OpenStack 网络管理指南

最新版本下载地址：https://github.com/yeasy/tech_writing/tree/master/OpenStack

v0.4:2013-05-30

补充 Grizzly 版本内容，更新 folsom 版本配置到 Grizzly。

v0.3:2013-04-11

补充内容。

v0.2:2013-04-10

完成 1-10 部分，剩余 11.附录部分。

v0.1: 2013-04-06

基于 Folsom 版本，开始整体结构。

1.1 概览

1.1.1 什么是 Quantum

作为 OpenStack 的子项目，Quantum 试图提供对网络的统一管理。一方面提供给租户丰富的网络管理 API（包括网络连通性和地址管理），另一方面提供给管理员选用多种不同的网络技术来管理网络的灵活性。

1.1.1.1 丰富的网络 API

Quantum 是一个虚拟网络服务，可以为其他 OpenStack 组件（主要是 Nova）提供管理网络连通性和地址服务。

OpenStack 的 Nova API 提供了服务器层面的虚拟化来描述计算资源，类似的，Quantum 则提供了网络层面的虚拟化，通过三种概念来描述网络资源：网络（Network）、子网（Subnet）和端口（Port）。

- 网络：隔离的二层网段，类似物理网络中的 Vlan 概念；
- 子网：一段 IPv4 或者 v6 地址段和配置状态；
- 端口：每个连接端口挂载一个单独的网络设备，比如虚拟机的网卡。还包括端口上绑定的 MAC 和 IP 地址信息等网络配置信息。

可以通过创建和配置网络、子网来定义任意的网络拓扑，然后让其他 OpenStack 组件（主要是 Nova）来挂载虚拟设备到这些网络中。特别的，Quantum 支持每个租户拥有多个私有的网络，并且允许租户自行配置 IP 地址信息（即便跟其他租户的 IP 地址信息冲突）。这给提供虚拟机迁移等服务提供了便利。

可以通过 API 扩展来进一步增强 Quantum 的功能。

1.1.1.2 选择不同的网络技术

传统网络很难扩展和更改配置。

一开始，Nova 的网络实现是通过 Linux Vlan 和 IP tables 来实现简单的隔离。Quantum 通过支持插件（对 Quantum API 的后端实现）来提供强大的网络管理功能。支持的网络技术除了简单的 Linux Vlan 和 IP tables（某些插件仍基于这些简单技术来实现），还包括 L3 上做 L2 隧道的技术和 OpenFlow 等技术（其他插件）。

现在的插件技术类型包括：

- **Open vSwitch**. http://www.openvswitch.org/openstack/documentation
- **Cisco**. quantum/plugins/cisco/README and http://wiki.openstack.org/cisco-quantum
- **Linux Bridge**. quantum/plugins/linuxbridge/README and http://wiki.openstack.org/Quantum-Linux-Bridge-Plugin
- **Nicira NVP**. quantum/plugins/nicira/nicira_nvp_plugin/README and http://www.nicira.com/support.
- **Ryu**. quantum/plugins/ryu/README and http://www.osrg.net/ryu/using_with_openstack.html
- **NEC OpenFlow**. http://wiki.openstack.org/Quantum-NEC-OpenFlow-Plugin
- **Big Switch, Floodlight REST Proxy**. http://www.openflowhub.org/display/floodlightcontroller/Quantum+REST+Proxy+Plugin
- **PLUMgrid** https://wiki.openstack.org/wiki/Plumgrid-quantum
- **Hyper-V Plugin**.
- **Brocade Plugin**.
- **Midonet Plugin**.

插件支持各种属性，下面列出了部分插件的兼容性。

表格 1

	Libvirt（KVM/QEMU）	XenServer	VMware	Hyper-V	Bare-metal	PowerVM
OpenvSwitch	Yes					
Linux Bridge	Yes					
Cisco	Yes					
Nicira NVP	Yes	Yes	Yes			
Ryu	Yes					
NEC	Yes					
Bigswitch/Floodlight	Yes					
Hyper-V				Yes		
Brocade	Yes					
Midonet	Yes					

Plumgrid	Yes					

1.1.2 Quantum 架构

Quantum 包括一系列的子程序组成。核心的程序是 quantum-server。quantum-server 是一个 Python 的 daemon，对外提供 Quantum 的 API，并接收用户请求转发给 Quantum 的插件。插件如果需要持久性的存储，需要访问数据库。

如果部署了单独一台主机作为控制器，运行集中式的 Nova 组件，那也可以把 Quantum 放到同一台机器上，或者单独一台机器上。根据部署的不同可能需要额外的一些代理：

- 插件代理：所有的 quantum-*-agent，为插件服务，运行在每个 hypervisor 上，管理本地的 vswitch 配置。某些插件并不需要代理。
- dhcp 代理（quantum-dhcp-agent）：quantum-dhcp-agent，为租户网络提供 DHCP 服务。该代理跟插件类型无关。
- L3 代理（quantum-l3-agent）：为租户网络中的 VM 提供对外网访问的 L3/NAT 转发。该代理跟插件类型无关。

这些代理通过两种方式跟 quantum-server 程序交互：

- 通过 RPC，例如 rabbitmq 或者 qpid。
- 通过标准的 Quantum API。

Quantum 依赖 Keystone 组件来提供 API 访问中的认证和鉴权。

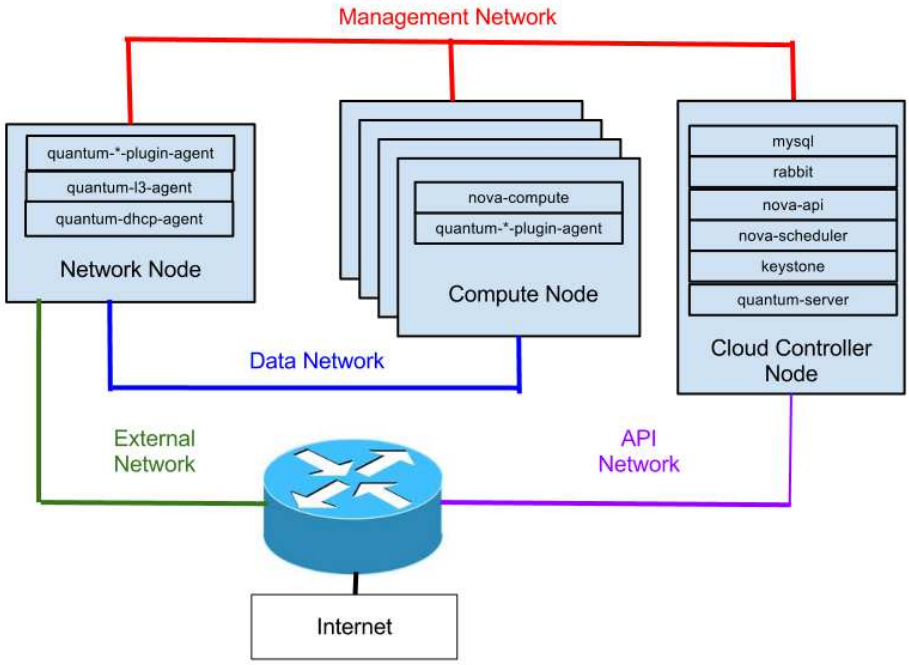
Nova 在创建 vm 的过程中，nova-compute 需要跟 Quantum API 通信把每个虚拟网卡端口插到某个 Quantum 网络上。

Horizon 则集成了 Quantum 的 API，允许租户通过 GUI 来创建网络和子网，并制定 VM 如何连接入网络。

在本文档中，我们需要一台控制器主机，一台网关主机（如果虚拟机需要发送或接收大量数据到互联网，则建议单独设置网关主机），和运行虚拟机的计算主机。

标准的 Quantum 安装包括 4 个独立的数据中心网络，如图表 1 所示。

- 管理网络（management network）：用于在 OpenStack 各个组件之间的内部通信。该网络的 IP 地址只能在数据中心内部可访问到。
- 数据网络（data network）：用于虚拟机数据传输，该网络的 IP 地址可根据使用的 Quantum 插件来灵活配置。
- 外部网络（External network）：用于提供一些带有 Internet 访问的虚拟机。该网络的 IP 地址必须可以从 Internet 上访问到。
- API 网络（API network）：提供所有的可用 API 给用户。该网络的 IP 地址必须是从 Internet 可访问的。该网络跟外部网络可以用同一个网络。



