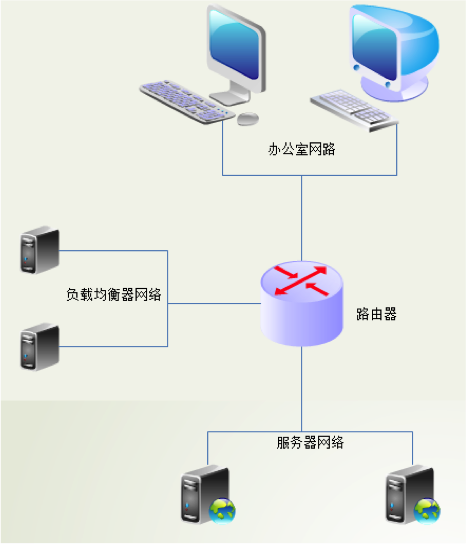
Neutron 网络之负载均衡

负载均衡在G版落户Neutron以来经历了几次大的变化。G版中实现了API模型和一个Haproxy的参考实现，H版增加 了多个agent的调度和以服务的方式重构了代码。当然，由于服务链还没有在Neutron中完全实现，所以暂时还不能看到负载均衡作为一个网络服务如何 能动态地插入到虚拟机的网络路径中去。本博客试图展示Neutron负载均衡当前功能的使用过程以及少许介绍一下背后的原理。

# 网络规划

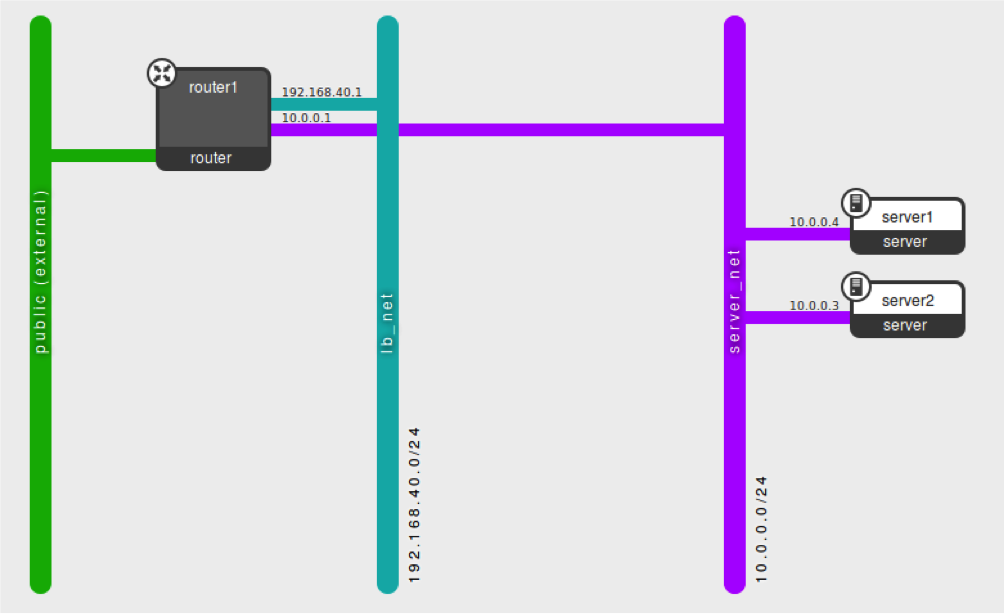
在“Neutron网络入门”中我们谈到过使用Neutron我们需要进行一些网络规划。这里我们讨论几种网络图：

[[](http://www.ustack.com/wp-content/uploads/2013/10/1.png)](http://www.ustack.com/wp-content/uploads/2013/10/1.png)

如上图所示，我们设计三种网络，负载均衡器网络中部署各种负载均衡设备，服务器网络中部署后台服器。访问者则在办公室网路中。 我们限制服务器只能被负载均衡器访问。在办公室网络中的用户只能通过负载均衡器访问服务器提供的服务。这里我们在服务器上安装的是tomcat, 监听在8080端口。

