# [OpenStack云平台的网络模式及其工作机制](http://blog.csdn.net/hilyoo/article/details/7721401)

网络，是OpenStack的部署中最容易出问题的，也是其结构中难以理清的部分。经常收到关于[OneStack](http://onestack.googlecode.com" \t "_blank)部署网络方面问题和OpenStack网络结构问题的邮件。下面根据自己的理解，谈一谈OpenStack的虚拟网络、网络拓扑和网络流。个人理解有限，仅抛砖引玉，有问题请指正，谢谢。

**一、相关概念和要点（可跳过）**

1、OpenStack中nova-network的作用

OpenStack平台中有两种类型的物理节点，控制节点和计算节点。控制节点包括网络控制、调度管理、api服务、存储卷管理、数据库管理、身份管理和镜像管理等，计算节点主要提供nova-compute服务。控制节点的服务可以分开在多个节点，我们把提供nova-network服务的节点称为网络控制器。

OpenStack的网络由nova-network（网络控制器）管理，它会创建虚拟网络，使主机之间以及与外部网络互相访问。

OpenStack的API服务器通过消息队列分发nova-network提供的命令，这些命令之后会被nova-network处理，主要的操作有：分配ip地址、配置虚拟网络和通信。

区分以下两个概念：控制节点和网络控制器

在最简单的情况下，所有服务都部署在一个主机，这就是all-in-one；

稍微复杂点，除了nova-compute外所有服务都部署在一个主机，这个主机进行各种控制管理，因此也就是控制节点（本文把2个或以上节点的部署都称为“多节点”）；

但是，很多情况下（比如为了高可用性），需要把各种管理服务分别部署在不同主机（比如分别提供数据库集群服务、消息队列、镜像管理、网络控制等）。这个时候网络控制器（运行nova-network）只是控制节点群中的一部分。

2、OpenStack中network的2种ip、3种管理模式

Nova有固定IP和浮动IP的概念。固定IP被分发到创建的实例不再改变，浮动IP是一些可以和实例动态绑定和释放的IP地址。

Nova支持3种类型的网络，对应3种“网络管理”类型：Flat管理模式、FlatDHCP管理模式、VLAN管理模式。默认使用VLAN摸式。

这3种类型的网络管理模式，可以在一个ОpenStack部署里面共存，可以在不同节点不一样，可以进行多种配置实现高可用性。

简要介绍这3种管理模式，后面再详细分析。

1. Flat（扁平）： 所有实例桥接到同一个虚拟网络，需要手动设置网桥。
2. FlatDHCP： 与Flat（扁平）管理模式类似，这种网络所有实例桥接到同一个虚拟网络，扁平拓扑。不同的是，正如名字的区别，实例的ip提供dhcp获取（nova-network节点提供dhcp服务），而且可以自动帮助建立网桥。
3. VLAN： 为每个项目提供受保护的网段（虚拟LAN）。

**二、3种网络模式的工作机制**

**•Flat模式**

1）指定一个子网，规定虚拟机能使用的ip范围，也就是一个ip池（

* 分配ip不会超过这个范围，也就是配置里面的fixed\_range，比如10.0.0.1/27，那么可用ip就有32个；
* 这个网络是可以改变的，比如配置好节点nova.conf和interfaces后，nova-manage network delete 10.0.0.1/27 1 32；nova-manage network create192.168.1.0/24 1 255

）；

2）创建实例时，从有效ip地址池接取一个IP，为虚拟机实例分配，然后在虚拟机启动时候注入虚拟机镜像（文件系统）；

3）必须手动配置好网桥（br100），所有的系统实例都是和同一个网桥连接；网桥与连到网桥的实例组成一个虚拟网络，nova-network所在的节点作为默认网关。比如flat\_interface=eth1;eth1的ip为10.0.0.1，其它网络ip在10.0.0.1/27内。flat interface-->br100-->flat network

4）此后，网络控制器（nova-network节点）对虚拟机实例进行NAT转换，实现与外部的通信。

注意：目前好像配置注入只能够对Linux类型的操作系统实例正常工作，网络配置保存在/etc/network/interfaces文件。

**•Flat DHCP模式**

与Flat模式一样，从ip池取出ip分配给虚拟机实例，所有的实例都在计算节点中和一个网桥相关。不过，在这个模式里，控制节点做了更多一些的配置，尝试和以太网设备(默认为eth0)建立网桥，通过dhcp自动为实例分配flat网络的固定ip，可以回收释放ip。