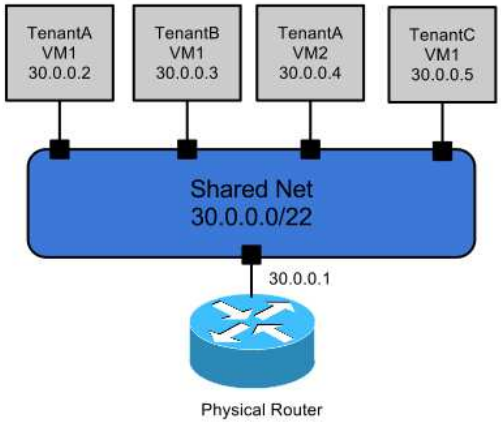
图表 1 OpenStack 中的四种网络

1.1.3 Quantum 部署用例

1.1.3.1 一个扁平网络

最简单的用例。该网络只有一个网络存在，所有的租户都可看到该网络。租户的虚拟机有一个网卡，获取到一个子网给定的 IP。不支持浮动 IP（floating IP）。

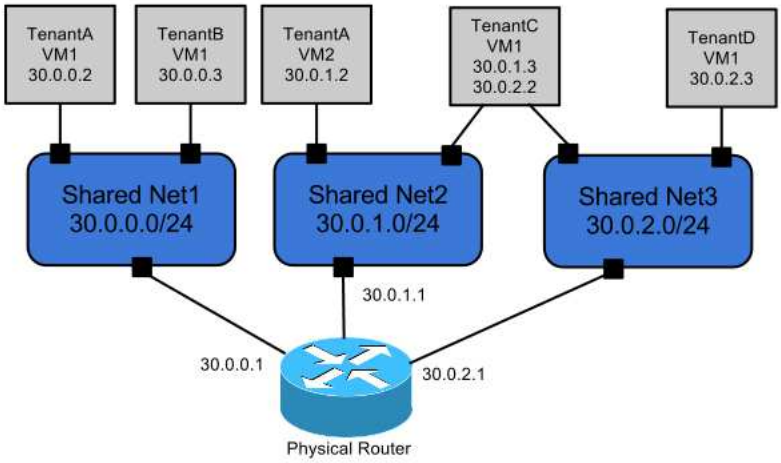
这样的网络通常情况下是一个提供者网络，意味着是由管理员创建，直接映射到数据中心中的一个实际存在的物理网络上。网络中的虚拟机可以通过一个路由器访问到外部网络。



图表 2 一个扁平网络

1.1.3.2 多个扁平网络

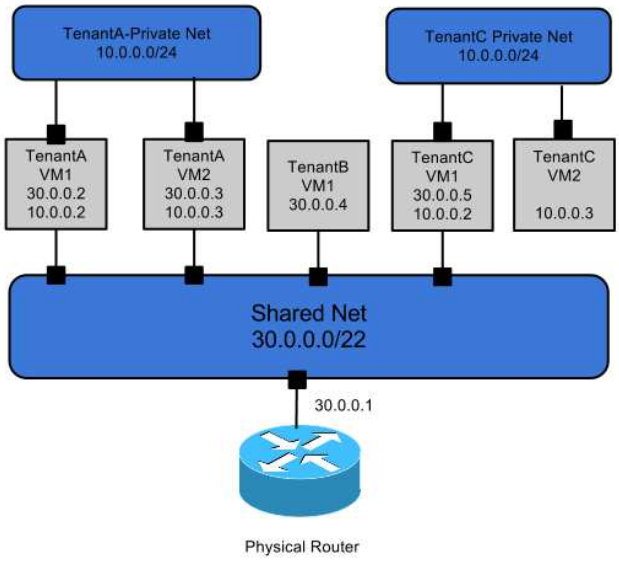
与一个扁平网络类似。不同的是，租户看到的是多个共享的网络，可以选择加入哪个网络。租户的虚拟机上可以有多个网卡，分别加入不同的网络。



图表 3 多个扁平网络。

1.1.3.3 混合扁平和私有网络

更进一步，租户除了共享的扁平网络之外，还可以拥有私有网络。这些私有网络只有该租户自身可以看到。当创建 VM 的时候，多个网卡可以分别配置到不同的网络中。



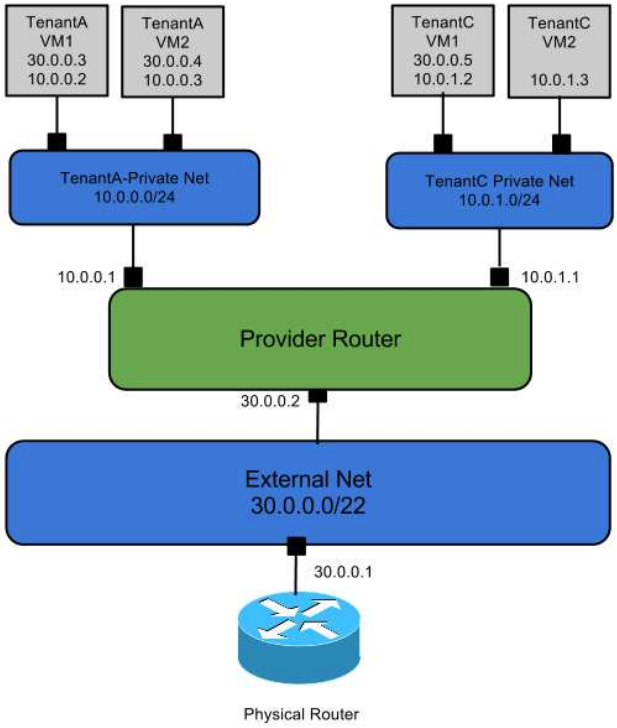
图表 4 混合扁平和私有网络

1.1.3.4 可路由私有网络

每个租户有用一个或多个私有网络，经过路由器连接到外面。每个租户只能看到分配到该租户的网络，看不到共享的网络。路由器由管理员来负责管理。

在这个模型里，VM 可以分配到公开的浮动 IP，路由器负责完成内外地址的转换。

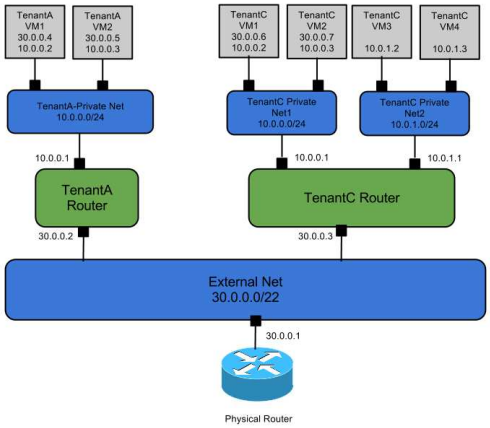
路由器提供在多个私有网络之间的 L3 连接，因此，在安全策略的允许下，租户主机之间可以相互访问。因为只有一台路由器，租户网络无法使用相互覆盖的 IP 地址。因此，这种情况下，大部分情况需要管理员为租户分配好网络地址。



图表 5 可路由的私有网络

1.1.3.5 每租户的可路由私有网络

更高级和复杂的情景是每个租户至少拥有一个路由器，并可能通过 API 来创建额外的路由器。租户可以创建自有的网络，并通过路由器把这些网络相互连通起来。这种模型下，用户可以拥有多级的、多个子网。不同租户子网之间 IP 地址可以任意相互覆盖。



图表 6 每租户的可路由私有网络

1.2 安装

1.2.1 安装包

使用 ubuntu cloud archive 来安装。

```
# echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom main >> /etc/apt/sources.list.d/folsom.list
# apt-get install ubuntu-cloud-keyring
# apt-get update
# apt-get upgrade
```

1.2.2 安装 quantum-server

安装 quantum-server 和访问 API 的 CLI。

```
apt-get -y install quantum-server python-quantumclient
```

根据需要安装可能的插件，比如 openvswitch：

```
apt-get -y install quantum-plugin-openvswitch
```

安装可能需要的数据库（如果其他组件已经安装了，也可以共用），例如：

```
apt-get -y install mysql-server python-mysqldb python-sqlalchemy
```

创建 quantum 数据库：

```
mysql -u <user> -p <pass> -e “create database quantum”
```

配置插件使用数据库，在/etc/quantum/plugins/<plugin-name>下找到插件的配置文件，设置：

```
sql_connection = mysql://<user>:<password>@localhost/quantum?charset=utf8
```

1.2.2.1 RPC 设置

Quantum 使用 RPC 来允许 DHCP 代理和其他的插件代理跟 quantum-server 程序进行通信。这个 RPC 机制跟其他组件中使用的一致。

为了使用 RabbitMQ 作为消息总线，需要确保 rabbit 已经安装在管理网络中一台可访问的主机上。

```
apt-get install rabbitmq-server
rabbitmqctl change_password guest <password>
```

然后更新/etc/quantum/quantum.conf 文件。

```
rabbit_host=<mgmt-IP-of-rabbit-host>
rabbit_password=<password>
rabbit_user=guest
```

1.2.2.2 配置 OpenvSwitch 插件

OpenvSwitch 插件可以提供多个网络的隔离服务，包括隧道和 vlan 等，以隧道为例。

在 quantum 服务主机上，修改/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini 文件，指定

```
enable_tunneling=True
tenant_network_type=gre
tunnel_id_ranges=1:1000
# only if node is running the agent
local_ip=<data-net-IP-address-of-node>
```

之后，重启 quantum-server 来启用新的配置。

```
service quantum-server restart
```

1.2.2.3 配置 Nicira NVP 插件

通过下面命令安装插件。

```
apt-get -y install quantum-plugin-nicira
```

要使用 NVP 插件，首先修改/etc/quantum/quantum.conf 文件，设置

```
core_plugin = quantum.plugins.nicira.nicira_nvp_plugin.QuantumPlugin.NvpPluginV2
```

之后，修改 NVP 的配置文件/etc/quantum/plugins/nicira/nvp.ini，指定数据库位置

```
sql_connection = mysql://<user>:<password>@localhost/quantum?charset=utf8
```

为了告诉 quantum 关于控制器集群的信息，在[NVP]段下创建[CLUSTER:<name>]段，指定传输域。

```
default_tz_uuid = <uuid_of_the_transport_zone>
```

指定集群中的控制器，为每个控制器添加一条：

```
nvp_controller_connection =
<ip>:<port>:<user>:<pw>:<req_timeout>:<http_timeout>:<retries>:<redirects>
```