# 创建网络

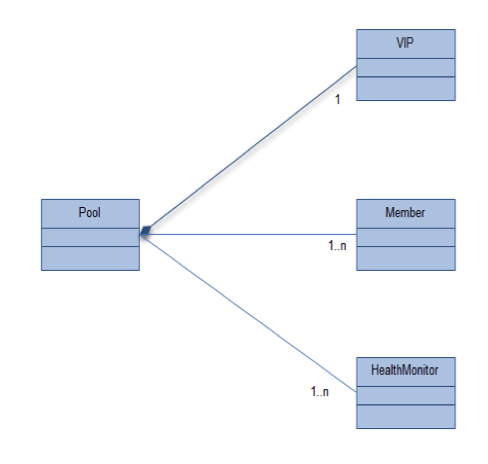
首先我们按照“Neutron网络入门”中的操作构建上面规划等价的Neutron网络：

[](http://www.ustack.com/wp-content/uploads/2013/10/2.png)

如上图所示，我们的服务器网络的网址范围为10.0.0.0/24，负载均衡器网络的网址范围是192.168.40.0 /24, public 网络链接办公网络，网址范围是192.168.10.224/28。路由器链接了所有三个网络。public网络和路由器是通过路由器的”网关臂 （Neutron API中router的gateway）”相连的。路由器把服务器网络和负载均衡器网络的IP地址SNAT成路由器的”网关臂”在public网络的地 址。这样他们就可以访问办公网络的IP啦。但是如果要想从办公网络访问服务器网络和负载均衡器网络，我们还需要动态地址(FloatingIP)。

在实际操作负载均衡器之前我们需要了解一下它的数据模型。

# 负载均衡器数据模型

[](http://www.ustack.com/wp-content/uploads/2013/10/3.png)

如上图所示，数据模型由四个对象组成。处在核心位置的是Pool（我倾向于把它命名成loadballancer）, 它代表一个负载均衡器。一个负载均衡器拥有一个VIP，也就是虚拟IP。虚拟IP中的虚拟其实是相对后面的Member而言，也就是说这个VIP不固定在 任何一个Member上。用户访问这个VIP，有时由这个成员提供服务，有时由那个成员提供服务。Member是后台提供服务的服务器。 HealthMonitor用来监控和检查后台服务器的联通情况。当检查到某个服务器不能使用时，负载均衡器就不会用它来向用户提供服务。