1）网络控制器（运行nova-network服务的节点）运行dusmasq作为DHCP服务器监听这个网桥；

2）实例做一次dhcp discover操作，发送请求；

3）网络控制器把从一个指定的子网中获得的IP地址响应给虚拟机实例；

4）实例通过网络控制器与外部实现互相访问。

**•VLAN网络模式**

OpenStack的默认网络管理模式，没有设置--network\_manager=nova.network.manager.FlatDHCPManager或者FlatManager的时候默认为vlan。为了实现多台机器的安装，VLAN网络模式需要一个支持VLAN标签(IEEE 802.1Q)的交换机（switch）。

在这个模式里，为每个项目创建了VLAN和网桥。所有属于某个项目的实例都会连接到同一个VLAN，必要的时候会创建Linux网桥和VLAN。

每个项目获得一些只能从VLAN内部访问的私有IP地址，即私网网段。每个项目拥有它自己的VLAN，Linux网桥还有子网。被网络管理员所指定的子网都会在需要的时候动态地分配给一个项目。

1）网络控制器上的DHCP服务器为所有的VLAN所启动，从被分配到项目的子网中获取IP地址并传输到虚拟机实例。

2）为了实现用户获得项目的实例，访问私网网段，需要创建一个特殊的VPN实例（代码名为cloudpipe，用了创建整数、key和vpn访问实例）。

3）计算节点为用户生成了证明书和key，使得用户可以访问VPN，同时计算节点自动启动VPN。

4）vpn访问。

**Flat与vLAN的比较**

在两种Flat模式里，网络控制器扮演默认网关的角色，实例都被分配了公共的IP地址（扁平式结构，都在一个桥接网络里）。

vLAN模式功能丰富，很适合提供给企业内部部署使用。但是，需要支持vLAN的switches来连接，而且相对比较复杂，在小范围实验中常采用FlatDHCP模式。

三、**详解FlatDHCP模式**（Flat模式类似，只是少了dhcp的部分而已，就略过了）

可以有多种部署方式，比如为了实现高可用性，可以使用多网卡、外部网关、multi\_host 等方法。这里主要介绍基本的部署方式（一个控制节点，或者说一个网络控制器）。

1、网卡与节点

由于网卡和节点数的不同，可以简单分为：单节点（all-in-one）单网卡、多节点单网卡、多节点单网卡、多节点多网卡

单节点的情况下，网络控制器（运行nova-network）与计算（运行nova-compute，或者更确切的说，运行虚拟机实例）部署在一个主机。这样就不需要控制节点与计算节点之间的通信，也就少了很多网络概念，这也是入门者常用的方式。

多节点时，网络控制器与计算节点分别在不同主机，普通部署方式下（不是multi\_host），只有nova-network控制网络，而它仅仅在控制节点运行。因此，所有计算节点的实例都需要通过控制节点来与外网通信。

单网卡时，网卡需要作为public网络的接口使用，也需要作为flat网络的接口，因此需要处于混杂模式。不过建立的网络与双网卡类似，都分为flat网络和public网络。