当创建路由器的时候，默认用 NVP 的 L3 网关的 UUID，这个值可以通过 NVP Manager 网关服务页上找到。

```
default_l3_gw_service_uuid = <uuid_of_the_gateway_service>
```

另外，检查/etc/default/quantum-server 中的配置，确保为

```
QUANTUM_PLUGIN_CONFIG = /etc/quantum/plugins/nicira/nvp.ini
```

最后，重启 quantum-server 来启用插件。

```
service quantum-server restart
```

一个 quantum.conf 文件的例子为：

```
core_plugin = quantum.plugins.nicira.nicira_nvp_plugin.QuantumPlugin.NvpPluginV2
rabbit_host = 192.168.203.10
allow_overlapping_ips = True
```

一个 nvp.ini 文件的例子为：

```
[DATABASE]
sql_connection=mysql://root:root@127.0.0.1/quantum

[CLUSTER:main]
default_tz_uuid = d3afb164-b263-4aaa-a3e4-48e0e09bb33c
default_l3_gw_service_uuid=5c8622cc-240a-40a1-9693-e6a5fca4e3cf
nvp_controller_connection=10.0.0.2:443:admin:admin:30:10:2:2
nvp_controller_connection=10.0.0.3:443:admin:admin:30:10:2:2
nvp_controller_connection=10.0.0.4:443:admin:admin:30:10:2:2
```

在网络节点上可以运行如下命令来检查 nvp.ini 是否配置正确。

```
check-nvp-config <path/to/nvp.ini>
```

1.2.2.4 配置 bigswitch Floodlight REST Proxy 插件

首先编辑/etc/quantum/quantum.conf 文件，设置

```
core_plugin = quantum.plugins.bigswitch.plugin.QuantumRestProxyV2
```

编辑/etc/quantum/plugins/bigswitch/restproxy.ini 文件，设置 sql 连接信息和指定控制器 rest 接口连接信息

```
sql_connection =
mysql://<user>:<password>@localhost/restproxy_quantum?charset=utf8
server = <controller-ip>:<port>
```

编辑/etc/default/quantum-server，指定 plugin 的配置文件位置

```
QUANTUM_PLUGIN_CONFIG="/etc/quantum/plugins/bigswitch/restproxy.ini"
```

最后重启 quantum-server。

```
service quantum-server restart
```

1.2.3 在数据转发节点上安装软件

插件通常需要一些特定软件来处理数据包。包括在 nova-compute 节点上，运行一些特定的网络服务代理的节点（quantum-dhcp-agent、quantum-l3-agent）上。

通常，这些数据转发节点都拥有两个接口：一个管理网络接口，一个数据网络接口。

1.2.3.1 OpenvSwitch 插件

在数据转发节点上需要安装 quantum-plugin-openvswitch-agent：

```
apt-get -y install quantum-plugin-openvswitch-agent
```

安装会创建配置文件 ovs_quantum_plugin.ini。当使用隧道时，每个运行代理的节点需要有一个 IP 地址，在配置文件中的 local_ip 值来指定，之后重启代理服务。

```
service quantum-plugin-openvswitch-agent restart
```

代理需要在每个节点上创建网桥“br-int”。

```
ovs-vsctl add-br br-int
```

1.2.3.2 Nicira NVP 插件

该插件需要安装 OpenvSwitch，不需要额外的代理。

但需要注意，默认自带的 OpenvSwitch 并不兼容 NVP，需要手动安装兼容版本。

1.2.4 安装 DHCP 代理

运行 quantum-server 的主机上需要至少两个接口，一个连接到管理网络，一个连接到数据网络。

```
apt-get -y install quantum-dhcp-agent
```

配置文件在/etc/quantum/dhcp_agent.ini。

1.2.4.1 OVS 插件

下面的 DHCP 代理选项是 OVS 插件需要的。

```
[DEFAULT]
ovs_use_veth = True
enable_isolated_metadata = True
use_namespaces = True
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
```

1.2.4.2 NVP 插件

下面的 DHCP 代理选项是 NVP 插件需要的。

```
[DEFAULT]
```

```
ovs_use_veth = True
enable_metadata_network = True
enable_isolated_metadata = True
use_namespaces = True
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
```

1.2.5 安装 L3 代理

L3 代理可以让 Quantum 创建路由器来连接不同的 L2 网络。很多插件都需要依赖 L3 代理，然而下面的这些插件不能跟 Quantum 自带的 L3 代理一起使用。

- Nicira NVP Plugin
- Floodlight/Bigswitch Plugin
- PLUMgrid Plugin

在网络节点上安装代理。

```
apt-get -y install quantum-l3-agent
```

安装后创建网桥“br-ex”，用于将节点上接到外部网络，然后将网卡挂载到网桥上。例如使用 OpenvSwitch 和网卡 eth1，则运行：

```
ovs-vsctl add-br br-ex
ovs-vsctl add-port br-ex eth1
```

运行 quantum-l3-agent 的节点不应该自己配置所用连接外网网卡的地址信息。Quantum 会自行分配一些 ip 用于完成路由。

quantum-l3-agent 使用 Linux IP 栈和 iptables 来进行 L3 的转发和 NAT。为了提供多个路由器之间可能的 IP 地址覆盖，quantum-l3-agent 默认使用 Linux 网络名字空间来提供转发隔离。因此，节点上的 ip 地址无法采用常规的 ip adder list 或 ifconfig 等看到，同时也无法直接 ping 通 ip 地址。要完成这些功能，必须在路由器指定的网络名字空间中进行。名字空间的名字类似“qrouter-<UUID of the router>”。如下面例子所示：

```
ip netns exec qrouter-47af3868-0fa8-4447-85f6-1304de32153b ip addr list
ip netns exec qrouter-47af3868-0fa8-4447-85f6-1304de32153b ping <fixed-ip>
```

1.2.6 安装 LBaas 代理

Quantum 通过在网络节点上运行 quantum-lbaas-agent 来提供 Load balancing 服务。通过如下命令安装

apt-get install quantum-lbaas-agent

如果使用基于 OVS 的插件（OVS、NVP、Ryu、NEC、BigSwitch），需要配置

```
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
```

否则，如果使用 Linux Bridge，则需要设置

```
interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver
```

为了使用参考实现，需要配置

```
device_driver = quantum.plugins.services.agent_loadbalancer.drivers.haproxy.namespace_driver.HaproxyNSDriver
```

最后，需要在网络节点上的 quantum.conf 中确保配置了：

```
service_plugins = quantum.plugins.services.agent_loadbalancer.plugin.LoadBalancerPlugin
```

1.2.7 安装 Quantum CLI 客户端

在 quantum-server 节点上执行命令：

```
apt-get -y install python-pyparsing python-cliff python-quantumclient
```

1.2.8 初始化、配置和日志文件位置

quantum 服务可以通过 service 命令来进行管理，例如

```
service quantum-server stop
service quantum-server status
service quantum-server start
service quantum-server restart
```

Log 文件在/var/log/quantum。

配置文件在/etc/quantum。

1.3 配置其他相关组件

1.3.1 Keystone

1.3.1.1 创建 Quantum 服务项

Quantum 需要在 Keystone 的登记服务中进行登记。根据使用的是 Keystone 的 SQL 目录驱动，还是模板目录驱动（template catalog driver），步骤略有不同。

如果是 SQL 驱动，对于 Quantum 给定区域、IP 地址（Quantum 服务器）和服务 ID，运行

```
keystone service-create --name quantum --type network --description 'OpenStack Networking Service'
```

记录返回的 ID，并放进下面命令$ID 的位置，创建 endpoint。

```
keystone endpoint-create --region $REGION --service-id $ID --publicurl 'http://$IP:9696/' --adminurl 'http://$IP:9696/' --internalurl 'http://$IP:9696/'
```

例如：

```
$ keystone service-create --name quantum --type network --description 'OpenStack Networking Service'
```

```
+-------------+----------------------------------+
|    Property |                 Value            |
+-------------+----------------------------------+
| description | OpenStack Networking Service     |
| id          | 26a55b340e254ad5bb78c0b14391e153 |
| name        | quantum                          |
| type        | network                          |
+-------------+----------------------------------+
```

```
$ keystone endpoint-create --region myregion --service-id 26a55b340e254ad5bb78c0b14391e153 \
 --publicurl "http://10.211.55.17:9696/" --adminurl "http://10.211.55.17:9696/" --internalurl http://10.211.55.17:9696/
```

如果使用的是模板驱动，对于 Quantum 给定区域、IP 地址（Quantum 服务器）和服务 ID，则添加如下内容到模板文件（default_catalog.templates）中：

```
catalog.$REGION.network.publicURL = http://$IP:9696
catalog.$REGION.network.adminURL = http://$IP:9696
catalog.$REGION.network.internalURL = http://$IP:9696
catalog.$REGION.network.name = Network Service
```

一个真实的例子是：

```
catalog.$Region.network.publicURL = http://10.211.55.17:9696
catalog.$Region.network.adminURL = http://10.211.55.17:9696
catalog.$Region.network.internalURL = http://10.211.55.17:9696
catalog.$Region.network.name = Network Service
```

1.3.1.2 创建 Quantum 服务用户

Nova 或者一些内部组件要想使用 Quantum 的 API，必须提供管理用户的权限。一种推荐的方式是创建一个专门的“service”租户，并在该租户中创建一个“quantum”用户，并指定用户一个管理员的角色。