# 创建和配置负载均衡器

我们假设各种网络，路由器和两个具有Tomcat的虚拟机都已经创建完毕

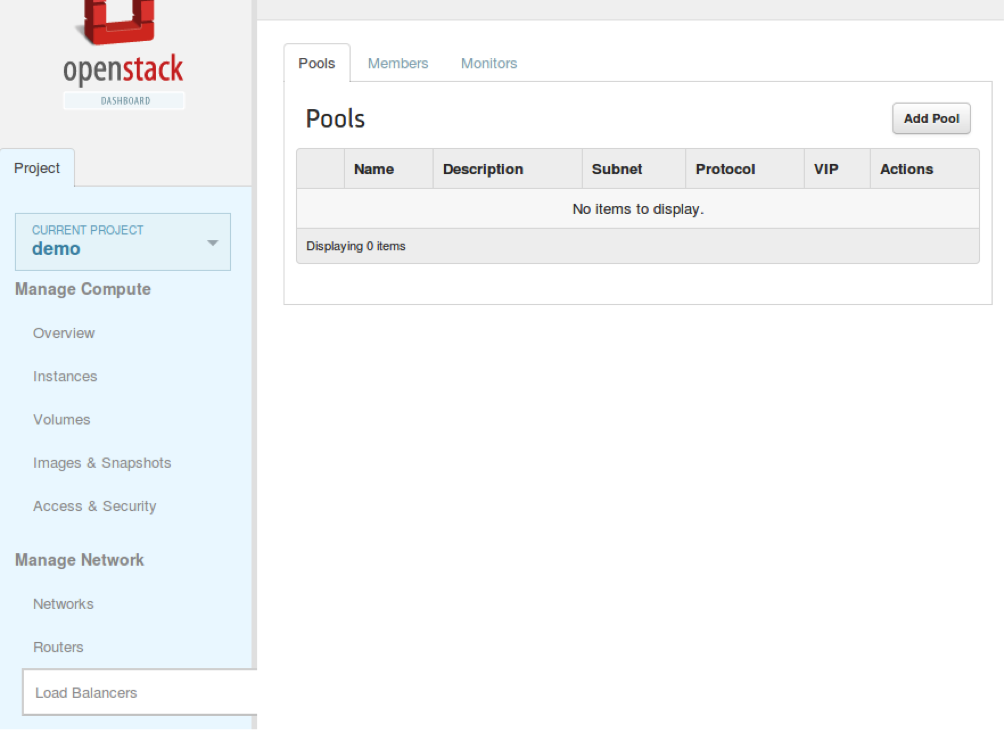
负载均衡器的操作遵循如下步骤：

1. 创建一个pool。
2. 设置VIP
3. 增加Member
4. 设置 healthmonitor
5. 配置Floating IP

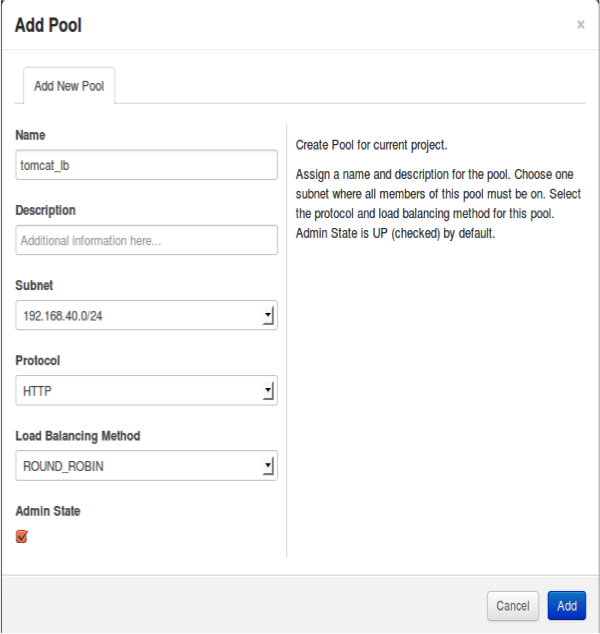
下面我们演示如何创建负载均衡器。

## 创建一个pool

1.     使用demo用户登录，并点击“Load Balancers”, 在右边的屏幕区域，我们可以看到三个标签页面： Pools, Members 和Monitors。如下图所示：

[](http://www.ustack.com/wp-content/uploads/2013/10/4.png)

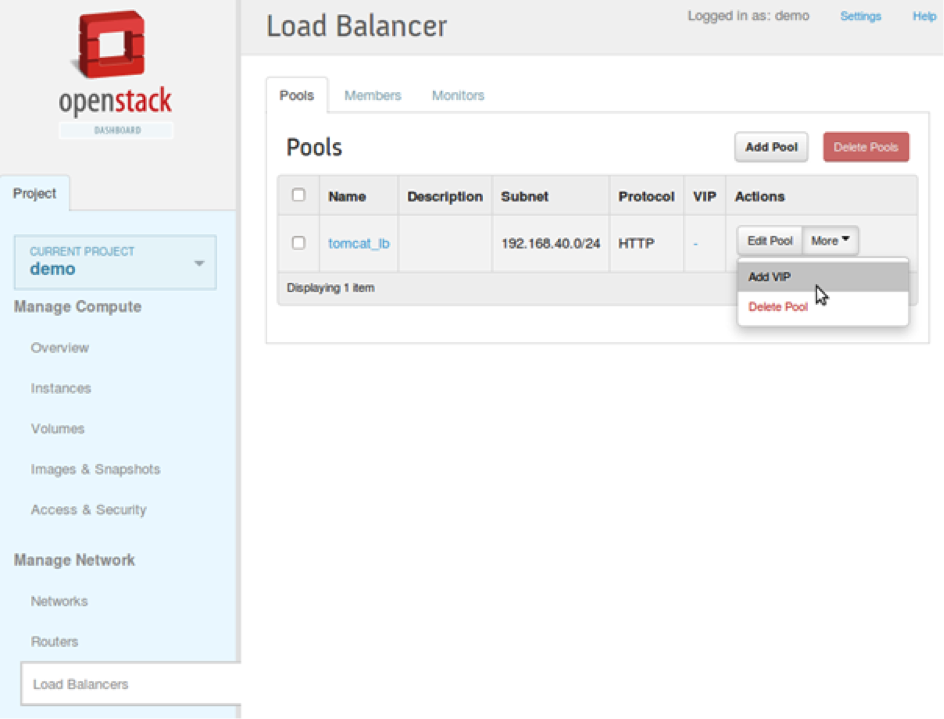
2.     点击”Add Pool”, 弹出”Add Pool”窗口，并填入相应的值：