使用单网卡，需要在nova.conf中使public\_interface和flat\_interface都为eth0。

2、网络流

如上面分析，在普通部署方式下，只有一个控制节点（或网络控制器），dhcp和外网访问都需要经过它。

dhcp时：

1）网络控制器（运行nova-network服务的节点）一直运行dusmasq作为DHCP服务器监听网桥（br100）；

2）实例做一次dhcp discover操作，发送请求；

3）网络控制器把从一个指定的子网中获得的IP地址响应给虚拟机实例。

实例访问外网时：

1）实例经过所在主机的flat\_interface（这是一个flat网络），连接到nova-network所在的主机（控制节点）；

2）网络控制器对外出网络流进行转发。

外网访问实例时：

1）网络控制器对floating ip进行nat；

2）通过flat网络将流入数据路由给对应的实例。

下图1、图2可以比较单网卡和双网卡的网络流（traffic）情况，图2、图3可以比较单节点和多节点的网络流。

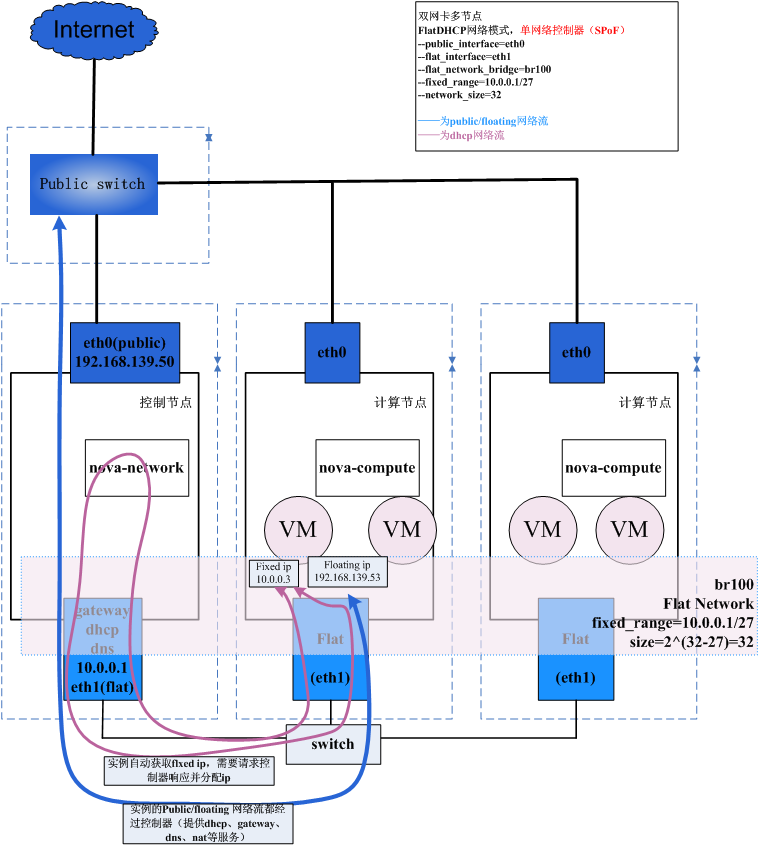


图1：双网卡多节点OpenStack网络流

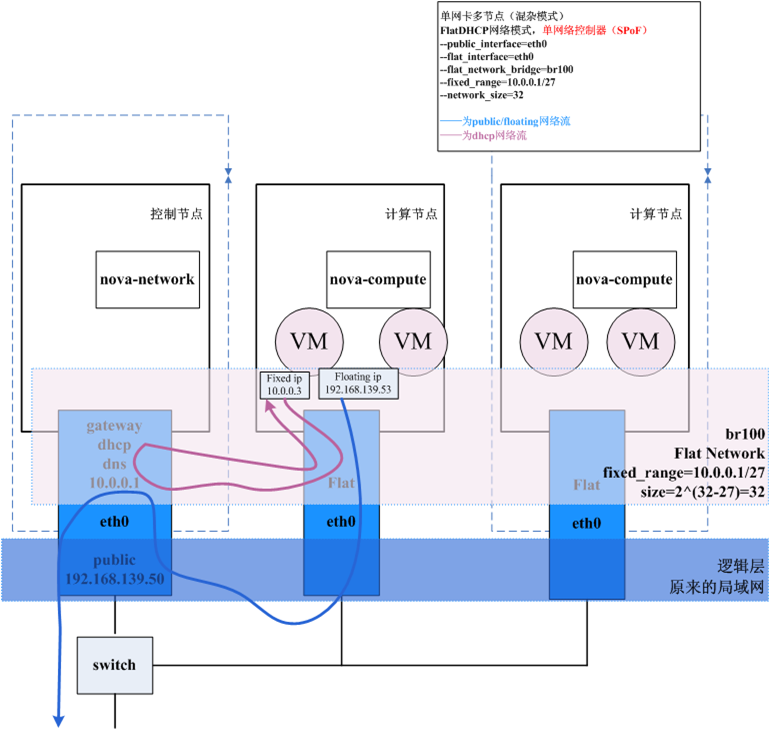


图2：单网卡多节点OpenStack网络流

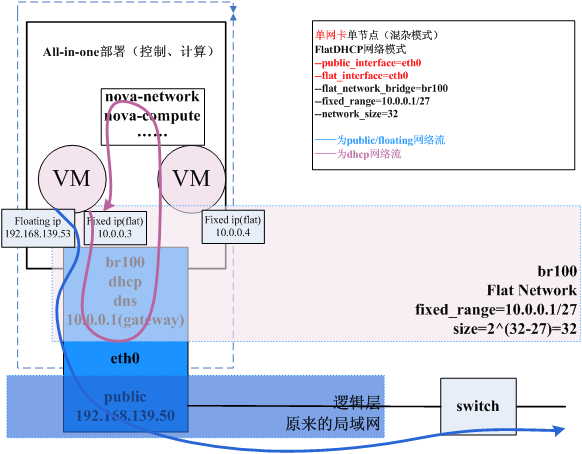


图3：单网卡单节点OpenStack网络流

3、多节点时控制节点和计算节点的工作原理

控制节点：

1）在主机上创建一个网桥（br100），把网关ip赋给这个桥；如果已经有ip，会自动把这个ip赋给网桥作为网关，并修复网关；

2）建立dhcp server，监听这个网桥；并在数据库记录ip的分配和释放，从而判定虚拟机释放正常关闭dhcp；

3）监听到ip请求时，从ip池取出ip，响应这个ip给实例；

4）建立iptables规则，限制和开放与外网的通信或与其它服务的访问。

计算节点：

1）在主机上建立一个对应控制节点的网桥（br100），把其上实例（虚拟机）桥接到一个网络（br100所在的网络）；

