 ARP 获得其 MAC 地址，然后将包发给它。

SRC MAC： Router interface 2 的 MAC

DST MAC：B 的 MAC

SRA IP： A 的 IP

DST IP: B 的 IP

**2. 使用 Open vSwitch （OVS）+ VLAN 组网**

Neutron 基于 VLAN 模式的 tenant network 同 provider network 一样，都必须使用物理的 VLAN 网络。

**2.1 物理 VLAN 网络配置**

本例子中，交换机上划分了三个 VLAN 区域：

1. 管理网络，用于 OpenStack 节点之间的通信，假设 VLAN ID 范围为 50 - 99.
2. 数据网络，用于虚拟机之间的通讯。由于Vlan模式下，租户建立的网络都具有独立的 Vlan ID，故需要将连接虚机的服务器的交换机端口设置为 Trunk 模式，并且设置所允许的 VLAN ID 范围，比如 100~300。
3. 外部网络，用于连接外部网络。加上 VLAN ID 范围为 1000-1010。

（图8）

关于网段之间的路由：

* 如果该物理交换机接到一个物理路由器并做相应的配置，则数据网络可以使用这个物理路由器，而不需要使用 Neutron 的虚拟路由器。
* 如果不使用物理的路由器，可以在网络节点上配置虚拟路由器。

**2.2 Neutron 配置**

**2.2.1 配置进行**

控制节点上：

[复制代码](javascript:void(0);)

# vim /etc/neutron/plugins/ml2/ml2\_conf.ini

[ml2]

type\_drivers = flat,vlan   
tenant\_network\_types = vlan  
mechanism\_drivers = openvswitch

[ml2\_type\_flat]  
flat\_networks = external

[ml2\_type\_vlan]   
network\_vlan\_ranges = physnet1:100:300

[复制代码](javascript:void(0);)

网络节点上：

[复制代码](javascript:void(0);)

#为连接物理交换机的网卡 eth2 和 eth3 建立 OVS physical bridge，其中，eth2 用于数据网络，eth3 用于外部网络

ovs-vsctl add-br br-eth2  
ovs-vsctl add-br br-ex

ovs-vsctl add-port br-eth2 eth2  
ovs-vsctl add-port br-ex eth3

# vim /etc/neutron/plugins/ml2/ml2\_conf.ini   
[m12]

type\_drivers = flat,vlan  
tenant\_network\_types = vlan  
mechanism\_drivers = openvswitch  
  
[ml2\_type\_flat]  
flat\_networks = external

[ml2\_type\_vlan]   
network\_vlan\_ranges = physnet1:100:300,external:1000:1010   
  
[ovs]   
bridge\_mappings = physnet1:br-eth2,external:br-ex

[复制代码](javascript:void(0);)

计算节点上：

[复制代码](javascript:void(0);)

#为连接物理交换机的网卡 eth2 建立 OVS physical bridge

ovs-vsctl add-br br-eth2

ovs-vsctl add-port br-eth2 eth2

# vim /etc/neutron/plugins/ml2/ml2\_conf.ini   
[m12]

type\_drivers = vlan  
tenant\_network\_types = vlan  
mechanism\_drivers = openvswitch

[ml2\_type\_vlan]   
network\_vlan\_ranges = physnet1:100:300   
  
[ovs]   
bridge\_mappings = physnet1:br-eth2

[复制代码](javascript:void(0);)

注意：

* network\_vlan\_ranges 中的 VLAN ID 必须和物理交换机上的 VLAN ID 区间一致。
* bridge\_mappings 中所指定的 bridge 需要和在个节点上手工创建的 OVS bridge 一致。

然后重启相应的 Neutron 服务。

**2.2.2 配置生效过程**

当 Neutron L2 Agent （OVS Agent 或者 Linux Bridge agent）在计算和网络节点上启动时，它会根据各种配置在节点上创建各种 bridge。以 OVS Agent 为例，

（1）创建 intergration brige（默认是 br-int）；如果 enable\_tunneling = true 的话，创建 tunnel bridge （默认是 br-tun）。

（2）根据 bridge\_mappings，配置每一个 VLAN 和 Flat 网络使用的 physical network interface 对应的预先创建的 OVS bridge。

（3）所有虚机的 VIF 都是连接到 integration bridge。同一个虚拟网络上的 VM VIF 共享一个本地 VLAN （local VLAN）。Local VLAN ID 被映射到虚拟网络对应的物理网络的 segmentation\_id。

（4）对于 GRE 类型的虚拟网络，使用 LSI （Logical Switch identifier）来区分隧道（tunnel）内的租户网络流量（tenant traffic）。这个隧道的两端都是每个物理服务器上的 tunneling bridge。使用 Patch port 来将 br-int 和 br-tun 连接起来。

（5）对于每一个 VLAN 或者 Flat 类型的网络，使用一个 veth 或者一个 patch port 对来连接 br-int 和物理网桥，以及增加 flow rules等。