[](http://www.ustack.com/wp-content/uploads/2013/10/5.png)

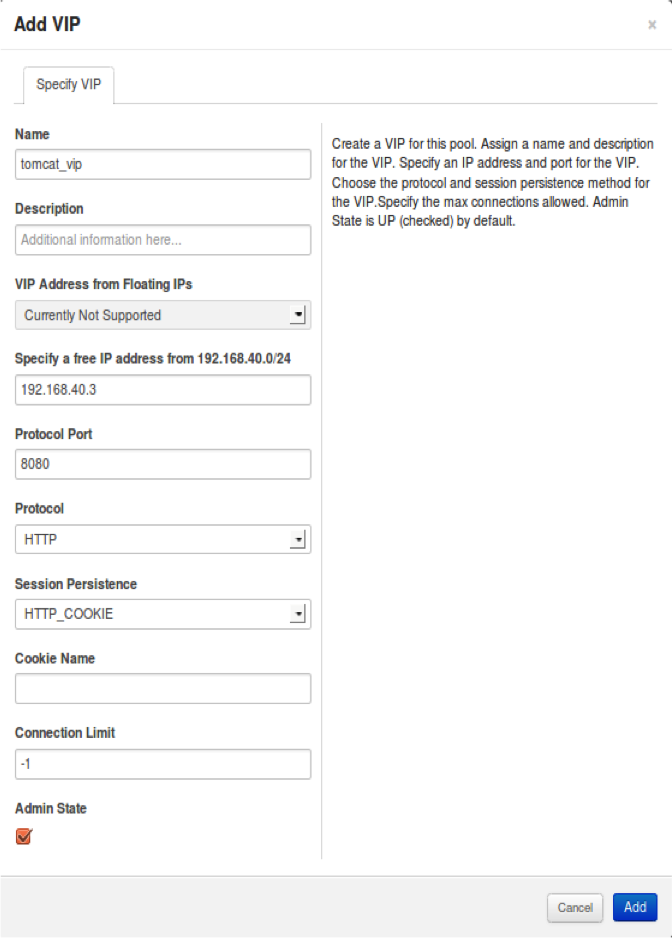
点击”Add” 关闭弹出窗口。

## 设置VIP

如下图所示，点击“More”按钮的”Add VIP”:

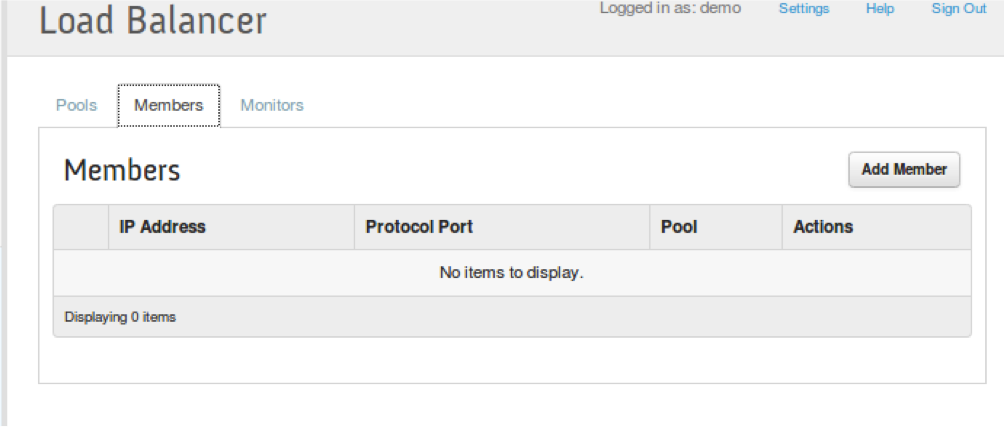
[](http://www.ustack.com/wp-content/uploads/2013/10/6.png)