样例为：

```
$ ADMIN_ROLE=$(get_id keystone role-create --name=admin)
```

```
$ QUANTUM_USER=$(get_id keystone user-create --name=quantum --pass="$QUANTUM_PASSWORD" --email=demo@example.com --tenant-id service)
$ keystone user-role-add --user_id $QUANTUM_USER --role_id $ADMIN_ROLE --tenant_id service
```

1.3.2 Nova

在使用 quantum 的情况下，不能运行 nova-network。相反的，nova 把所有网络相关的决策都发给 Quantum。如果节点之前安装了 nova-network，推荐卸载 nova-network，并重启物理节点。

1.3.2.1 配置 Nova 访问 Quantum API

在提供 vm 和关闭 vm 服务时，nova 都需要跟 Quantum 的 API 进行交互。在 nova.conf 配置文件中需要有如下的信息：

network_api_class：必须修改为 nova.network.quantumv2.api.API，指定使用 Quantum 作为网络服务；

quantum_url：必须指定 quantum 服务器的主机名/IP 地址和端口信息。

quantum_auth_strategy：必须保留默认的“keystone”。

quantum_admin_tenant_name：必须指定为在 keystone 配置中创建的租户名（即“service”）。

quantum_admin_username：必须指定为在 keystone 配置中创建的用户名（即“quantum”）。

quantum_admin_password：必须指定为在 keystone 配置中创建的用户密码。

quantum_admin_auth_url：必须修改为 keystone 服务器的 IP 和端口。是默认的管理 API 服务器的 IP 和端口。

1.3.2.2 在 Nova 中配置 Vif-plugging

当 nova-compute 创建 vm 时，必须将每个虚拟机的各个网卡插入到 Quantum 控制的虚拟交换机上，并告诉虚拟交换机，每个网卡的 Quantum 端口 id。这需要在 nova.conf 中指定 vif-plugging 的类型，这个类型依赖于具体使用的插件。

1.3.2.2.1 基于 OpenvSwitch

选择面向 OpenvSwitch 的 vif-plugging 取决于使用的 libvirt 版本和是否使用 Nova 的安全过滤。

当使用安全过滤时候，对任意版本的 libvirt，需要设置 libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver；

当使用<0.9.11 版本的 libvirt 且不需要使用安全过滤时，设置 libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver；

当使用>=0.9.11 版本的 libvirt 且不需要使用安全过滤时，设置 libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver。

1.3.2.2.2 基于 Linux Bridge 插件

设置 libvirt_vif_driver=nova.virt.libvirt.vif.QuantumLinuxBridgeVIFDriver。

1.3.2.2.3 基于 Nicira NVP 插件

当 libvirt 版本<0.9.11 时，libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver；

当 libvirt 版本>=0.9.11 时，libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver；

当使用 XenServer 时，xenapi_vif_driver=nova.virt.xenapi.vif.XenAPIOpenVswitchDriver。

1.3.2.2.4 基于 NEC OpenFlow 插件

跟 OpenvSwitch 类似。

当使用安全过滤时候，对任意版本的 libvirt，需要设置 libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver；

当使用<0.9.11 版本的 libvirt 且不需要使用安全过滤时，设置 libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver；

当使用>=0.9.11 版本的 libvirt 且不需要使用安全过滤时，设置 libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver。

1.3.2.3 Nova.conf 样例（为 nova-compute 和 nova-api）

假设一个控制节点运行 Keystone 和 Quantum，IP 地址为 192.168.1.2，并且 vif-plugging 使用 LibvirtOpenVswitchDriver，启用了 virtio。则配置为：

```
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.1.2:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://192.168.1.2:35357/v2.0

# needed only for nova-compute
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver
libvirt_use_virtio_for_bridges=True
```

1.4 使用 Quantum

使用 Quantum 主要有两类目的，一个是为云租户提供 Quantum 的 API，来创建任意的网络拓扑；一个是为云管理员提供网络连通性的管理。本文档中大部分的操作是面向租户的。

1.4.1 核心的 API 功能

Quantum 安装成功后，租户即可以通过 API 或 CLI 工具（调用 API）来进行 create-read-update-delete（CRUD）操作。跟其他组件中类似，CLI 仅仅是封装并调用了后台 API 而已。

1.4.1.1 API 抽象

Quantum v2.0 API 提供了基于 L2 网络和 IP 网络的控制。并同时提供了类似于 nova-network 的基于 L3 的简单转发和 NAT。

在 Quantum 的 API 中，一个隔离的 L2 网段（类似于 Vlan）被称为“网络（network）”，提供 L2 网络拓扑的基础。

“子网（Subnet）”用于描述 Quantum 网络中的一些 IP 地址块和其他的网络配置（默认网关、dns 服务器）。每个 Quantum 网络可以拥有多个子网。

“端口（Port）”标识一个 L2 网络挂载接口。当端口被创建时，默认分配到一个对应子网的 IP 地址。当端口被删除时，分配的 IP 地址返回子网的 IP 池。用户可以指定 IP，或者让 Quantum 来自动设置。

下面几张表中总结了这几个抽象类型的各个属性。

表格 2 网络

属性名称	类型	默认值	描述
id	uuid-str	生成	网络的 uuid
name	字符串	None	可读的名称，不一定唯一
admin_state_up	布尔	真	网络的管理状态，如果为假，则网络不转发包。
status	字符串	N/A	是否正在运行
subnets	列表（uuid-str）	空列表	网络的子网
shared	布尔	假	是否任意租户都可以访问该网络资源
tenant_id	uuid-str	N/A	网络拥有者

表格 3 子网属性

属性名称	类型	默认值	描述
id	uuid-str	生成	子网的 uuid
network_id	uuid-str	N/A	子网所属的网络
name	字符串	None	可读的名称，不一定唯一
ip_version	整数类型	4	IP 版本
cidr	字符串	N/A	子网的 IP 范围
gateway_ip	字符串	cidr 中的首个地址	子网中设备的默认网

			关
dns_nameservers	列表（字符串）	空列表	子网中主机的 dns 服务器
allocation_pools	列表（字典）	cidr 中的每个地址	cidr 中的子范围
host_routes	列表（字典）	空列表	子网内的路由
enable_dhcp	布尔	真	子网中是否使用 dhcp
tenant_id	uuid	N/A	网络拥有者

表格 4 端口属性

属性名称	类型	默认值	描述
id	uuid-str	生成	端口的 uuid
network_id	uuid-str	N/A	子网所属的网络
name	字符串	None	可读的名称，不一定唯一
status	字符串	N/A	是否正在运行
mac_address	字符串	生成	端口的 MAC 地址
fixed_ips	列表（字典）	自动从池中分配	从子网中指定端口的 IP 地址
device_id	字符串		使用该端口的设备
device_owner	字符串		使用该端口的实体（比如 dhcp 代理）
tenant_id	uuid	N/A	网络拥有者

1.4.1.2 Quantum CLI 指南

参考 OpenStack CLi 指南：http://docs.openstack.org/cli/quick-start/content/index.html。

1.4.1.3 基本操作

在 quantum 服务器上执行命令，创建一个网络和一个子网。

```
quantum net-create net1
```

在网络上创建一个子网。

```
quantum subnet-create net1 10.0.0.0/24
```

列出租户能看到的端口。

```
quantum port-list
```

device_owner 域描述了端口的所有者，端口所有者以“network：”开头的说明是被网络服务创建的，端口所有者以“compute：”开头的说明是被计算服务创建的。

```
quantum port-list -c id -c fixed_ips -c device_owner
```

port-show 给出了某个端口的详细描述。

```
quantum port-show <port-id>
```

1.4.1.4 管理员 API 配置

管理员可以通过指定 tenant_id 来针对某个 tenant 来执行命令，例如

```
quantum net-create net1 --tenant_id=<tenant-id>
```

这些 tenant_id 必须是从 keystone 中获取的，可以通过如下命令：

```
keystone tenant-list
```

1.4.1.5 高级 API 操作用例

1.4.1.5.1 创建网络、子网、端口

创建一个共享网络。

```
quantum net-create public-net --shared True
```

创建一个子网，指定网关 IP。

```
quantum subnet-create --gateway 10.0.0.254 net1 10.0.0.0/24
```

创建一个没有指定网关的子网。

```
quantum subnet-create --no-gateway net1 10.0.0.0/24
```

创建禁用 dhcp 的子网。

```
quantum subnet-create net1 10.0.0.0/24 --enable_dhcp False
```

创建指定主机路由的子网。

```
quantum subnet-create test-net1 40.0.0.0/24 --host_routes type=dict list=true destination=40.0.1.0/24,nexthop=40.0.0.2
```

创建指定 dns 服务器的子网。

```
quantum subnet-create test-net1 40.0.0.0/24 --dns_nameservers list=true 8.8.8.7 8.8.8.8
```

1.4.1.5.2 搜索

搜索某个网络上的所有端口和 IP 信息。

```
quantum port-list -- --network_id <net-id>
```

1.4.1.5.3 高级创建 VM 和端口操作

启动一台拥有多网卡的虚拟机，指定每个网卡接入网络的 id。

```
nova boot --image <img> --flavor <flavor> --nic net-id=<net1-id> --nic net-id=<net2-id> <vm-name>
```

先指定端口 id，之后启动虚拟机绑定到该端口。

```
quantum port-create --fixed-ip subnet_id=<subnet-id>,ip_address=192.168.57.101 net1
nova boot --image <img> --flavor <flavor> --nic port-id=<port-id> <vm-name>
```