2）此后，这个桥、控制节点的桥和实例的虚拟网卡都在同一虚拟网络，通过控制节点对外访问。

可见，这种方式有以下特点：

1）所有实例与外网通信都经过网络控制器，这也就是SPoF（单故障点）；

2）控制节点提供dhcp服务、nat、建立子网，作为虚拟网络的网关；

3）计算节点可以没有外网ip，同其上的实例一样，可以把控制节点作为网关对外访问；

4）实例与外网通信太多，会造成控制节点网络的堵塞或者高负载。

**四、VLAN模式的特点**

VLAN模式的目的是为每个项目提供受保护的网段，具有以下特点：

* NAT实现public ip
* 除了public NAT外没有其它途径进入每个lan
* 受限的流出网络，project-admin可以控制
* 受限的项目之间的访问，同样project-admin控制
* 所以实例和api的连接通过vpn

vLAN模式了解较少，借用OpenStack文档里的一个图供对比了解：

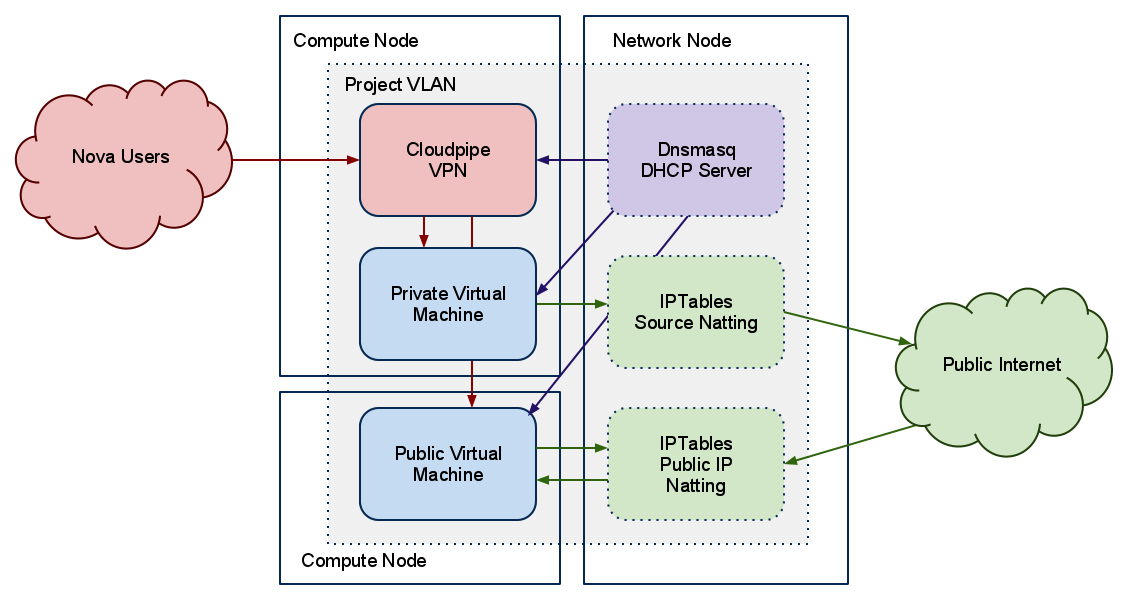


图4：VLAN模式OpenStack网络结构

**五、网络部署**

1、网络配置

apt-get install bridge-utils

安装bridge-utils就是为了建立虚拟网桥，实现虚拟网络。OpenStack会自动的创建br100这个网桥，所以不用自己创建。

--network\_manager=nova.network.manager.FlatDHCPManager

设置网络管理模式，一般使用FlatDHCP，还可以配合multi\_host实现高可用。

# Network Configuration  
--dhcpbridge\_flagfile=/etc/nova/nova.conf  
--dhcpbridge=/usr/bin/nova-dhcpbridge  
--flat\_network\_bridge=br100  
--flat\_interface=eth1  
--flat\_injected=False