在弹出的“Add VIP”窗口里，填入需要的信息：

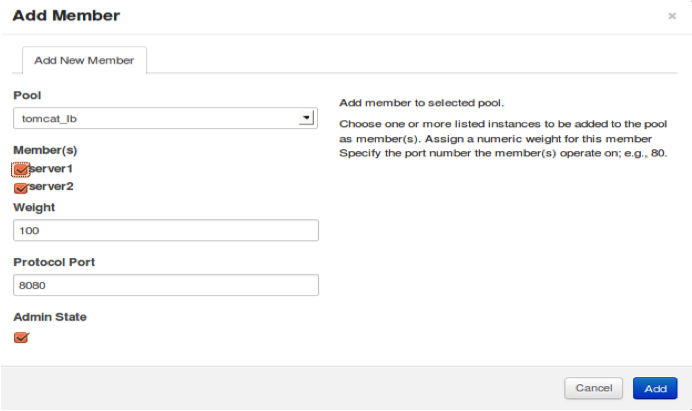
[](http://www.ustack.com/wp-content/uploads/2013/10/7.png)

## 增加Member

点击Members标签页上的”Add Member”按钮：

[](http://www.ustack.com/wp-content/uploads/2013/10/8.png)

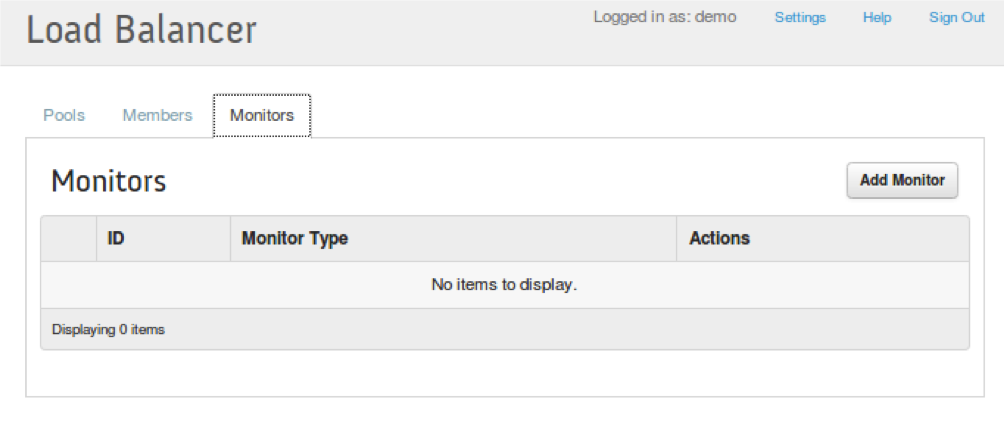
在弹出的”Add Member”对话框上，填入如下信息：

[](http://www.ustack.com/wp-content/uploads/2013/10/9.png)