创建虚拟机，不指定网卡，则默认情况下该虚拟机接入到用户可访问的所有网络中。

```
nova boot --image <img> --flavor <flavor> <vm-name>
```

1.4.2 跟 Quantum 一起使用计算服务

1.4.2.1 基本流程

查看当前的网络。

```
quantum net-list
```

启动一个单网卡的虚拟机，并绑定到指定网络上。

```
nova boot --image <img> --flavor <flavor> --nic net-id=<net-id> <vm-name>
```

现在已经拥有了一个带有一台虚拟机的网络，net1 上带有一个端口连接到虚拟机，可以查看该端口。

```
quantum port-list -- --device_id=<vm-id>
```

如果只查看端口的某几个域，则可以通过-c 来指定，例如查看端口的 mac 地址。

```
quantum port-list -c mac_address -- --device_id=<vm-id>
```

禁用端口，可以通过设置 admin_state_up=False。

```
quantum port-update <port-id> --admin_state_up=False
```

在 Nova 中删除 vm 之后，对应的 Quantum 端口会自动被删除，查看信息：

```
quantum port-list -c mac_address -- --device_id=<vm-id>
```

1.4.2.2 高级 VM 创建

启动一个具有多个网卡的 VM。

```
nova boot --image <img> --flavor <flavor> \
--nic net-id=<net1-id> --nic net-id=<net2-id> <vm-name>
```

启动一台 VM，指定 IP。首先创建一个具有某个指定 IP 的端口，然后再启动 VM。

```
quantum port-create --fixed-ip subnet_id=<subnet-id>,ip_address=192.168.57.101 <net-id>
nova boot --image <img> --flavor <flavor> --nic port-id=<port-id> <vm-name>
```

如果不指定—nic 选项，则 VM 启动后，自动连接到租户所能访问的所有的网络上。

```
nova boot --image <img> --flavor <flavor> <vm-name>
```

1.4.2.3 安全组

如果插件支持 quantum 安全策略，要想通过网络访问 vm，需要添加相应的安全策略。

```
quantum security-group-rule-create --protocol icmp --direction ingress default
quantum security-group-rule-create --protocol tcp --port-range-min 22 --port-range-max 22 --direction ingress default
```

如果插件并没有实现 quantum 安全策略，则通过 nova 来添加规则。

```
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

如果插件实现了 quantum 安全策略，用户仍然可以通过 nova 来管理安全策略，需要在 nova.conf 中配置 security_group_api = quantum，这样通过 nova 执行的安全策略命令将会被转给 quantum。

1.5 高级功能（基于 API 扩展）

1.5.1 提供者网络

提供者网络（Provider Network）允许管理员将 Quantum 网络直接映射到物理网络上。这种情况下，租户直接共享一个公开的网络，并可以访问到 Internet。也可以使用到已有 Vlan 的网络中。

提供者扩展帮助管理员管理 Quantum 虚拟网络和下层的 Vlan 和隧道等。目前 OpenvSwitch 和 linux bridge 都已经支持该扩展。

1.5.1.1 术语

虚拟网络：一个 L2 的网络，支持端口来连接 nova 的实例。OpenvSwitch 和 Linux Bridge 插件支持多种实现虚拟网络的机制。

物理网络：连接各个节点之间的网络。每个物理网络可以支持多个虚拟网络。

租户网络：为某个租户分配的虚拟网络。

提供者网络：一个虚拟网络，可以直接映射到一个物理网络，提供对非 OpenStack 资源的直接访问。租户也可以访问到提供者网络。

VLAN 网络：虚拟网络，带有 IEEE 802.1Q 包头（特定的 VLAN ID，从 1 到 4094）。不同 ID 的 VLAN 在二层上彼此隔离，三层上则可能相互覆盖。

扁平网络：不包含 IEEE 802.1Q 包头的虚拟网络。

本地网络：支持在各个主机内部通信的虚拟网络，不允许跨网络通信。

GRE 网络：通过 GRE 封装形成的虚拟网络，或者被称为隧道，是通过主机的 IP 路由表来进行转发。不绑定到某个 Quantum 网络。

OpenvSwitch 和 linux Bridge 插件都支持 VLAN、扁平网络和本地网络，而目前只有 OpenvSwitch 支持提供 GRE 网络。

1.5.1.2 提供者属性

表格 5 提供者属性

属性名称	类型	默认值	描述
provider:network_type	字符串	N/A	虚拟网络实现的物理机制，包括扁平、vlan、local、gre。
provider:physical_network	字符串	如果物理网络名称为“default”，属性provider:network_type为“flat”或者“vlan”，则默认为“default”	扁平或 VLAN 虚拟网络实现的物理载体网络。
provider:segmentation_id	整数	N/A	对于 VLAN 网络是 VLAN ID，对于 GRE 网络是 tunnel 的 ID。

1.5.1.3 提供者 API 工作流程

查看网络的属性。

```
quantum net-show <name or net-id>
```

创建一个本地的提供者网络（仅管理员）。

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type local
```

创建一个扁平的提供者网络（仅管理员）。

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type flat --provider:physical_network <phys-net-name>
```

创建一个 VLAN 的提供者网络（仅管理员）。

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type vlan --provider:physical_network <phys-net-name> --provider:segmentation_id <VID>
```

创建一个 GRE 的提供者网络（仅管理员）。

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type gre --provider:segmentation_id <tunnel-id>
```

当创建扁平网络和 VLAN 网络时，<phys-net-name>必须是插件已知的。

1.5.2 L3 路由和 NAT

Quantum 提供了 API 扩展来实现抽象的 L3 路由器。这些路由器可以连通多个二层网络，并提供一个网关功能，连接多个私有的二层网络到外部的共享网络。

L3 路由器在上联到外部网络的“网关”端口上提供了基本的 NAT 功能。通过 NAT，可以支持浮动 IP，将外部 IP 跟内部的某个子网的私有 IP 相互进行映射。

1.5.2.1 L3 API 抽象

表格 6 路由器

属性名称	类型	默认值	描述
id	uuid 字符串	自动生成	路由器唯一标识 id
name	字符串	None	可读的字符串
admin_state_up	布尔	True	工作状态，是否转发包
status	字符串	N/A	是否正常运行
tenant_id_	uuid 字符串	N/A	所有者
external_gateway_info	包含 network_idkey-value 对的字典	Null	路由器提供网关服务（例如 NAT）所连接的外部网络

表格 7 浮动 IP

属性名称	类型	默认值	描述
id	uuid 字符串	自动生成	浮动 IP 唯一标识 id
floating_ip_address	字符串（IP 地址）	网络服务分配	能映射到一个内部网络 IP 的外部网络 IP
floating_network_id	uuid 字符串	N/A	分配浮动 IP 的网络
router_id	uuid 字符串	N/A	只读，连接内部端口到外网的路由器
port_id	uuid 字符串	Null	跟外部浮动 IP 关联的内部端口
fixed_ip_address	字符串（IP 地址）	Null	被外部浮动 IP 映射的内部端口的 IP
tenant_id	uuid 字符串	N/A	所有者

1.5.2.2 基本的 L3 工作流程

1.5.2.2.1 网络

创建外部网络（仅管理员）。

```
quantum net-create public --router:external=True
quantum subnet-create public 172.16.1.0/24
```

查看外部网络。

```
quantum net-list -- --router:external=True
```

1.5.2.2.2 创建路由器

内部路由器可以连接多个私有二层网络。

```
quantum net-create net1
quantum subnet-create net1 10.0.0.0/24
quantum net-create net2
quantum subnet-create net2 10.0.1.0/24
quantum router-create router1
quantum router-interface-add router1 <subnet1-uuid>
quantum router-interface-add router1 <subnet2-uuid>
```

路由器将创建一个端口，绑定子网的 gateway_ip，这个端口将挂载到该子网对应的 L2 网络上。这个路由器同时也获得一个连接到外网的端口，支持连通到外网，并支持浮动 IP。

路由器也可以仅连接到一个外部网络上，作为外部连接的一个 NAT 网关。

```
quantum router-gateway-set router1 <ext-net-id>
```

1.5.2.2.3 查看路由器信息

列出所有的路由器。

```
quantum router-list
```

查看某个特定的路由器。

```
quantum router-show <router_id>
```

列出特定路由器的所有内部接口。

```
quantum port-list -- --device_id=<router_id>
```

1.5.2.2.4 分配/删除浮动 IP。

首先要获取该 IP 对应的 VM 网卡的端口 ID。

```
quantum port-list -c id -c fixed_ips -- --device_id=ZZZ
```

创建/申请一个浮动 IP，并分配它。

```
quantum floatingip-create <ext-net-id>
quantum floatingip-associate <floatingip-id> <internal VM port-id>
```

上面两步可以合并为一步。

```
quantum floatingip-create --port_id <internal VM port-id> <ext-net-id>
```

1.5.2.2.5 查看浮动 IP

查看所有的浮动 IP。

```
quantum floatingip-list
```

为某个 VM 端口找到浮动的 IP。

```
quantum floatingip-list -- --port_id=ZZZ
```

取消浮动 IP。

```
quantum floatingip-disassociate <floatingip-id>
```

1.5.2.2.6 关闭 L3 网络

删除浮动 IP。