（6）最后，Neutron L2 Agent 启动后会运行一个RPC循环任务来处理 端口添加、删除和修改。管理员可以通过配置项 polling\_interval 指定该 RPC 循环任务的执行间隔，默认为2秒。

**2.3 创建虚拟网络和子网**

**2.3.1 创建命令**

[复制代码](javascript:void(0);)

s1@controller:~$ neutron net-create net1 （或者 Admin 用户运行 neutron net-create net1 --provider:network\_type vlan --provider:physical\_network physnet1 --provider:segmentation\_id 101。效果相同）

Created a new network:

+---------------------------+--------------------------------------+

| Field | Value |

+---------------------------+--------------------------------------+

| admin\_state\_up | True |

| id | dfc74f44-a9f2-4497-a53d-1723804a49a8 |

| name | net1 |

| provider:network\_type | vlan |

| provider:physical\_network | physnet1 |

| provider:segmentation\_id | 101 |

| router:external | False |

| shared | False |

| status | ACTIVE |

| subnets | |

| tenant\_id | 74c8ada23a3449f888d9e19b76d13aab |

+---------------------------+--------------------------------------+

s1@controller:~$ neutron subnet-create subnet1 10.0.0.0/24 --name net1

[复制代码](javascript:void(0);)

**2.3.2 Neutron 代码实现**

做完以上的步骤之后，用户就可以在 subnet 上 boot 虚机了。

boot 虚机的过程中，Nova 依次会：

（1）调用 Neutron REST API 申请一个或者多个 port。Neutron 会根据数据库中的配置来进行分配。

（2）在计算节点上，Nova 调用 ovs-vsctl 命令将虚机的 VIF 被 plug 到 br-int 上。

（3）启动虚机。

Neutron L2 Agent 的循环任务每隔两秒会依次：

（1）调用 ”ovs-vsctl list-ports“ 命令获取到 br-int 上的 port，再根据上次保存的历史数据，生成所有变更端口的列表（包括添加的、更新的、删除的端口）。比如：

{'current': set([u'04646b21-78a0-429e-85be-3167042b77be', u'592740b0-0768-4e57-870d-6495e6c22135']), 'removed': set([]), 'added': set([u'04646b21-78a0-429e-85be-3167042b77be', u'592740b0-0768-4e57-870d-6495e6c22135'])}

（2）为每一个待处理端口，根据其 ID 从 DB 中取得其详细信息。比如：

{u'profile': {}, u'admin\_state\_up': True, u'network\_id': u'e2022937-ec2a-467a-8cf1-f642a3f777b6', u'segmentation\_id': 4, u'device\_owner': u'compute:nova', u'physical\_network': phynet1, u'mac\_address': u'fa:16:3e:fd:ed:22', u'device': u'592740b0-0768-4e57-870d-6495e6c22135', u'port\_id': u'592740b0-0768-4e57-870d-6495e6c22135', u'fixed\_ips': [{u'subnet\_id': u'13888749-12b3-462e-9afe-c527bd0a297e', u'ip\_address': u'91.1.180.4'}], u'network\_type': u'vlan'}

（3）针对每一个增加或者变更的 port，设置 local VLAN Tag；调用 ”ovs-ofctl mod-flows “ 命令来设置 br-tun 或者 物理 bridge 的 flow rules；并设置 db 中其状态为 up。

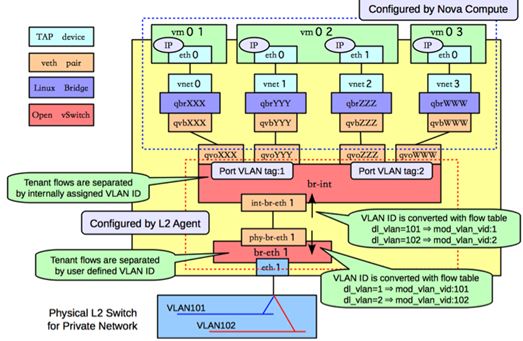
（4）针对每一个被删除的 port，设置 db 中其状态为 down。

**2.4 Neutron 虚拟网络**

（1）一个计算节点上的网络实例

它反映的网络配置如下：

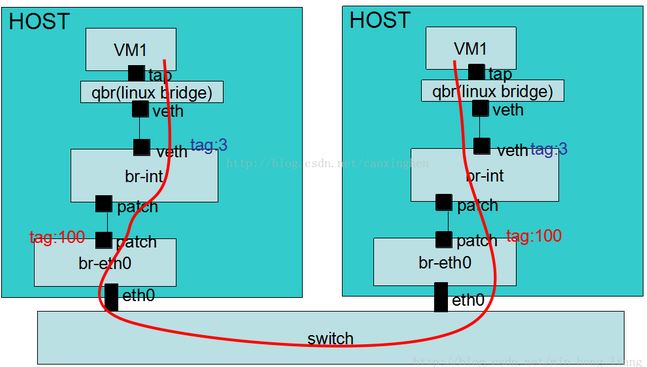
1. Neutron 使用 Open vSiwtch。
2. 一台物理服务器，网卡 eth1 接入物理交换机，预先配置了网桥 br-eth1。
3. 创建了两个 neutron VLAN network，分别使用 VLAN ID 101 和 102。
4. 该服务器上运行三个虚机，虚机1 和 2 分别有一个网卡接入 network 1；虚机2 和 3 分别有一个网卡接入 network 2.