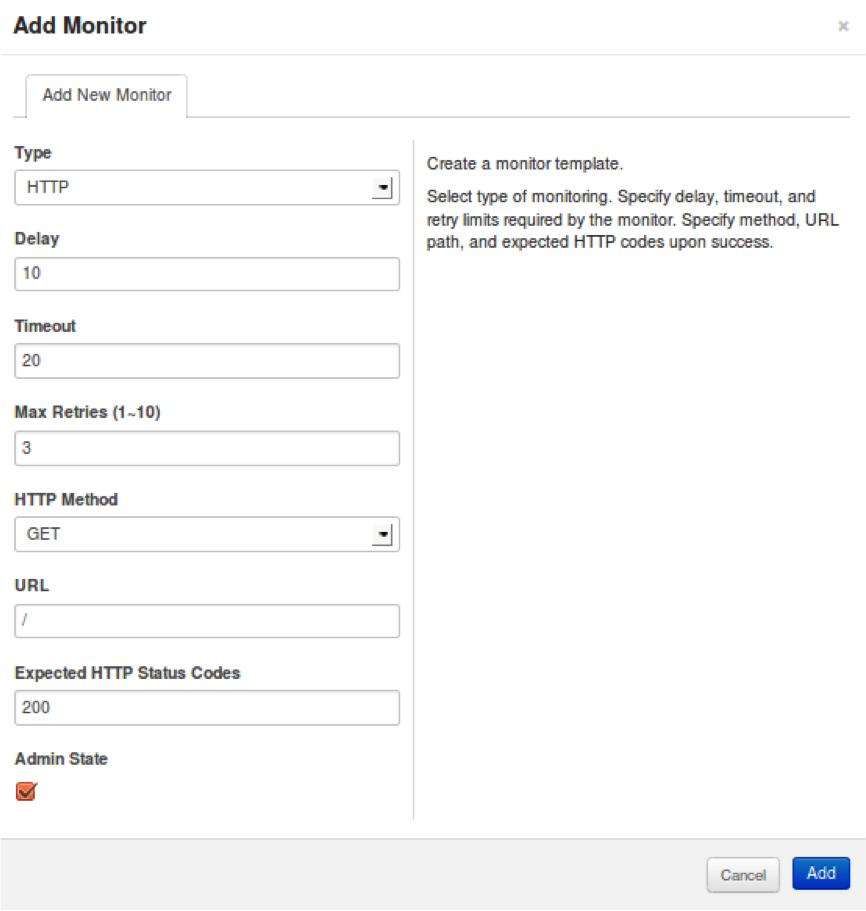
点击”Add”完成Member的设置。

## 设置 healthmonitor

点击Monitors标签页上的”Add Monitor”按钮：

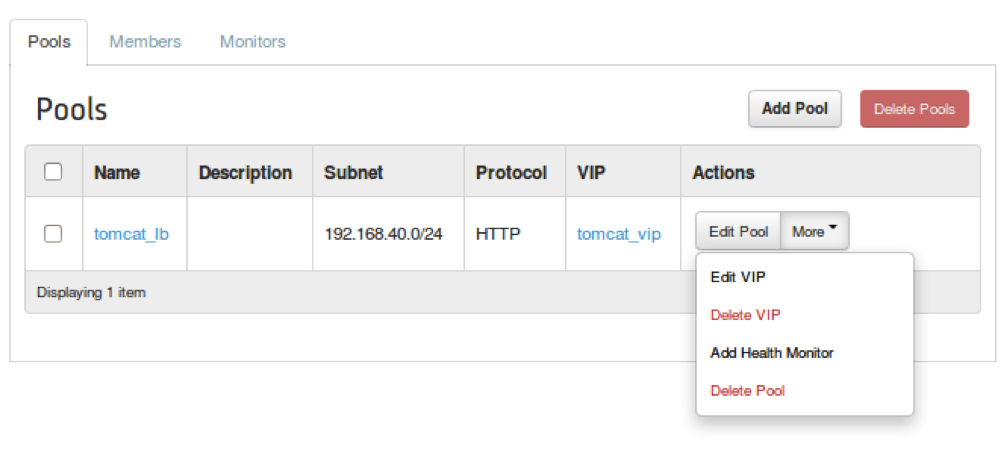
[](http://www.ustack.com/wp-content/uploads/2013/10/10.png)

在弹出的“Add Monitor”窗口上，填入如下信息：

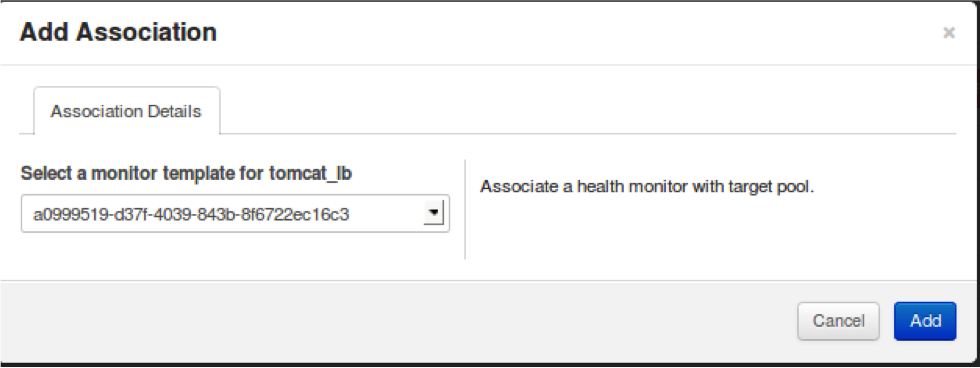
[](http://www.ustack.com/wp-content/uploads/2013/10/11.png)

点击”Add”完成。

接着我们需要把刚才创建的Monitor和我们的均衡器绑在一起。点击“tomcat\_lb”的”More”按钮菜单项“Add Health Monitor”。如下图所示：

[](http://www.ustack.com/wp-content/uploads/2013/10/12.png)

在弹出的”Add association”对话框中选择我们刚创建的monitor:

[](http://www.ustack.com/wp-content/uploads/2013/10/13.png)

点击”Add”完成绑定工作。

## 添加FloatingIP

我们说过要想外部网络访问负载均衡器，必须使用动态IP。遗憾的是，Horizon没有提供界面以给VIP绑定FloatingIP。然而我们可以通过命令行来完成这个功能。