```
quantum floatingip-delete <floatingip-id>
```

清除网关。

```
quantum router-gateway-clear router1
```

从路由器上删除端口。

```
quantum router-interface-remove router1 <subnet-id>
```

删除路由器。

```
quantum router-delete router1
```

1.5.3 安全组

安全组和安全组规则允许管理员和租户指定允许通过端口的流量类型和方向。一个安全组包括若干个安全组规则。

在 OpenStack 网络中，当创建一个端口的时候，被关联到一个安全组，如果不指定，默认是关联到“default”组。这个组默认会禁止所有的进入（ingress）流量而允许所有的外出（egress）流量。可以添加规则到这个组中来改变默认的行为。

用户如果需要通过 nova 来管理安全组，需要在/etc/nova/nova.conf 中配置 security_group_api=quantum。

目前支持安全组的网络插件包括：Nicira NVP、Open vSwitch、Linux Bridge、NEC 和 Ryu。

当通过 nova 来管理安全组的时候，安全组会被应用到虚拟机实例的所有端口上。这是因为对于 nova 来说，安全组是基于虚拟机的；对于 quantum 来说，可以进行更细粒度的基于端口的管理。

1.5.3.1 API 抽象

表格 8 安全组

属性名称	类型	默认值	描述
id	uuid 字符串	自动生成	安全组规则唯一标识 id
name	字符串	None	名称
description	字符串	None	描述
tenant_id	uuid 字符串	N/A	所有者

表格 9 安全组规则

属性名称	类型	默认值	描述
id	uuid 字符串	自动生成	安全组规则唯一标识 id
security_group_id	uuid 字符串或整数	由网络服务分配	所属的安全组
direction	字符串	N/A	从一个 vm 所允许的流量方向
protocol	字符串	None	基于 IP 的协议（ICMP、TCP、UDP 等）
port_range_min	整数	None	端口范围开始
port_range_max	整数	None	端口范围结束
ethertype	字符串	None	L2 网包的类型（IPv4 或 IPv6 等）
remote_ip_prefix	字符串（IP cidr）	None	地址范围的 CIDR
remote_group_id	uuid 字符串或整数	由网络或计算服务分配	应用规则的源安全组
tenant_id	uuid 字符串	N/A	所有者

1.5.3.2 常见命令

为 web 服务器创建一个安全组。

```
quantum security-group-create webservers --description "security group for webservers"
```

查看安全组。

```
quantum security-group-list
```

创建安全组规则，允许 80 端口上进入流量。

```
quantum security-group-rule-create --direction ingress --protocol tcp --port_range_min 80 --port_range_max 80 <security_group_uuid>
```

列出安全组规则。

```
quantum security-group-rule-list
```

删除安全组规则。

```
quantum security-group-rule-delete <security_group_rule_uuid>
```

删除安全组。

```
quantum security-group-delete <security_group_uuid>
```

创建端口并关联到两个安全组上。

```
quantum port-create --security-group <security_group_id1> --security-group <security_group_id2> <network_id>
```

从端口删除安全组。

```
quantum port-update --no-security-groups <port_id>
```

1.5.4 负载均衡即服务

常见的工作过程为：

创建一个 load balancer 资源池。

```
quantum lb-pool-create --lb-method ROUND_ROBIN --name mypool --protocol HTTP --subnet-id <subnet-uuid>
```

关联两个 web 服务器到资源池。

```
quantum lb-member-create --address <webserver one IP> --protocol-port 80 mypool
quantum lb-member-create --address <webserver two IP> --protocol-port 80 mypool
```

创建一个健康监控器（health monitor），确保实例在给定的协议端口上运行。

```
quantum lb-healthmonitor-create --delay 3 --type HTTP --max-retries 3 --timeout 3
```

关联健康监控器到资源池。

```
quantum lb-healthmonitor-associate <healthmonitor-uuid> mypool
```

创建虚拟 IP，当通过 load balancer 访问的时候，转到资源池中的某个成员。

```
quantum lb-vip-create --name myvip --protocol-port 80 --protocol HTTP --subnet-id <subnet-uuid> mypool
```

1.6 高级配置选项

1.6.1 带插件的 Quantum 服务器

Quantum 服务器提供 API Web 服务，将 API 请求发送给后端的插件。可以通过命令行来指定 quantum-server 的配置文件。

```
quantum-server --config-file <quantum config> --config-file <plugin config>
```

大部分插件需要 SQL 支持，因此需要安装和配置 SQL 服务，指定 root 用户密码，并删除匿名用户。

```
$> mysql -u root
mysql> update mysql.user set password = password('iamroot') where user = 'root';
mysql> delete from mysql.user where user = '';
```

为插件创建数据库。

```
mysql> create database <database-name>;
mysql> create user '<user-name>'@'localhost' identified by '<user-name>';
mysql> create user '<user-name>'@'%' identified by '<user-name>';
mysql> grant all on <database-name>.* to '<user-name>'@'%';
```

之后可以修改各个插件的配置文件，一般都在$QUANTUM_CONF_DIR/plugins/目录下。

有些插件需要 L2 代理来提供实际的网络，此时所有节点上都要运行有代理，并且代理将虚拟网卡挂载到 Quantum 网络上。

```
quantum-plugin-agent --config-file <quantum config> --config-file <plugin config>
```

在使用插件前，需要完成两件任务：

确保核心的插件已经更新。

确保数据库连接正常。

下表给出了这些配置的一个例子。

表格 10 配置例子

参数	值
OpenvSwitch	
core_plugin ($QUANTUM_CONF_DIR/quantum.conf)	quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
sql_connection (in the plugin configuration file)	mysql://<username>:<password>@localhost/ovs_quantum?charset=utf8
Plugin Configuration File	$QUANTUM_CONF_DIR/plugins/openvswitch/ovs_quantum_plugin.ini
Agent	quantum-openvswitch-agent
Linux Bridge	
core_plugin ($QUANTUM_CONF_DIR/quantum.conf)	quantum.plugins.linuxbridge.lb_quantum_plugin.LinuxBridgePluginV2
sql_connection (in the plugin configuration file)	mysql://<username>:<password>@localhost/quantum_linux_bridge?charset=utf8
Plugin Configuration File	$QUANTUM_CONF_DIR/plugins/linuxbridge/linuxbridge_conf.ini
Agent	quantum-linuxbridge-agent

1.6.2 DHCP 代理

DHCP 服务器可以为网络中虚拟机分配 IP 地址。当创建了 Quantum 子网后，默认启用了 DHCP。

节点需要运行 DHCP 代理，读入配置信息：

目前 DHCP 代理使用 dnsmasp 来分配地址。

运行服务的插件，需要匹配响应的驱动配置。

表格 11 DHCP 驱动基本配置

参数	值
OpenvSwitch	
interface_driver ($QUANTUM_CONF_DIR/dhcp_agent.ini)	quantum.agent.linux.interface.OVSInterfaceDriver

Linux Bridge	
interface_driver ($QUANTUM_CONF_DIR/dhcp_agent.ini)	quantum.agent.linux.interface.BridgeInterfaceDriver

1.6.2.1 名字空间

默认情况下，DHCP 代理使用 Linux 网络名字空间来支持地址覆盖。如果节点运行的 Linux 操作系统不支持网络名字空间，则需要在配置文件中禁用。

```
use_namespaces = False
```

1.6.3 L3 代理

要想使用 L3 的转发和浮动 IP 支持，节点需要运行 L3 的代理。

```
quantum-l3-agent --config-file <quantum config>
--config-file <l3 config>
```

同样的，也需要配置驱动来匹配插件，创建路由接口。

表格 12 L3 驱动基本配置

参数	值
OpenvSwitch	
interface_driver ($QUANTUM_CONF_DIR/l3_agent.ini)	quantum.agent.linux.interface.OVSInterfaceDriver
external_network_bridge ($QUANTUM_CONF_DIR/l3_agent.ini)	br-ex
Linux Bridge	
interface_driver ($QUANTUM_CONF_DIR/l3_agent.ini)	quantum.agent.linux.interface.BridgeInterfaceDriver
external_network_bridge ($QUANTUM_CONF_DIR/l3_agent.ini)	必需为空（或者为外网的网桥名）

L3 代理需要通过 API 跟 Quantum 服务器通信，因此要进行如下的配置：

Keystone 的认证

```
auth_url="$KEYSTONE_SERVICE_PROTOCOL://$KEYSTONE_AUTH_HOST:$KEYSTONE_
AUTH_PORT/v2.0"
```

例如

```
http://10.56.51.210:5000/v2.0
```

管理用户

```
admin_tenant_name $SERVICE_TENANT_NAME
admin_user $Q_ADMIN_USERNAME
admin_password $SERVICE_PASSWORD
```

1.6.3.1 名字空间

默认情况下，L3 代理使用 Linux 网络名字空间来支持地址覆盖。如果节点运行的 Linux 操作系统不支持网络名字空间，则需要在配置文件中禁用。

```
use_namespaces = False
```

当 user_namespace 选项被禁用时，每个节点仅支持一个路由器 ID，需要在配置文件中的 router_id 来进行配置。

```
# If use_namespaces is set as False then the agent can only configure one router.
# This is done by setting the specific router_id.
router_id = 1064ad16-36b7-4c2f-86f0-daa2bcbd6b2a
```

为了获取到 id，可以运行 quantum 服务来创建一个路由器，然后将生成的路由器 ID 写到 L3 代理的配置文件中。