（图9）

Neutron 在该计算节点上做的事情：

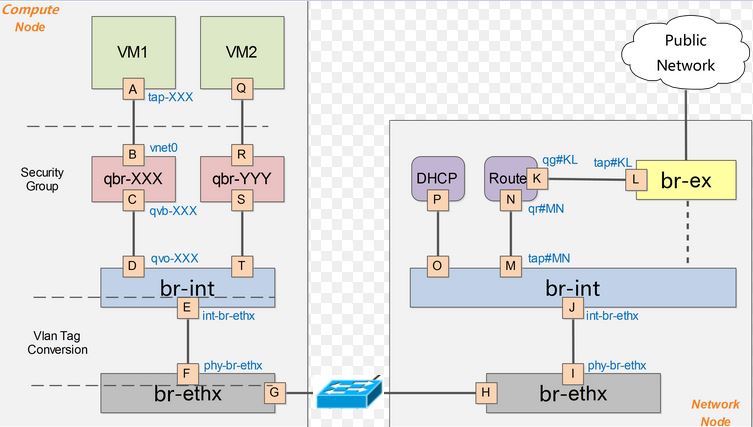
* 创建了 OVS Integration bridge br-int。它的四个 Access 端口中，两个打上了内部 Tag 1，连接接入 network 1 的两个网卡；另两个端口的 VLAN Tag 为 2。
* 创建了一对 patch port，连接 br-int 和 br-eth1。
* 设置 br-int 中的 flow rules。对从 access ports 进入的数据帧，加上相应的 VLAN Tag，转发到 patch port；从 patch port 进入的数据帧，将 VLAN ID 101 修改为 1, 102 修改为 2，再转发到相应的 Access ports。
* 设置 br-eth1 中的 flow rules。从 patch port 进入的数据帧，将内部 VLAN ID 1 修改为 101，内部 VLAN ID 2 修改为 102，再从 eth1 端口发出。对从 eth1 进入的数据帧做相反的处理。

（2）再加上另一个连接到同一个物理交换机的服务器（加上 neutron 网络使用的 VLAN ID 为 100，物理 brige 为 br-eth0）：

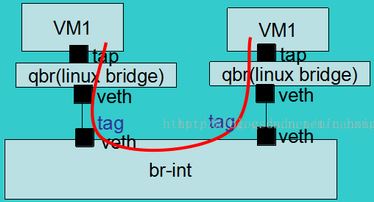
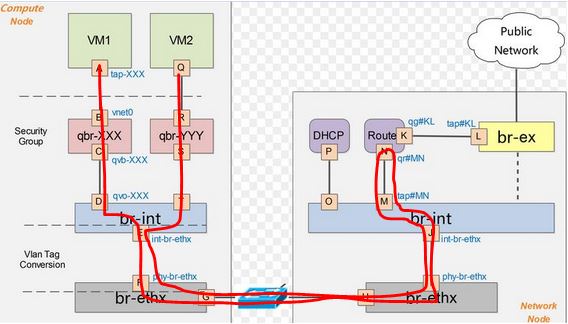
（图10）

Neutron 实现了基于物理 VLAN 交换机的跨物理服务器二层虚拟网络。

（3）连接到同一物理交换机的网络节点的情况

（图11）

（4）网络流向

* 不同物理服务器上的虚机，如果 VM1 和 VM2 属于同一个 tenant network 的同一个subnet，那么两者的通信直接经过 物理交换机 进行，不需要做到网络节点。如图10 所示。
* 相同物理服务器上的虚机，如果 VM1 和 VM2 属于同一个 tenant network 的同一个subnet，那么两者的通信直接经过 br-int 进行。
* 
* 对其他虚机之间数据交换情形，都算作跨子网的数据流向，都需要经过网络节点中的 Router 进行 IP 包的路由。（也可以直接使用连接物理交换机的物理路由器）。
* 

更详细的网络流向分析可以参考我另外的几篇文章：

* [探索 OpenStack 之（8）：Neutron 深入探索之 OVS + GRE 之 完整网络流程 篇](http://www.cnblogs.com/sammyliu/p/4204190.html)
* [探索 OpenStack 之（7）：Neutron 深入探索之 Open vSwitch (OVS) + GRE 之 Neutron节点篇](http://www.cnblogs.com/sammyliu/p/4201721.html)
* [学习OpenStack之（6）：Neutron 深入学习之 OVS + GRE 之 Compute node 篇](http://www.cnblogs.com/sammyliu/p/4201143.html)

注：本文所有的图片皆来自互联网。