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3 | $ neutron port-list -F id -F name | grep vip  | 7f9a7278-a9d5-4b50-b318-916b702e2e76 | vip-3a1be80c-70e0-4ae5-9bf2-25ca18ae9bae | |

上表中“7f9a7278-a9d5-4b50-b318-916b702e2e76”就是VIP地址所在端口的id。

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | $ neutron floatingip-create public  Created a new floatingip:  +---------------------+--------------------------------------+  | Field            | Value                             |  +---------------------+--------------------------------------+  | fixed\_ip\_address |                                   |  | floating\_ip\_address | 192.168.10.11                     |  | floating\_network\_id | eeb6e13a-f0fb-44b7-b895-b51e3fe32269 |  | id               | 565da56d-0ca1-4bb8-8ee7-5284bd7633bd |  | port\_id          |                                   |  | router\_id        |                                   |  | tenant\_id        | cb103c485b3a4f7f947f798cc93e45b4 |  +---------------------+--------------------------------------+  $ neutron floatingip-associate 565da56d-0ca1-4bb8-8ee7-5284bd7633bd 7f9a7278-a9d5-4b50-b318-916b702e2e76  Associated floatingip 565da56d-0ca1-4bb8-8ee7-5284bd7633bd |

# 验证

如何验证呢? 我们先来了解一下Neutron负载均衡器参考实现的部分原理。

Neutron现在提供的开源参考实现是基于Haproxy的。我们可以用下面的命令得出这个haproxy进程的启动参数：

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ ps -ef | grep haprox[y]  nobody   22602     1  0 14:01 ?     00:00:00 haproxy -f /opt/stack/data/neutron/lbaas/d7690d6d-9d8a-49b0-918c-08f91456f403/conf -p /opt/stack/data/neutron/lbaas/d7690d6d-9d8a-49b0-918c-08f91456f403/pid -sf 17471  gongysh  27892  5445  0 09:16 pts/15   00:00:27 python /usr/local/bin/neutron-lbaas-agent --config-file /etc/neutron/neutron.conf --config-file=/etc/neutron/services/loadbalancer/haproxy/lbaas\_agent.ini |

从上面的输出可以得出Haproxy的配置文件是/opt/stack/data/neutron/lbaas/d7690d6d-9d8a-49b0-918c-08f91456f403/conf。我们来看看它内容：

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | global     daemon     user nobody     group nogroup     log /dev/log local0     log /dev/log local1 notice     stats socket /opt/stack/data/neutron/lbaas/d7690d6d-9d8a-49b0-918c-08f91456f403/sock mode 0666 level user  defaults     log global     retries 3     option redispatch     timeout connect 5000     timeout client 50000     timeout server 50000  frontend 3a1be80c-70e0-4ae5-9bf2-25ca18ae9bae     option tcplog     bind 192.168.40.3:8080     mode http     default\_backend d7690d6d-9d8a-49b0-918c-08f91456f403     option forwardfor  backend d7690d6d-9d8a-49b0-918c-08f91456f403     mode http     balance roundrobin     option forwardfor     timeout check 30s     option httpchk GET /     http-check expect rstatus 200     cookie SRV insert indirect nocache     server 231f1329-8ead-4602-9fcc-260027eb622b 10.0.0.3:8080 weight 100 check inter 30s fall 3 cookie 0     server 6d936b4d-102f-46d7-80fa-4ff1d851deeb 10.0.0.5:8080 weight 100 check inter 30s fall 3 cookie 1 |

上面的内容显示了下列重要信息：

1. 在“frontend” 下的“bind 192.168.40.3:8080 ”是我们的VIP, “backend” 下的“cookie SRV insert indirect nocache ” 和“server” 行中的”cookie”对应着VIP对象的持久性方法HTTP\_COOKIE。“server” 行中的”cookie”值表示Haproxy用来记住某个客户端正在访问哪个后台服务用的。

2. 在“backend”下的“timeout check 30s ”和“option httpchk GET /”是我们的monitor对象的内容；

3. 在“backend”下的“server” 代表我们的Member对象；

4. 在“backend”下的“balance roundrobin ”代表了我们的Pool对象的负载均衡方法。

在做验证的过程中，我们还需要了解一下下面的curl指令：

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1 | $curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s |