```
$ quantum router-create myrouter1
Created a new router:
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| admin_state_up       | True                                 |
| external_gateway_info |                                     |
| id                   | 338d42d7-b22e-42c5-9df6-f3674768fe75 |
| name                 | myrouter1                            |
| status               | ACTIVE                               |
| tenant_id            | 0c236f65baa04e6f9b4236b996555d56     |
+----------------------+--------------------------------------+
```

1.6.3.2 多个浮动 IP 池

浮动 IP 池是一个外部网络，一个浮动 IP 从子网分配，绑定到该外部网络上。每个 L3 代理最多绑定到一个外部网络，所以需要让多个 L3 代理来定义多个浮动 IP 地址池。L3 代理配置中的“gateway_external_network_id”指定了该 L3 代理所处理的外部网络。在同一主机上可以同时运行多个 L3 代理。

此外，运行多个 L3 代理时，需要确认“handle_internal_only_routers”选项只对一个 L3 代理设置为 True，其他都设置为 False。该选项默认为 True。

在启动 L3 代理前，需要创建路由器和外部网络，然后更新配置文件中外部网络的 uuid，之后启动 L3 代理。

对于第一个 L3 代理，在配置文件 l3_agent.ini 中指定 handle_internal_only_routers 为 True，之后启动代理。

```
handle_internal_only_routers = True
gateway_external_network_id = 2118b11c-011e-4fa5-a6f1-2ca34d372c35
external_network_bridge = br-ex
```

```
python /opt/stack/quantum/bin/quantum-l3-agent
  --config-file /etc/quantum/quantum.conf
  --config-file=/etc/quantum/l3_agent.ini
```

对于其他的代理，则在配置文件 l3_agent.ini 中指定 handle_internal_only_routers 为 False。

```
handle_internal_only_routers = False
gateway_external_network_id = e828e54c-850a-4e74-80a8-8b79c6a285d8
external_network_bridge = br-ex-2
```

1.6.3.3 Nova 元数据服务器支持

要使用 Nova 元数据服务（metadata service），L3 代理配置文件中的 metadata_ip 和 metadata_port 需要进行配置。从虚拟机到 Nova 元数据服务的访问通过 L3 路由器被转发到外部网络。Nova 元数据服务必须从外部网络是可访问的。Quantum 的 IP 地址覆盖支持和 Nova 元数据服务不能同时使用。

```
metadata_ip = 10.56.51.210
metadata_port = 8775
```

此外外，运行元数据服务的主机必须进行路由配置。例如，当运行在网络 172.18.1.0/24 中的虚拟机要访问 Nova 元数据服务，源 IP 地址是在上面的子网中的，需要在虚拟机上进行路由配置。

```
route add -net 172.18.11.0/24 gw $ROUTER_GW_IP
```

其中，$ROUTER_GW_IP$是 Quantum 路由器连接到外部网络的接口，这个 IP 地址可以通过如下命令获取。

```
$ quantum port-list -- --device_id <router-id> --device_owner network:router_gateway
+--------------------------------------+------+-------------------+----------------------------------------------------------------------------------------+
| id                                   | name | mac_address       | fixed_ips                                                                              |
+--------------------------------------+------+-------------------+----------------------------------------------------------------------------------------+
| b476808c-327a-4535-b64e-c1a4932ad962 |      | fa:16:3e:2b:85:d7 | {"subnet_id": "22b5e685-aa18-4716-92cc-9d23f5f5050a", "ip_address": "172.24.4.226"} |
+--------------------------------------+------+-------------------+----------------------------------------------------------------------------------------+
```

1.7 认证和鉴权

Quantum 使用 Keyston 来进行认证，所有用户的请求必须提供一个 X-Auth-Token 头，其中

有认证的口令（token）信息。口令中带有 tenant 信息，所以在请求中不需要指定 tenant_id。默认的认证配置下，只有管理员可以为用户创建资源，用户无法自己创建资源。

Quantum 支持两种认证策略：

- 基于操作（operation）：为特定的操作指定访问规范，可能在指定属性上进行细粒度控制。
- 基于资源（resource）：根据资源（目前仅网络资源）的访问配置来进行控制。

策略配置在 policy.json 文件中，一旦更新文件，策略实时生效。在配置文件中，策略和规则没有语义区别，一般策略是直接跟策略引擎匹配的，而规则作为策略实现的基础。

目前 Quantum 策略定义了如下的规则：

- 基于角色的规则（Role-based rules）：用户是否属于某个角色提交请求。
- 基于域的规则（Field-based rules）：请求中指定资源的某个域是否匹配某个值。
- 通用规则（Generic rules）：比较资源和用户的属性。

默认的 policy.json 文件内容如下：

```
{

"admin_or_owner": [["role:admin"], ["tenant_id:%(tenant_id)s"]], [1]

"admin_or_network_owner": [["role:admin"], ["tenant_id:%(network_tenant_id)s"]],

"admin_only": [["role:admin"]], "regular_user": [],

"shared": [["field:networks:shared=True"]],

"default": [["rule:admin_or_owner"]],[2]

"create_subnet": [["rule:admin_or_network_owner"]],

"get_subnet": [["rule:admin_or_owner"], ["rule:shared"]],

"update_subnet": [["rule:admin_or_network_owner"]],

"delete_subnet": [["rule:admin_or_network_owner"]],

"create_network": [],

"get_network": [["rule:admin_or_owner"], ["rule:shared"]],[3]

"create_network:shared": [["rule:admin_only"]], [4]

"update_network": [["rule:admin_or_owner"]],
```

```
"delete_network": [["rule:admin_or_owner"]],

"create_port": [],

"create_port:mac_address": [["rule:admin_or_network_owner"]],[5]

"create_port:fixed_ips": [["rule:admin_or_network_owner"]],

"get_port": [["rule:admin_or_owner"]],

"update_port": [["rule:admin_or_owner"]],

"delete_port": [["rule:admin_or_owner"]]

}
```

如果想让用户仅能定义网络和看到自己的资源，其他所有操作仅限管理员实施，则配置文件内容应该为：

```
{

"admin_or_owner": [["role:admin"], ["tenant_id:%(tenant_id)s"]],

"admin_only": [["role:admin"]], "regular_user": [],

"default": [["rule:admin_only"]],

"create_subnet": [["rule:admin_only"]],

"get_subnet": [["rule:admin_or_owner"]],

"update_subnet": [["rule:admin_only"]],

"delete_subnet": [["rule:admin_only"]],

"create_network": [],

"get_network": [["rule:admin_or_owner"]],

"create_network:shared": [["rule:admin_only"]],

"update_network": [["rule:admin_or_owner"]],
```

```
"delete_network": [["rule:admin_or_owner"]],

"create_port": [["rule:admin_only"]],

"get_port": [["rule:admin_or_owner"]],

"update_port": [["rule:admin_only"]],

"delete_port": [["rule:admin_only"]]

}
```

1.8 高级管理功能

1.8.1 日志设置

Quantum 使用 Python 的 logging 模块来管理日志。日志配置可以通过 quantum.conf 或者命令行参数来指定。默认 quantum.conf 中的参数优先级比命令行参数低。

在 quantum.conf 中设置日志用例如下：

```
[DEFAULT]
# Default log level is WARNING
# Show debugging output in logs (sets DEBUG log level output)
# debug = False

# Show more verbose log output (sets INFO log level output) if debug is False
# verbose = False

# log_format = %(asctime)s %(levelname)8s [%(name)s] %(message)s
# log_date_format = %Y-%m-%d %H:%M:%S

# use_syslog = False
# syslog_log_facility = LOG_USER

# if use_syslog is False, we can set log_file and log_dir.
# if use_syslog is False and we do not set log_file,
# the log will be printed to stdout.
# log_file =
# log_dir =
```

1.8.2 通知

1.8.2.1 通知选项

当 Quantum 创建、更新或删除网络、子网和端口的时候可以发出通知，为了支持 DHCP 代理，rabbit_notifier 驱动必须被设置。配置也是在 quantum.conf 中指定：

```
# ============ Notification System Options =====================

# Notifications can be sent when network/subnet/port are create, updated or deleted.
# There are three methods of sending notifications: logging (via the
# log_file directive), rpc (via a message queue) and
# noop (no notifications sent, the default)

# Notification_driver can be defined multiple times
# Do nothing driver
# notification_driver = quantum.openstack.common.notifier.no_op_notifier
# Logging driver
# notification_driver = quantum.openstack.common.notifier.log_notifier
# RPC driver
notification_driver = quantum.openstack.common.notifier.rabbit_notifier

# default_notification_level is used to form actual topic names or to set logging level
# default_notification_level = INFO

# default_publisher_id is a part of the notification payload
# host = myhost.com
# default_publisher_id = $host

# Defined in rabbit_notifier for rpc way, can be comma separated values.
# The actual topic names will be %s.%(default_notification_level)s
notification_topics = notifications
```

1.8.2.2 设置用例

1.8.2.2.1 日志和 RPC

下面的配置让 Quantum 服务器通过日志和 RPC 来发出通知。RPC 通知会进入“notifications.info”队列，绑定到一个在 quantum.conf 中“control_exchange”选项定义的一个主题交换（topic exchange）中。

```
# ============ Notification System Options =====================

# Notifications can be sent when network/subnet/port are create, updated or deleted.
# There are three methods of sending notifications: logging (via the
# log_file directive), rpc (via a message queue) and
# noop (no notifications sent, the default)

# Notification_driver can be defined multiple times
# Do nothing driver
# notification_driver = quantum.openstack.common.notifier.no_op_notifier
# Logging driver
notification_driver = quantum.openstack.common.notifier.log_notifier
# RPC driver
notification_driver = quantum.openstack.common.notifier.rabbit_notifier

