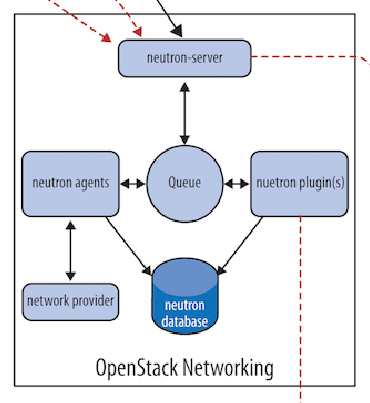


前面我们讨论了 Neutron 的基本概念，今天我们开始分析 Neutron 的架构。

**Neutron 架构**

与 OpenStack 的其他服务的设计思路一样，Neutron 也是采用分布式架构，由多个组件（子服务）共同对外提供网络服务。



Neutron 由如下组件构成：

**Neutron Server**对外提供 OpenStack 网络 API，接收请求，并调用 Plugin 处理请求。

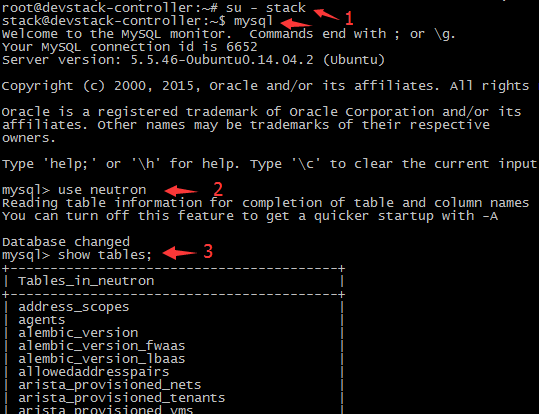
**Plugin**处理 Neutron Server 发来的请求，维护 OpenStack 逻辑网络的状态， 并调用 Agent 处理请求。

**Agent**处理 Plugin 的请求，负责在 network provider 上真正实现各种网络功能。

**network provider**提供网络服务的虚拟或物理网络设备，例如 Linux Bridge，Open vSwitch 或者其他支持 Neutron 的物理交换机。

**Queue**Neutron Server，Plugin 和 Agent 之间通过 Messaging Queue 通信和调用。

**Database**存放 OpenStack 的网络状态信息，包括 Network, Subnet, Port, Router 等。



Neutron 架构非常灵活，层次较多，其目的是：

1. 为了支持各种现有或者将来会出现的优秀网络技术。
2. 支持分布式部署，获得足够的扩展性。

通常鱼和熊掌不能兼得，虽然获得了这些优势，但这样使得 Neutron 更加复杂，更不容易理解。

后面我们会详细讨论 Neutron 的各个组件，但在这之前，非常有必要先通过一个例子了解这些组件各自的职责以及是如何协同工作。

以创建一个 VLAN100 的 network 为例，假设 network provider 是 linux bridge， 流程如下：

1. Neutron Server 接收到创建 network 的请求，通过 Message Queue（RabbitMQ）通知已注册的 Linux Bridge Plugin。
2. Plugin 将要创建的 network 的信息（例如名称、VLAN ID等）保存到数据库中，并通过 Message Queue 通知运行在各节点上的 Agent。
3. Agent 收到消息后会在节点上的物理网卡（比如 eth2）上创建 VLAN 设备（比如 eth2.100），并创建 bridge （比如 brqXXX） 桥接 VLAN 设备。

关于 linux bridge 如何实现 VLAN 大家可以参考本教程“预备知识->网络虚拟化”的相关章节。

这里进行几点说明：

1. plugin 解决的是 What 的问题，即网络要配置成什么样子？而至于如何配置 How 的工作则交由 agent 完成。
2. plugin，agent 和 network provider 是配套使用的，比如上例中 network provider 是 linux bridge，那么就得使用 linux bridge 的 plungin 和 agent；如果 network provider 换成了 OVS 或者物理交换机，plugin 和 agent 也得替换。
3. plugin 的一个主要的职责是在数据库中维护 Neutron 网络的状态信息，这就造成一个问题：所有 network provider 的 plugin 都要编写一套非常类似的数据库访问代码。为了解决这个问题，Neutron 在 Havana 版本实现了一个 ML2（Modular Layer 2）plugin，对 plgin 的功能进行抽象和封装。有了 ML2 plugin，各种 network provider 无需开发自己的 plugin，只需要针对 ML2 开发相应的 driver 就可以了，工作量和难度都大大减少。ML2 会在后面详细讨论。
4. plugin 按照功能分为两类： core plugin 和 service plugin。core plugin 维护 Neutron 的 netowrk, subnet 和 port 相关资源的信息，与 core plugin 对应的 agent 包括 linux bridge, OVS 等； service plugin 提供 routing, firewall, load balance 等服务，也有相应的 agent。后面也会分别详细讨论。

以上是Neutron的逻辑架构，下一节我们讨论 Neutron 的物理部署方案。