--public\_interface=eth0

dhcpbridge\_flagfile指定配置文件，flat\_injected实现ipv6地址的注入，因此关闭。

flat\_network\_bridge指定网桥。

flat\_interface指定网卡，这个主机节点（一般就是控制节点）用来建立桥，桥接实例和虚拟网络以及public网络。单网卡是设为eth0，与public的同一个。

#Block of IP addresses that are fixed IPs  
--fixed\_range=10.0.0.1/27

指定ip池的范围，文中多次提到的从指定的ip池取出ip分配给实例，就是这个ip池。

**2、OpenStack中网络的高可用性（HA）**

在基本的网络管理方式中，所有实例的网络流都要经过网络控制器。当网络控制器出现问题时，网络就出现故障，网络控制器是一个SPoF（单故障点）。《[构建OpenStack的高可用性（HA，High Availability）](http://blog.csdn.net/hilyoo/article/details/7704280" \t "_blank)》简单介绍了4种方法和未来的Quantum。

主要的部署方式是FlagDHCP + multi\_host：

1）、每个计算节点安装nova-network，设置multi\_host为true。这样，每个计算节点上flat\_interface作为网桥，提供dhcp、dns，作为其上所有实例的网关（gateway）。实例不再都从控制节点经过，控制节点出现问题不会影响网络。

2）、每个计算节点的flat\_interface提供switch连接，实现实例之间的虚拟网络的传输和通信。

3）、每个计算节点有个public\_interface，与外网连接。

4）、为每个实例分配floating ip，作为实例的第二个虚拟ip，与外网通信。

也就是发生了如下的变化：

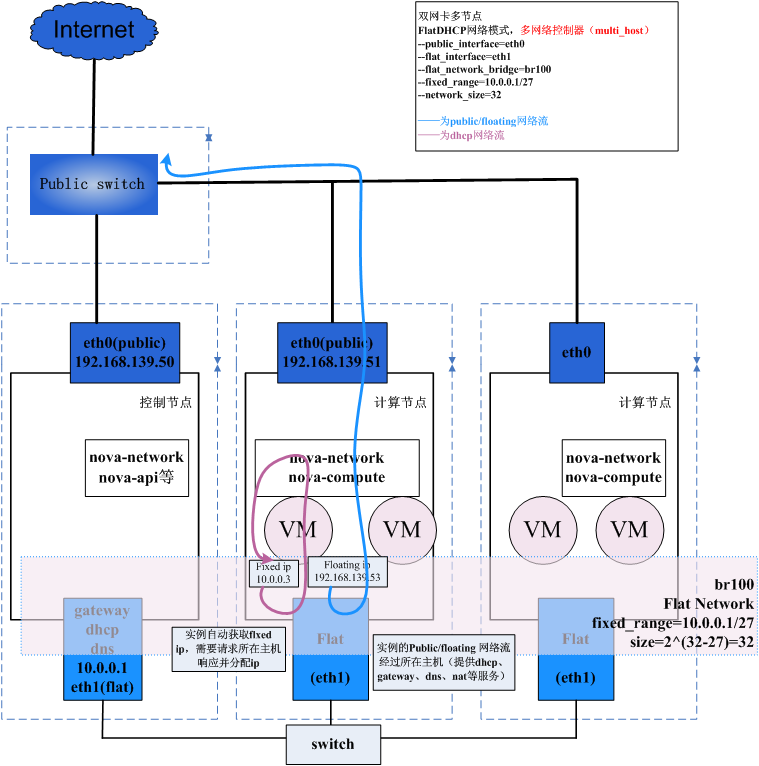


图5：multi\_host部署方式时的OpenStack网络流

未来的Quantum和Melarge提供更好的网络服务，值得期待。Quantum项目实现二层网络相关的功能，如创建和管理虚拟网络、端口等。Melange负责三层网络相关，它的主要任务是IP地址管理（IPAM）、DHCP、NAT甚至负载均衡。不过由于其实现需要一定的时间，需要多个阶段，现在还是需要了解以上的各种网络模式和部署。