# default_notification_level is used to form actual topic names or to set logging level
default_notification_level = INFO

# default_publisher_id is a part of the notification payload
# host = myhost.com
# default_publisher_id = $host

# Defined in rabbit_notifier for rpc way, can be comma separated values.
# The actual topic names will be %s.%(default_notification_level)s
notification_topics = notifications
```

1.8.2.2.2 多个 RPC 主题

下面的配置将让 quantum server 发送通知到多个 RPC 主题中。例子中分别发送到“notifications_one.info”和“notifications_two.info”两个队列。

```
# ============ Notification System Options =====================

# Notifications can be sent when network/subnet/port are create, updated or deleted.
# There are three methods of sending notifications: logging (via the
# log_file directive), rpc (via a message queue) and
# noop (no notifications sent, the default)

# Notification_driver can be defined multiple times
# Do nothing driver
```

```
# notification_driver = quantum.openstack.common.notifier.no_op_notifier
# Logging driver
# notification_driver = quantum.openstack.common.notifier.log_notifier
# RPC driver
notification_driver = quantum.openstack.common.notifier.rabbit_notifier

# default_notification_level is used to form actual topic names or to set logging level
default_notification_level = INFO

# default_publisher_id is a part of the notification payload
# host = myhost.com
# default_publisher_id = $host

# Defined in rabbit_notifier for rpc way, can be comma separated values.
# The actual topic names will be %s.%(default_notification_level)s
notification_topics = notifications_one,notifications_two
```

1.8.3 配额

配额可以限制使用的资源。当租户试图创建超过配额的资源时，会提示超出资源。

```
$ quantum net-create test_net
Quota exceeded for resources: ['network']
```

1.8.3.1 基本设置

默认情况下，所有的租户用同样的配额设置。配置信息在 quantum.conf 文件中，如果要取消某个资源的配额，删除对应条目即可。

```
[QUOTAS]
# resource name(s) that are supported in quota features
quota_items = network,subnet,port

# number of networks allowed per tenant, and minus means unlimited
quota_network = 10

# number of subnets allowed per tenant, and minus means unlimited
quota_subnet = 10

# number of ports allowed per tenant, and minus means unlimited
quota_port = 50

# default driver to use for quota checks
```

```
quota_driver = quantum.quota.ConfDriver
```

Quantum 也支持对 L3 资源的配额：路由器和浮动 IP。可以通过添加下面的代码到 quantum.conf 中的 QUOTAS 段中。

```
[QUOTAS]
# number of routers allowed per tenant, and minus means unlimited
quota_router = 10

# number of floating IPs allowed per tenant, and minus means unlimited
quota_floatingip = 50
```

1.8.3.2 每租户的配额设置

要使用每租户的配额，首先在 quantum.conf 中设置“quota_driver”：

```
quota_driver = quantum.extensions._quotav2_driver.DbQuotaDriver
```

配置好之后，下面的命令的输出中将包含 quotas 项：

```
$ quantum ext-list -c alias -c name
+----------+-----------------------+
| alias    | name                  |
+----------+-----------------------+
| router   | Quantum L3 Router     |
| quotas   | Quotas for each tenant |
| provider | Provider Network      |
+----------+-----------------------+
```

在 Folsom 版本中，每租户配额仅限 OpenvSwitch 和 Linux Bridge 插件支持。

下面四个命令可以来管理每租户配额。

```
  quota-delete                Delete defined quotas of a given tenant.
  quota-list                  List defined quotas of all tenants.
  quota-show                  Show quotas of a given tenant
  quota-update                Define tenant's quotas not to use defaults.
```

只有管理员能修改租户的配额。默认配额是对所有用户起效。

quota-list 命令列出所有租户的配额信息，默认配额设置的租户信息则没有列出。该命令只能由管理员角色执行。

```
$ quantum quota-list
+-----------+---------+------+--------+--------+----------------------------------+
| floatingip | network | port | router | subnet | tenant_id                        |
+-----------+---------+------+--------+--------+----------------------------------+
|          20 |       5 |   20 |     10 |      5 | 6f88036c45344d9999a1f971e4882723 |
```

```
| 25 | 10 | 30 | 10 | 10 |
bff5c9455ee24231b5bc713c1b96d422 |
+------------+---------+------+--------+--------+----------------------------------+
```

quota-show 命令则可以列出某个指定租户的配额信息。

```
$ quantum quota-show --tenant_id 6f88036c45344d9999a1f971e4882723
+------------+-------+
| Field | Value |
+------------+-------+
| floatingip | 20 |
| network | 5 |
| port | 20 |
| router | 10 |
| subnet | 5 |
+------------+-------+
```

```
$ quantum quota-show
+------------+-------+
| Field | Value |
+------------+-------+
| floatingip | 20 |
| network | 5 |
| port | 20 |
| router | 10 |
| subnet | 5 |
+------------+-------+
```

通过 quota-update 命令，可以更新租户的配额设置。

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723
--network 5
+------------+-------+
| Field | Value |
+------------+-------+
| floatingip | 50 |
| network | 5 |
| port | 50 |
| router | 10 |
| subnet | 10 |
+------------+-------+
```

同时更新多个资源的配额。

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723 --subnet
5 --port 20
```

```
+------------+-------+
| Field          | Value |
+------------+-------+
| floatingip | 50       |
| network       | 5         |
| port            | 20       |
| router         | 10       |
| subnet         | 5         |
+------------+-------+
```

更新 L3 资源配额。

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723 --
--floatingip 20
+------------+-------+
| Field          | Value |
+------------+-------+
| floatingip | 20       |
| network       | 5         |
| port            | 20       |
| router         | 10       |
| subnet         | 5         |
+------------+-------+
```

同时更新 L2 和 L3 资源配额。

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723
--network 3 --subnet 3 --port 3 -- --floatingip 3 --router 3
+------------+-------+
| Field          | Value |
+------------+-------+
| floatingip | 3         |
| network       | 3         |
| port            | 3         |
| router         | 3         |
| subnet         | 3         |
+------------+-------+
```

清除配额。

```
$ quantum quota-delete --tenant_id 6f88036c45344d9999a1f971e4882723
Deleted quota: 6f88036c45344d9999a1f971e4882723
$ quantum quota-show --tenant_id 6f88036c45344d9999a1f971e4882723
+------------+-------+
| Field          | Value |
+------------+-------+
```

```
| floatingip | 50       |
| network       | 10       |
| port           | 50       |
| router        | 10       |
| subnet        | 10       |
+------------+-------+
```

1.9 高可用性

quantum-server 和 quantum-dhcp-agent 可以通过同时运行多个实例来实现（active-active）。quantum-l3-agent 可以运行在主备模式（active-passive），避免 IP 地址冲突。

Pacemaker 可以帮助 quantum 实现 HA。

1.10 限制

- Quantum 的地址覆盖无法跟 Nova 安全组和 Nova 元数据服务器同时存在。Nova 假设每个 IP 同一时间仅被一个虚拟机使用。而 Quantum 在 allow_overlapping_ips 选项设置为 True 时支持地址覆盖。该选项默认为 False，如果打开了之后要禁用 Nova 安全组和 Nova 元数据服务。
- 不支持 nova-network --multi_host 类似功能，Nova-network 默认 L3、NAT 和 DHCP 都运行在计算节点上，nova-network --multi_host 在 L2 和 L3 上都存在很大局限。
- 运行 quantum-l3-agent 和 quantum-dhcp-agent 的主机上需要支持网络名字空间，以支持 IP 地址覆盖。一般 Linux 内核班版要新于 2.6.24，同时 iproute2 包版本新于 3.1.0。可以通过下面命令来检查是否支持网络名字空间。

```
ip netns create test-ns
ip netns exec test-ns ifconfig
```

如果要禁用网络名字空间，需要在 quantum.conf 中设置

```
allow_overlapping_ips=False
```

在 dhcp_agent.ini 和 l3_agent.ini 中设置

```
use_namespaces=False
```

如果在同一个节点上运行 L3 和 DHCP，需要启用名字空间：

```
use_namespaces=True
```

- L3 代理不支持 IPv6。目前仅支持 IPv4 转发。
- L3 代理支持有限规模的 Quantum 路由器。大量的路由或路由端口会导致数据库压力过载。需要在 l3_agent.ini 中增加 polling_interval 值。这将可能影响数据转发。
- ZeroMQ 支持还处在试验阶段。
- MetaPlugin 还处在试验阶段。
- Horizon 不支持路由器/浮动 IP。因此，必须通过 CLI 来进行配置。
- L3 路由器扩展不支持 IPv6。

1.11 附录

1.11.1 搭建 Demo

1.11.1.1 单独的 Flat Network

1.11.1.2 带有私有网络的路由网络

1.11.1.3 可扩展和高可用的 DHCP 代理

1.11.2 核心配置文件选项

1.11.2.1 quantum.conf

1.11.2.2 ovs_quantum_plugin.ini

1.11.2.3 linuxbridge_conf.ini

1.11.2.4 dhcp_agent.ini

1.11.2.5 l3_agent.ini

1.11.2.6 常见的设备管理选项

1.11.3 插件标记和排序支持

1.12 参考

http://docs.openstack.org/folsom/openstack-network/admin/content/index.html
http://docs.openstack.org/cli/quick-start/content/index.html
http://docs.openstack.org/trunk/openstack-network/admin/content/index.html