其 中， 192.168.10.11是我们VIP地址的动态地址(FloatingIP)。“/manager/html”是Tomcat自带的管理应用，其需要 用户认证。tomcat:tomcat是Tomcat的管理应用的访问用户和口令。“-D -”表示我们要列出HTTP回应的头信息。“-o /dev/zero”表示我们要扔掉HTTP返回的数据。”-s”表示我们不想看到curl打出的进度信息。

结合Haproxy的配置信息和以上的curl指令，我们的验证过程如下：

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | $ curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Cache-Control: private  Expires: Wed, 31 Dec 1969 19:00:00 EST  Set-Cookie: JSESSIONID=EA021AC3A9CD4B42A1A02764491E9297; Path=/manager/; HttpOnly  Content-Type: text/html;charset=utf-8  Transfer-Encoding: chunked  Date: Fri, 27 Sep 2013 09:02:05 GMT  Set-Cookie: SRV=0; path=/  $ curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Cache-Control: private  Expires: Wed, 31 Dec 1969 19:00:00 EST  Set-Cookie: JSESSIONID=0D9E1E7FF5F15CFC4F1CE1330ADF726F; Path=/manager/; HttpOnly  Content-Type: text/html;charset=utf-8  Transfer-Encoding: chunked  Date: Fri, 27 Sep 2013 09:02:06 GMT  Set-Cookie: SRV=1; path=/  $ curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Cache-Control: private  Expires: Wed, 31 Dec 1969 19:00:00 EST  Set-Cookie: JSESSIONID=7A7A4D865024408591CFB5DD7912204F; Path=/manager/; HttpOnly  Content-Type: text/html;charset=utf-8  Transfer-Encoding: chunked  Date: Fri, 27 Sep 2013 09:02:09 GMT  Set-Cookie: SRV=0; path=/ |

我们可以看出SRV=0 SRV=1 在交替出现，这就是Haproxy 的 roundrobin的均衡方法。如果在访问服务器时带上SRV cookie, 而且这个服务器是可用的，Haproxy会按照要求调度：

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | $ curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s --cookie "SRV=0"  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Cache-Control: private  Expires: Wed, 31 Dec 1969 19:00:00 EST  Set-Cookie: JSESSIONID=30A2865ADE5E44F02EC74D01DDCCDA90; Path=/manager/; HttpOnly  Content-Type: text/html;charset=utf-8  Transfer-Encoding: chunked  Date: Fri, 27 Sep 2013 09:11:41 GMT  $ curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s --cookie "SRV=1"  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Cache-Control: private  Expires: Wed, 31 Dec 1969 19:00:00 EST  Set-Cookie: JSESSIONID=D043F6F46502631D9DA9D7D657FA07CA; Path=/manager/; HttpOnly  Content-Type: text/html;charset=utf-8  Transfer-Encoding: chunked  Date: Fri, 27 Sep 2013 09:11:43 GMT |

因为我们只有两个后台服务器，如果我们用SRV=3访问，Haproxy找不到它对应的服务器，于是就按照roundrobin方法选择了一台并且设置了相应的Cookie。

[?](http://www.ustack.com/blog/neutron_loadbalance/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | $ curl 192.168.10.11:8080/manager/html --user tomcat:tomcat -D - -o /dev/zero -s --cookie "SRV=2"  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Cache-Control: private  Expires: Wed, 31 Dec 1969 19:00:00 EST  Set-Cookie: JSESSIONID=B63E30E9D7A4D42ECA1D46AE9D466F39; Path=/manager/; HttpOnly  Content-Type: text/html;charset=utf-8  Transfer-Encoding: chunked  Date: Fri, 27 Sep 2013 09:11:48 GMT  Set-Cookie: SRV=0; path=/ |

# 总结

Neutron的负载均衡抽象出一个负载均衡服务的模型和API。参考实现中，在agent方生成Haproxy的配置文件然 后启动Haproxy。很明显我们不能要求这个模型能使用上Haproxy的所有功能。现在一个VIP还是在一个Haproxy节点实现，为保证足够的 HA，我们日后还 需要把Haproxy和keepalived等软件配合。

另外API和Horizon之间还有一些差距。比如API支持TCP层的负载均衡，Horizon界面上就还没做到。Neutron API支持给VIP绑定动态IP，界面上就无法达到。

在今后的发展过程中，Neutron应该会引入对相关提供商的硬件负载均衡器的支持，比如F5。